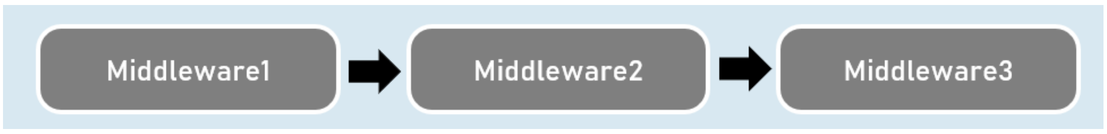
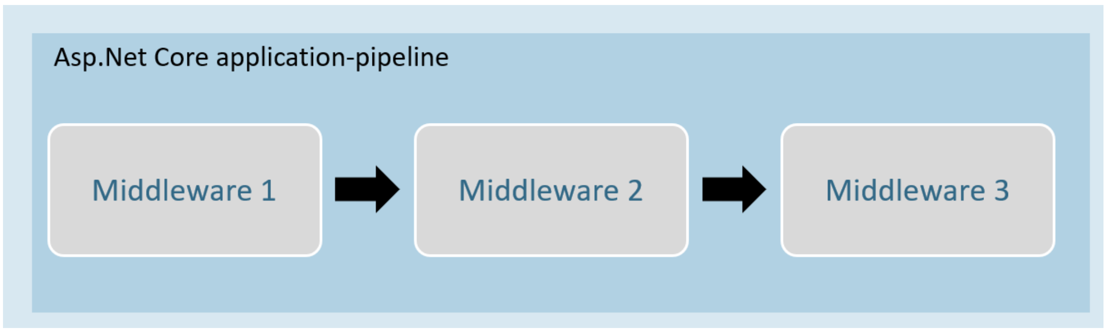
Section Cheat Sheet (PPT)

Introduction to Middleware

> Middleware is a component that is assembled into the application pipeline to handle requests and responses. Middlewares are chained one-after-other and execute in the same sequence how they're added.





Middleware can be a request delegate (anonymous method or lambda expression) [or] a class.

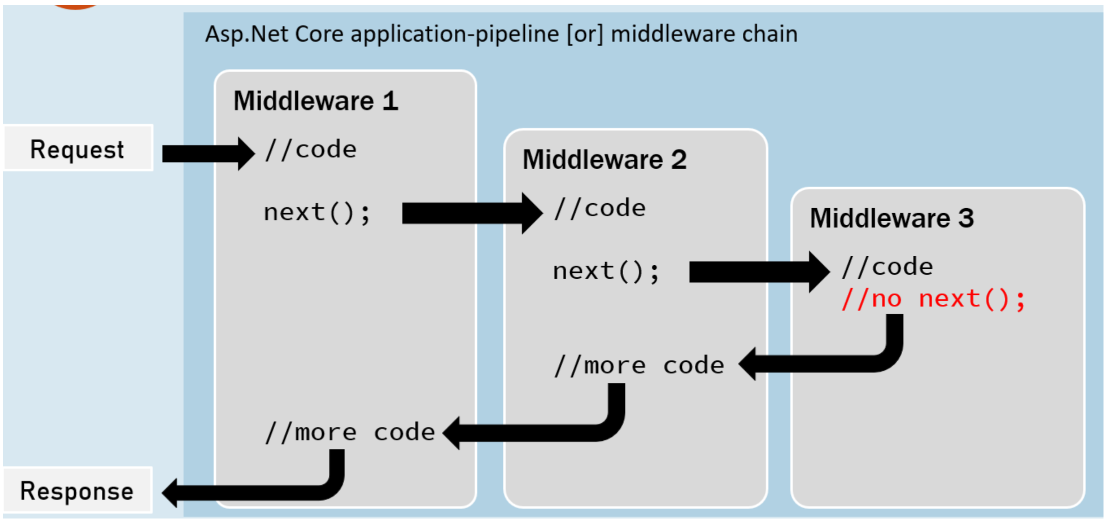
Middleware - Run

app.Run( )

```
app.Run(async (HttpContext context) =>
{
//code
});
```

The extension method called “Run” is used to execute a terminating / short-circuiting middleware that doesn’t forward the request to the next middleware.

Middleware Chain



app.Use( )

```
app.Use(async (HttpContext context, RequestDelegate next) =>
{
  //before logic
  await next(context);
  //after logic
});
```

The extension method called "Use" is used to execute a non-terminating / short-circuiting middleware that may / may not forward the request to the next middleware.

Middleware Class

Middleware class is used to separate the middleware logic from a lambda expression to a separate / reusable class.

```
class MiddlewareClassName : IMiddleware
{
  public async Task InvokeAsync(HttpContext context, RequestDelegate next)
  {
    //before logic
    await next(context);
    //after logic
  }
}
```

```
app.UseMiddleware<MiddlewareClassName>();
```

Middleware Extensions

```
class MiddlewareClassName : IMiddleware
{
  public async Task InvokeAsync(HttpContext context,RequestDelegate next)
  {
    //before logic
    await next(context);
    //after logic
  }
});
```

Middleware extension method is used to invoke the middleware with a single method call.

```
static class ClassName
{
  public static IApplicationBuilder ExtensionMethodName(this
IApplicationBuilder app)
  {
    return app.UseMiddleware<MiddlewareClassName>();
  }
}
```

```
app.ExtensionMethodName();
```

