Real World ASP.NET Core Middleware

Steve Smith

Ardalis.com

@ardalis

Tweet Away!

- Live Tweeting and Photos are encouraged
- Questions and Feedback are welcome
- Use #DevIntersection and/or #RealWorldMiddleware

Pluralsight

I have some 1-month free passes; see me after if you'd like one

Pair Programming
Beginner 2h 29m 7 Apr 2016

Domain-Driven Design Fundamentals
Intermediate 4h 16m 24 Jun 2014

Refactoring Fundamentals
Intermediate 8h 1m 13 Dec 2013

Creating N-Tier Applications in C#, Part 2
Intermediate 1h 40m 30 Dec 2012

Creating N-Tier Applications in C#, Part 1
Intermediate 2h 1m 16 Jul 2012

Kanban Fundamentals
Beginner 1h 31m 12 Feb 2012

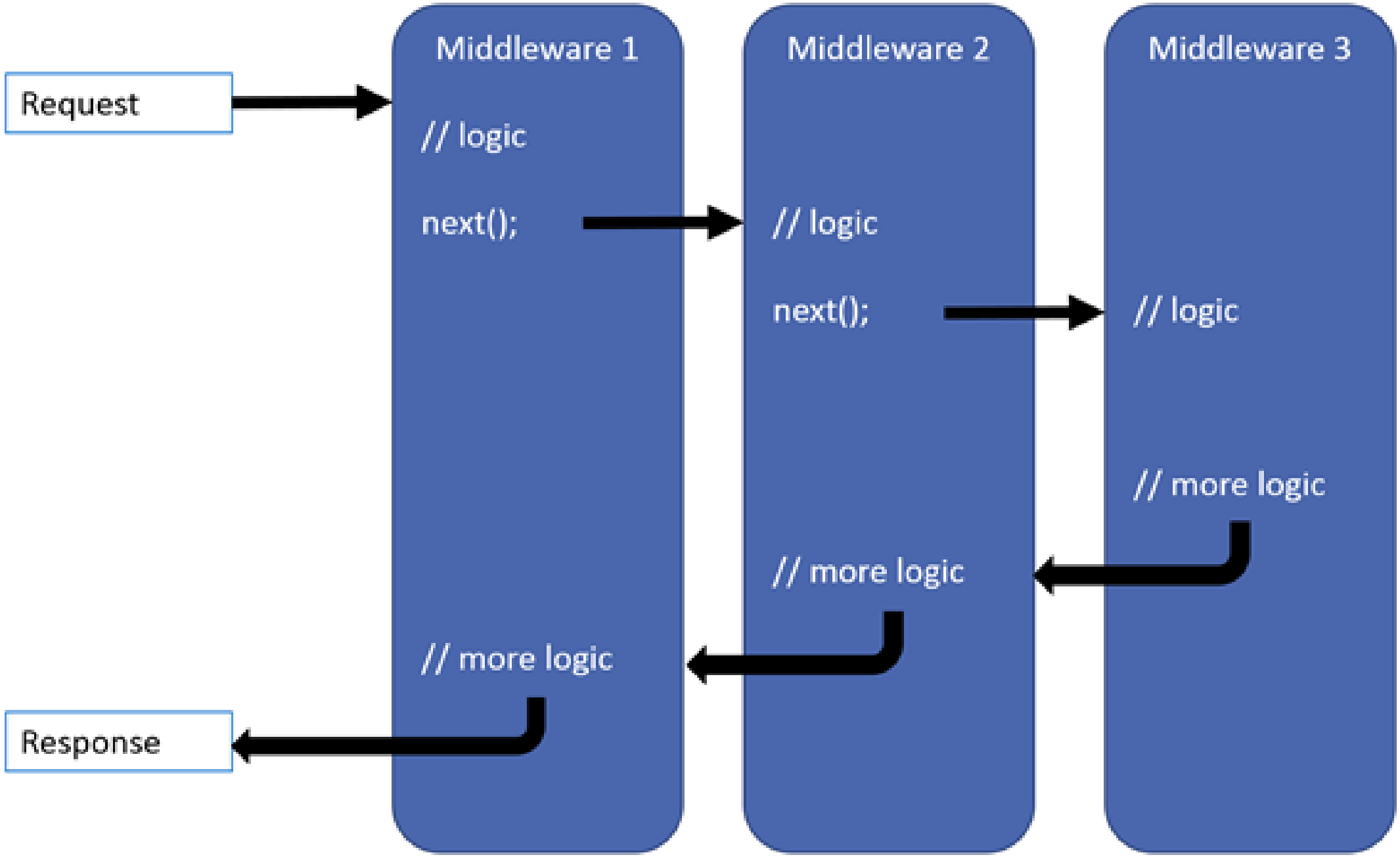
Web Application Performance and Scalability Testing
Intermediate 3h 19m 26 Jul 2011

Design Patterns Library
Intermediate 15h 38m 9 Sep 2010

SOLID Principles of Object Oriented Design
Intermediate 4h 8m 9 Sep 2010

What is Middleware?

