FREE eBook

LEARNING

asp.net-mvc-5

Free unaffiliated eBook created from
Stack Overflow contributors.

#asp.net-mvc-5

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: asp-net-mvc-5

It is an unofficial and free asp.net-mvc-5 ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official asp.net-mvc-5.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with asp.net-mvc-5

Remarks

This section provides an overview of what asp.net-mvc-5 is, and why a developer might want to use it.

It should also mention any large subjects within asp.net-mvc-5, and link out to the related topics. Since the Documentation for asp.net-mvc-5 is new, you may need to create initial versions of those related topics.

Examples

What's New in ASP.NET MVC 5

1. Authentication filters Are a new kind of filter added in ASP.NET MVC 5.0 .That run prior to authorization filters in the ASP.NET MVC pipeline and allow you to specify authentication logic per-action, per-controller, or globally for all controllers. Authentication filters process credentials in the request and provide a corresponding principal. Authentication filters can also add authentication challenges in response to unauthorized requests.

2.Filter overrides You can now override which filters apply to a given action method or controller by specifying an override filter.

3. Attribute routing

Install MVC5 or Update to Specific Version

To install/update MVC version, follow these steps:

1. In visual studio, open the Package Manager console (use CTRL + Q, and type package manager console)

2. In the console appearing, enter the following after the console cursor showing `PM>`:

   ```
   Install-Package Microsoft.AspNet.Mvc -Version 5.2.3
   ```

   Note: specify the version you want. In the above example we used 5.2.3 (the latest version when these instructions were written)

3. Verify the installation by using the following command in the package manager console:

   ```
   Get-Package -ListAvailable -Filter mvc
   ```

Read Getting started with asp.net-mvc-5 online: https://riptutorial.com/asp-net-mvc-5/topic/976/getting-started-with-asp-net-mvc-5

Chapter 2: Asynchronous Controller in MVC 5

Examples

Defination

Using an Asynchronous Controller in ASP.NET MVC. The AsyncController class enables you to write asynchronous action methods. You can use asynchronous action methods for long-running, non-CPU bound requests. This avoids blocking the Web server from performing work while the request is being processed.

Asynchronous Controller

```
public async Task<actionresult> Index()
{
     return  View("View", await db.UserMasers.ToListAsync());
}
```

Read Asynchronous Controller in MVC 5 online: https://riptutorial.com/asp-net-mvc-5/topic/7500/asynchronous-controller-in-mvc-5

Chapter 3: Attribute routing in mvc-5

Syntax

1. {productId:int}/{productTitle} Mapped to ProductsController.Show(int id)
2. {username} Mapped to ProfilesController.Show(string username)
3. {username}/catalogs/{catalogId:int}/{catalogTitle} Mapped to CatalogsController.Show(string username, int catalogId)

Remarks

Routing is how ASP.NET MVC matches a URI to an action. MVC 5 supports a 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.

The earlier style of routing, called convention-based routing, is still fully supported. In fact, you can combine both techniques in the same project.

Examples

How to implement attribute route

Enabling Attribute Routing To enable attribute routing, call MapMvcAttributeRoutes during configuration.

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

routes.MapMvcAttributeRoutes();

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

Optional URI Parameters and Default Values

You can make a URI parameter optional by adding a question mark to the route parameter. You can also specify a default value by using the form parameter=value.

```
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());
    }

// 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));
}
```

In this example, both /books and /books/1430210079 will route to the "View" action, the former will result with listing all books, and the latter will list the specific book. Both /books/lang and /books/lang/en will be treated the same.

Route Prefixes

Often, the routes in a controller all start with the same prefix. For example:

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

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) { … }
}
```

Use a tilde (~) on the method attribute to override the route prefix if needed:

```
[RoutePrefix(“reviews”)]
public class ReviewsController : Controller
{
    // eg.: /spotlight-review
    [Route(“~/spotlight-review”)]
    public ActionResult ShowSpotlight() { … }

    …
}
```

Default Route

You can also apply the [Route] attribute on the controller level, capturing the action as a parameter. That route would then be applied on all actions in the controller, unless a specific [Route] has been defined on a specific action, overriding the default set on the controller.

```
[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) { … }
}
```

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 chosen.

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

Read Attribute routing in mvc-5 online: https://riptutorial.com/asp-net-mvc-5/topic/6370/attribute-routing-in-mvc-5

Chapter 4: Create Html Helpers

Introduction

Html helpers are a very useful way of creating html elements in views using MVC framework. With a bit of time your team can really benefit from using them. It helps with keeping the code clean and error prone.

Remarks

To use the helpers you need to first add a `@using` directive inside the view, or add the namespace inside the `Web.config` file located in the `Views` folder.

Examples

Create a simple helper - a div with a text in it

```
public static class MyHelpers
{
   public static MvcHtmlString MyCustomDiv(this HtmlHelper htmlHelper, string text,
          object htmlAttributes = null)
   {
        var mainTag = new TagBuilder("div");
        mainTag.MergeAttributtes(htmlAttributes);
        mainTag.AddCssClass("some custom class");
        mainTag.SetInnerHtml(text);
        return MvcHtmlString.Create(mainTag.ToString());
   }
}
```

To use it in the views:

```
@Html.MyCustomDiv("Test inside custom div");
@Html.MyCustomDiv("Test inside custom div", new {@class="some class for the div element"});
```

Disposable Helper (like Html.BeginForm)

1. First create a disposable class:

```
public class MyDisposableHelper: IDisposable
{
   private bool _disposed;
   private readonly ViewContext _viewContext;

    public MyDisposableHelper(ViewContext viewContext)
    {
        if (viewContext == null)
        {
```

```
            throw new ArgumentNullException(nameof(viewContext));
        }
        _viewContext = viewContext;
    }

    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this);
    }

    protected virtual void Dispose(bool disposing)
    {
        if (_disposed)
            return;
        _disposed = true;
        _viewContext.Writer.Write("</div>");
    }

    public void EndForm()
    {
        Dispose(true);
    }
 }
```

This class inherits `IDisposable` because we want to use the helper like `Html.BeginForm(...)`. When disposed it closes the `div` created when we called the helper in the view.

2. Create the `HtmlHelper` extension method:

```
public static MyDisposableHelper BeginContainingHelper(this HtmlHelper htmlHelper)
{
     var containingTag = new TagBuilder("div");
     //add default css classes, attributes as needed
    htmlHelper.ViewContext.Writer.Write(containingTag .ToString(TagRenderMode.StartTag));
    return new MyDisposableHelper (htmlHelper.ViewContext);
}
```

What to notice here is the call to `Writer.Write` to write to the response page out custom element. The `TagRenderMode.StartTag` is used to inform the writer not to close the div just yet, because we are gonna close it when disposing the `MyDisposableHelper` class.

3. To use it in the view:

```
@using (Html.BeginContainingHelper()) {
    <div>element inside our custom element</div>
}
```

Read Create Html Helpers online: https://riptutorial.com/asp-net-mvc-5/topic/8922/create-html-helpers

Credits

S. No	Chapters	Contributors
1	Getting started with asp.net-mvc-5	Community, Hamzawey, hasan, Meghraj, Rosdi Kasim, Twister1002
2	Asynchronous Controller in MVC 5	vicky
3	Attribute routing in mvc-5	Anik Saha, PedroSouki
4	Create Html Helpers	Mihail Stancescu