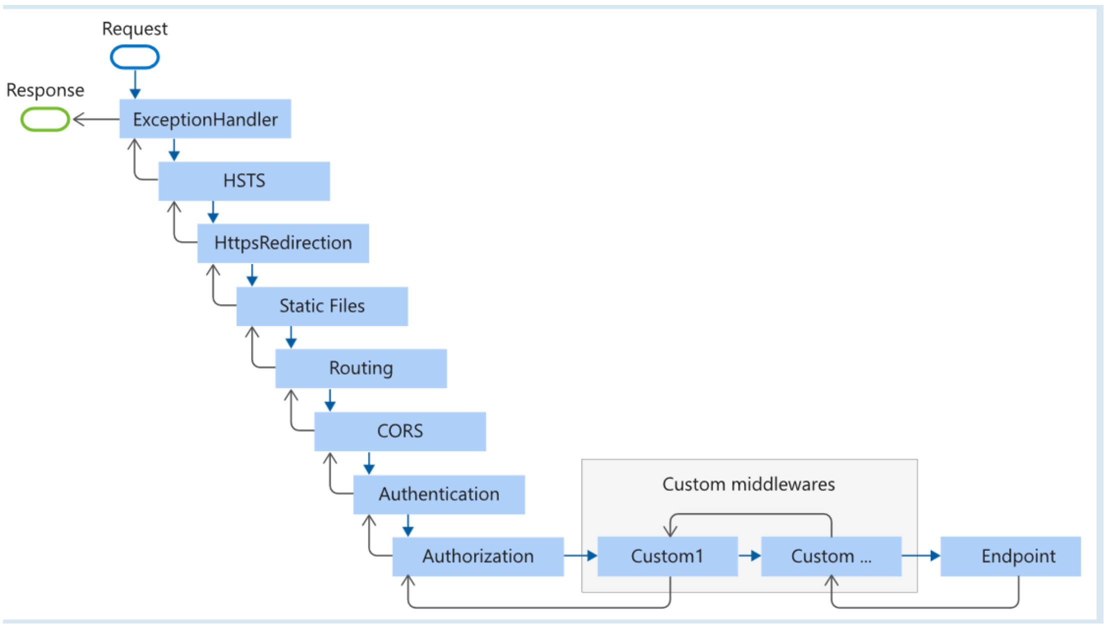
Conventional Middleware

```
class MiddlewareClassName
{
  private readonly RequestDelegate _next;

  public MiddlewareClassName(RequestDelegate next)
  {
    _next = next;
  }

  public async Task InvokeAsync(HttpContext context)
  {
   //before logic
   await _next(context);
   //after logic
  }
});
```

```
static class ClassName
{
  public static IApplicationBuilder ExtensionMethodName(this
IApplicationBuilder app)
  {
   return app.UseMiddleware<MiddlewareClassName>();
  }
}
```

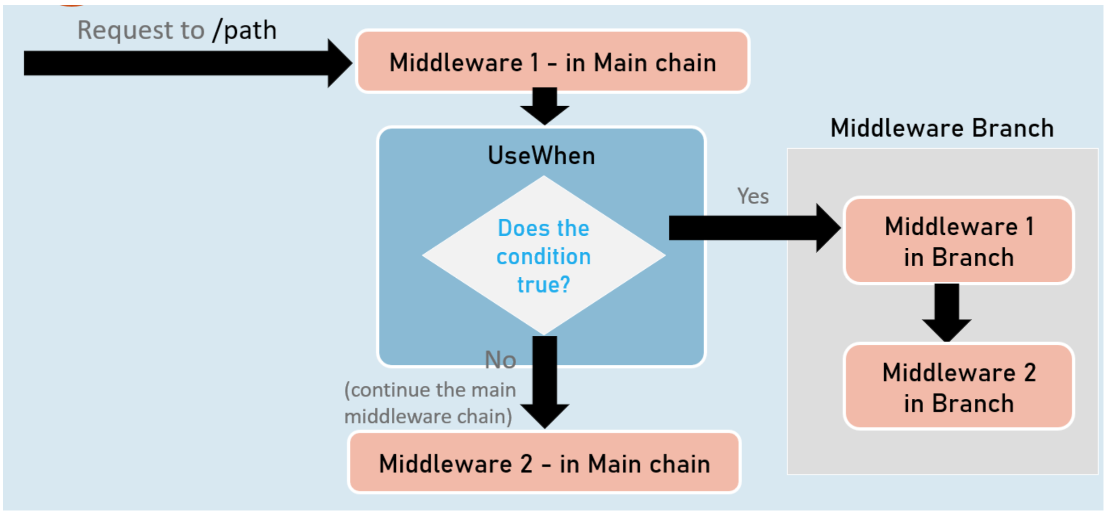
```
app.ExtensionMethodName();
```

The Right Order of Middleware



```
app.UseExceptionHandler("/Error");
app.UseHsts();
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseCors();
app.UseAuthentication();
app.UseAuthorization();
app.UseSession();
app.MapControllers();
//add your custom middlewares
app.Run();
```

Middleware - UseWhen



app.UseWhen( )

```
app.UseWhen(
  context => { return boolean; },
  app =>
  {
    //add your middlewares
  }
);
```

The extension method called “UseWhen” is used to execute a branch of middleware only when the specified condition is true.