- “Middleware are software components that are assembled into an application pipeline to handle requests and responses”
- Each component in the pipeline is a request delegate.
- Each delegate can invoke the next component in the chain, or short-circuit, returning back up the call chain


Request
Middleware 1
// logic
next();
Middleware 2
// logic
next();
Middleware 3
// logic
// more logic
// more logic
// more logic
Response

Sequence is Very Important

- The first middleware components called “wrap” all future middleware
- Think of it like Nesting Dolls
- Can act on request before and after middleware later in the pipeline

Creating Middleware in Startup: Run

```
public void Configure(IApplicationBuilder app)
{
    // app.Run terminates the pipeline without calling any later middleware
    app.Run(async context =>
    {
        await context.Response.WriteAsync("Hello, World!");
    });
}
```

Creating Middleware in Startup: Use

```
public void Configure(IApplicationBuilder app)
{
    // app.Use can execute code before and after other middleware
    app.Use(async (context, next) =>
    {
        // do some work

      // call the next middleware
        await next.Invoke();

      // do some additional work
    });
}
```

Avoid modifying HttpResponse after invoking next. Other middleware may already have begun sending the response to the client.

Creating Middleware in Startup: Map

```
public void Configure(IApplicationBuilder app)
{
    // app.Map can map a path to a function
    app.Map(“hello”, HandleHello);
    app.Run(async context =>
    {
        await context.Response.WriteAsync(“Invalid path!");
    });
}

public void HandleHello(IApplicationBuilder app)
{
    app.Run(async context =>
    {
        await context.Response.WriteAsync(“Invalid path!");
    });
}
```

Demonstration

Custom Middleware in Startup

Built-in Middleware : Diagnostics

- UseDeveloperExceptionPage
- UseDatabaseErrorPage – helps configure database
- UseBrowserLink
- UseExceptionHandler
- UseStatusCodePages
- UseRuntimeInfoPage
- UseWelcomePage
- UseElmPage / UseElmCapture (preview, requires services)

Demonstration

Diagnostic Middleware

Built-in Middleware

- UseIdentity
- UseStaticFiles
- UseDefaultFiles
- UseDirectoryBrowser (requires services)
- UseFileServer (replaces/combines previous three)
- UseRouter
- UseMvc (replaces/wraps UseRouter; requires services)
- UseRewriter (1.1)
- UseResponseCompression (1.1)
- UseResponseCaching (1.1, requires services)

Rewriter (coming in 1.1)

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{

    var options = new RewriteOptions()
        .AddRedirect("(.*)/$", "$1") // Redirect using a regular expression
        .AddRewrite(@"app/(\d+)", "app?id=$1",
skipRemainingRules: false) // Rewrite based on a Regular expression
        .AddRedirectToHttps(302, 5001) // Redirect to a different port and use HTTPS
        .AddIISUrlRewrite(env.ContentRootFileProvider, "UrlRewrite.xml") // Use IIS
UrlRewriter rules to configure
        .AddApacheModRewrite(env.ContentRootFileProvider, "Rewrite.txt"); // Use Apache
mod_rewrite rules to configure

    app.UseRewriter(options);
}
```

Moving Middleware to its own classes

```
public class MyMiddleware
{
  private readonly RequestDelegate _next;
  public MyMiddleware(RequestDelegate next)
  {
    _next = next;
  }
  public Task Invoke(HttpContext httpContext)
  {
    return _next(httpContext);
  }
}
// Extension method used to add the middleware to the HTTP request pipeline.
public static class MyMiddlewareExtensions
{
  public static IApplicationBuilder UseMyMiddleware(this IApplicationBuilder builder)
  {
    return builder.UseMiddleware<MyMiddleware>();
  }
}
```

Moving Middleware to its own classes

```
public class MyMiddleware
{
  private readonly RequestDelegate _next;
  public MyMiddleware(RequestDelegate next)
  {
    _next = next;
  }
  public async Task Invoke(HttpContext httpContext)
  {
    await _next(httpContext);
  }
}
// Extension method used to add the middleware to the HTTP request pipeline.
public static class MyMiddlewareExtensions
{
  public static IApplicationBuilder UseMyMiddleware(this IApplicationBuilder builder)
  {
    return builder.UseMiddleware<MyMiddleware>();
  }
}
```

Detect and Fix 404s

Using Custom Middleware

Detecting and Fixing 404s

```
public async Task Invoke(HttpContext httpContext)
{
   // if we have a fixed link, redirect to it now

   await _next(httpContext);

   if (httpContext.Response.StatusCode == 404)
   {
     // record the 404 to fix later
   }
}
```

Inject Dependencies into Middleware

```
public class NotFoundMiddleware
{
  private readonly RequestDelegate _next;
  private readonly RequestTracker _requestTracker;
  private readonly ILogger _logger;
  private NotFoundMiddlewareOptions _options;

  public NotFoundMiddleware(RequestDelegate next,
    ILoggerFactory loggerFactory,
    RequestTracker requestTracker,
    IOptions<NotFoundMiddlewareOptions> options)
  {
    _next = next;
    _requestTracker = requestTracker;
    _logger = loggerFactory.CreateLogger<NotFoundMiddleware>();
    _options = options.Value;
  }
}
```

Record 404 Requests

```
_requestTracker.Record(httpContext.Request.Path); // NOTE: might not be same as
original path at this point
```

Request Tracking

```
public class RequestTracker
{
    private readonly INotFoundRequestRepository _repo;
    private static object _lock = new object();

    0 references | Steve Smith, 187 days ago | 1 author, 3 changes
    public RequestTracker(INotFoundRequestRepository repo)
    {
        _repo = repo;
    }

    1 reference | Steve Smith, 187 days ago | 2 authors, 4 changes
    public void Record(string path)...

    1 reference | Steve Smith, 187 days ago | 2 authors, 4 changes
    public IEnumerable<NotFoundRequest> ListRequests()...

    2 references | Steve Smith, 187 days ago | 1 author, 3 changes
    public NotFoundRequest GetRequest(string path)...

    1 reference | Steve Smith, 187 days ago | 1 author, 3 changes
    public void UpdateRequest(NotFoundRequest request)...
}
```

Record 404 Requests

```
public void Record(string path)
{
    lock (_lock)
    {
        var request = _repo.GetByPath(path);
        if (request != null)
        {
            request.IncrementCount();
        }
        else
        {
            request = new NotFoundRequest(path);
            request.IncrementCount();
            _repo.Add(request);
        }
    }
}
```

"Upsert" a NotFoundRequest record

NotFoundRequest

```
public class NotFoundRequest
{
    1 reference | Steve Smith, 190 days ago | 2 authors, 2 changes
    public NotFoundRequest(string path)...

    0 references | Steve Smith, 187 days ago | 1 author, 2 changes
    public NotFoundRequest()...
    8 references | Steve Smith, 187 days ago | 2 authors, 2 changes
    public string Path { get; private set; }
    4 references | Steve Smith, 192 days ago | 1 author, 1 change
    public int Count { get; private set; }
    4 references | Steve Smith, 192 days ago | 1 author, 1 change
    public string CorrectedPath { get; private set; }

    2 references | Steve Smith, 192 days ago | 1 author, 1 change
    public void IncrementCount()
    {
        Count++;
    }

    1 reference | Steve Smith, 192 days ago | 1 author, 1 change
    public void SetCorrectedPath(string path)
    {
        CorrectedPath = path;
    }
}
```

Fixing Known 404s

```
public async Task Invoke(HttpContext httpContext)
{
    string path = httpContext.Request.Path;
    _logger.LogTrace("Path: {path}");
    string correctedPath =
_requestTracker.GetRequest(path)?.CorrectedPath;
    if (correctedPath != null)
    {
      // redirect/rewrite using the corrected path
    }
    await _next(httpContext);
    // other logic
}
```

301 Permanent Redirect

- Sends a response to the browser
- Browser makes a new request to the new URL
- Changes the URL in the browser

Rewrite

- Rewrites the path in the request pipeline itself
- ASP.NET continues processing the request, with the new path
- No additional requests – rewrites are invisible to the client
- Browser URL remains unchanged

Options Pattern (Configuration Support)

```
namespace NotFoundMiddlewareSample.Middleware
{
    public enum FixPathBehavior
    {
        Redirect,
        Rewrite
    }

    public class NotFoundMiddlewareOptions
    {
        public string Path { get; set; } = "/fix404s";
        public FixPathBehavior FixPathBehavior { get; set; } =
FixPathBehavior.Redirect;
    }
}
```

Options Pattern

```
// add in Startup.ConfigureServices

services.Configure<NotFoundMiddlewareOptions>(
Configuration.GetSection("NotFoundMiddleware"));

// Options can be flexibly via Configuration
// programmatically, or from any config source

// appsettings.json
 "NotFoundMiddleware": {
    "Path": "/fixbadlinks",
    "FixPathBehavior":  "Redirect"
  }
```

Options Pattern

```
// request IOptions<T> - assign to local variable T

 private NotFoundMiddlewareOptions _options;

public NotFoundMiddleware(RequestDelegate next,
  IOptions<NotFoundMiddlewareOptions> options)
{
    _next = next;
    _options = options.Value;
}
```

Options: Interface Segregation

Options: Interface Segregation

Software Craftsmanship Calendar

DevIQ

2017 Software Craftsmanship Wall Calendar

$18.95

Add to Cart

Twelve months of motivational images and quotes for software teams and developers. This calendar features original imagery as well as some of the most popular images from previous editions of the Software Craftsmanship calendar. To learn more about the calendar, its creation, and history, check out its Kickstarter Page.

store.DevIQ.com

Tracking and Managing 404s

Fix 404s

Path	404 Count	Corrected Path
/foo	6	Save
/baz	2	Save
/bar	1	Save

NotFoundPageMiddleware

- Check if the request path is the “manage 404s” path
  - Read from Options value
  - If not, let request pass through
- Check if querystring includes any commands
  - If so, perform the operation
- Render the table to show all of the tracked NotFoundRequests

Detect the Path

```
public async Task Invoke(HttpContext httpContext)
{
    if (!httpContext.Request.Path.StartsWithSegments(_options.Path))
    {
        await _next(httpContext);
        return;
    }
...
}
```

Detect Commands and Execute

```
if (httpContext.Request.Query.Keys.Contains("path") &&
    httpContext.Request.Query.Keys.Contains("fixedpath"))
{
    var request = _requestTracker.GetRequest(httpContext.Request.Query["path"]);

    request.SetCorrectedPath(httpContext.Request.Query["fixedpath"]);

    _requestTracker.UpdateRequest(request);
}
```

Render Views from Middleware

```
private async void Render404List(HttpContext httpContext)
{
    var model = _requestTracker.ListRequests()
        .OrderByDescending(r => r.Count);
    var requestPage = new RequestPage(model);
    await requestPage.ExecuteAsync(httpContext);
}
```

Middleware Data Access

- Use separate DbContext (assuming middleware data is self-contained)
- Package up with Middleware
- Let application configure in ConfigureServices

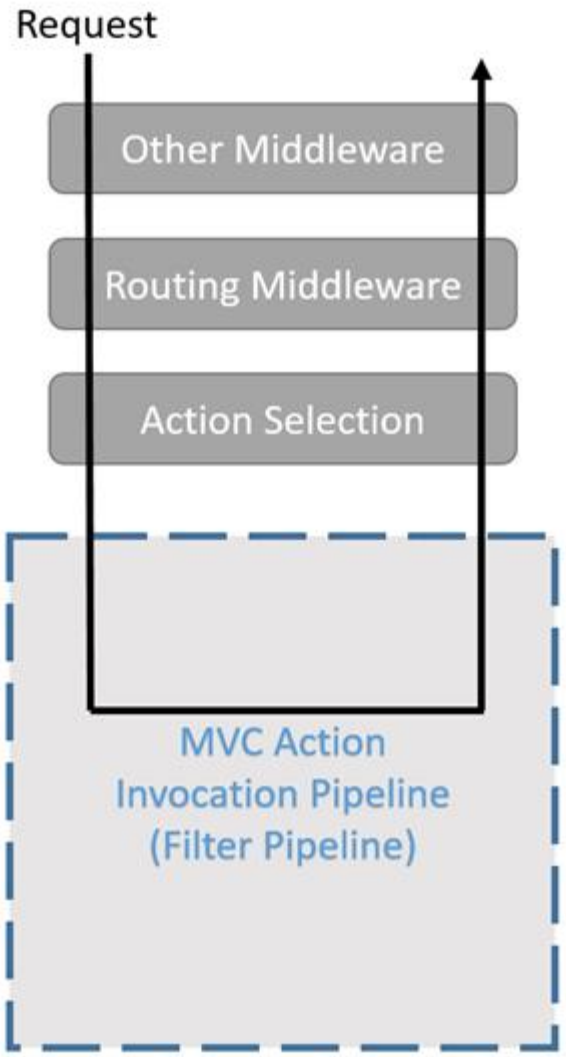
Demonstration

Not Found Middleware

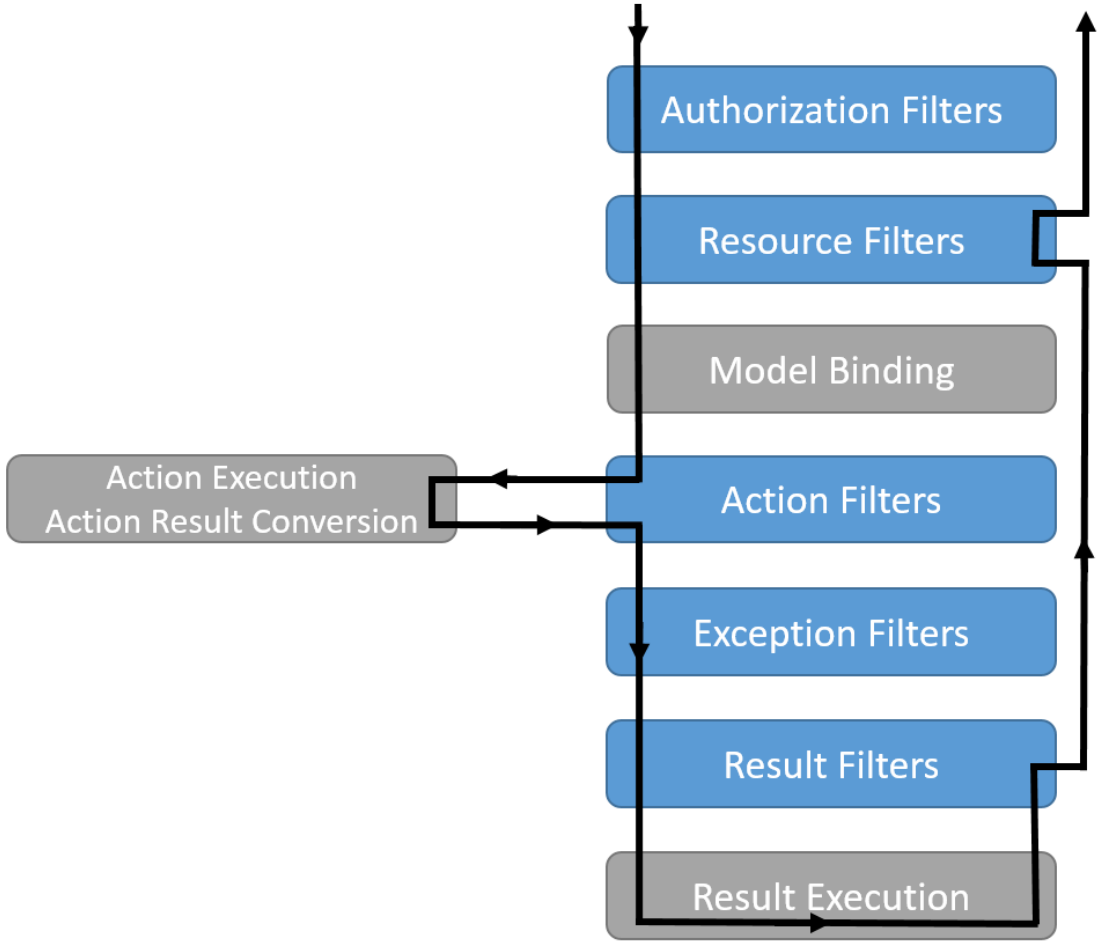
Middleware or Filters?

- MVC Filters offer similar capabilities
- Filters have MVC context:
  - Routes
  - Model Binding
  - Action Results
- Middleware is MVC-agnostic
  - Can be used on *any* ASP.NET Core app

Filters in MVC



Filters in MVC



Demonstration

Add middleware to new app

Summary

- Middleware provides a great way to add functionality to your site
- Use built-in or roll your own
- Follow best practices to support configuration, dependency injection

Questions?

Or tweet me @ardalis and I'll answer later.

DevIQ
DEVELOP INTELLIGENCE

Steve Smith
Ardalis.com
@ardalis

Thanks!

Follow us:

- @deviq
- /DevIQPage
- DevIQ.com/updates

Enroll in DevIQ for access to:

- full-length courses
- samples
- member-only events

DevIQ.com/enroll