Middleware

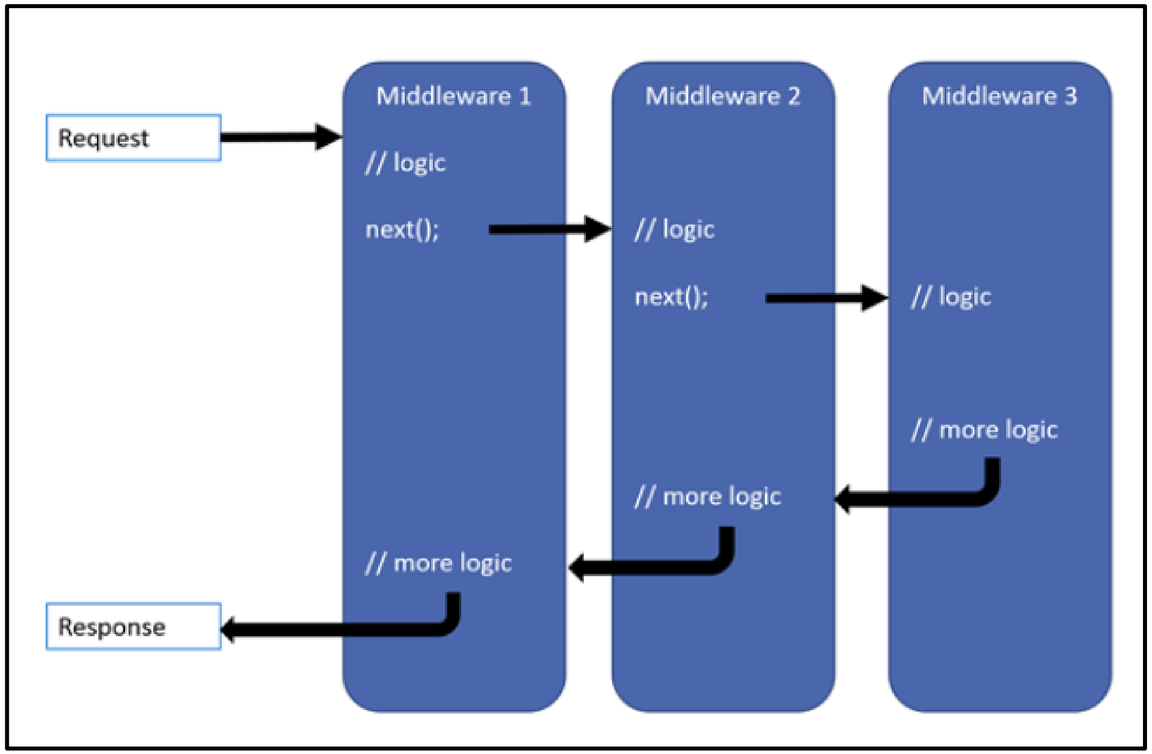
Tan Cher Wah (cherwah@nus.edu.sg)

Middleware

- In ASP.NET, Middleware refers to software components that modify or handle incoming HTTP requests and outgoing HTTP responses
- Middleware is often used to implement cross-cutting concerns such as authentication, logging, caching, and static files requests

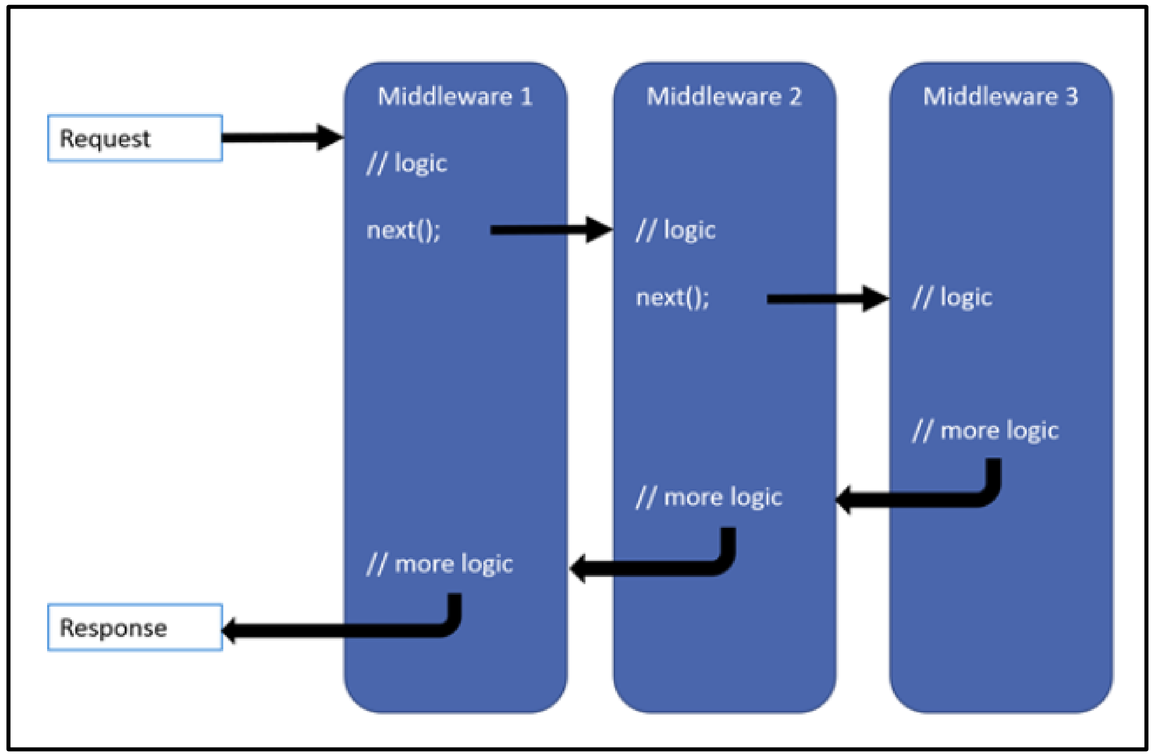
Middleware Pipeline

The diagram depicts a middleware pipeline in an ASP.NET application where each Middleware is chained to the next in line.



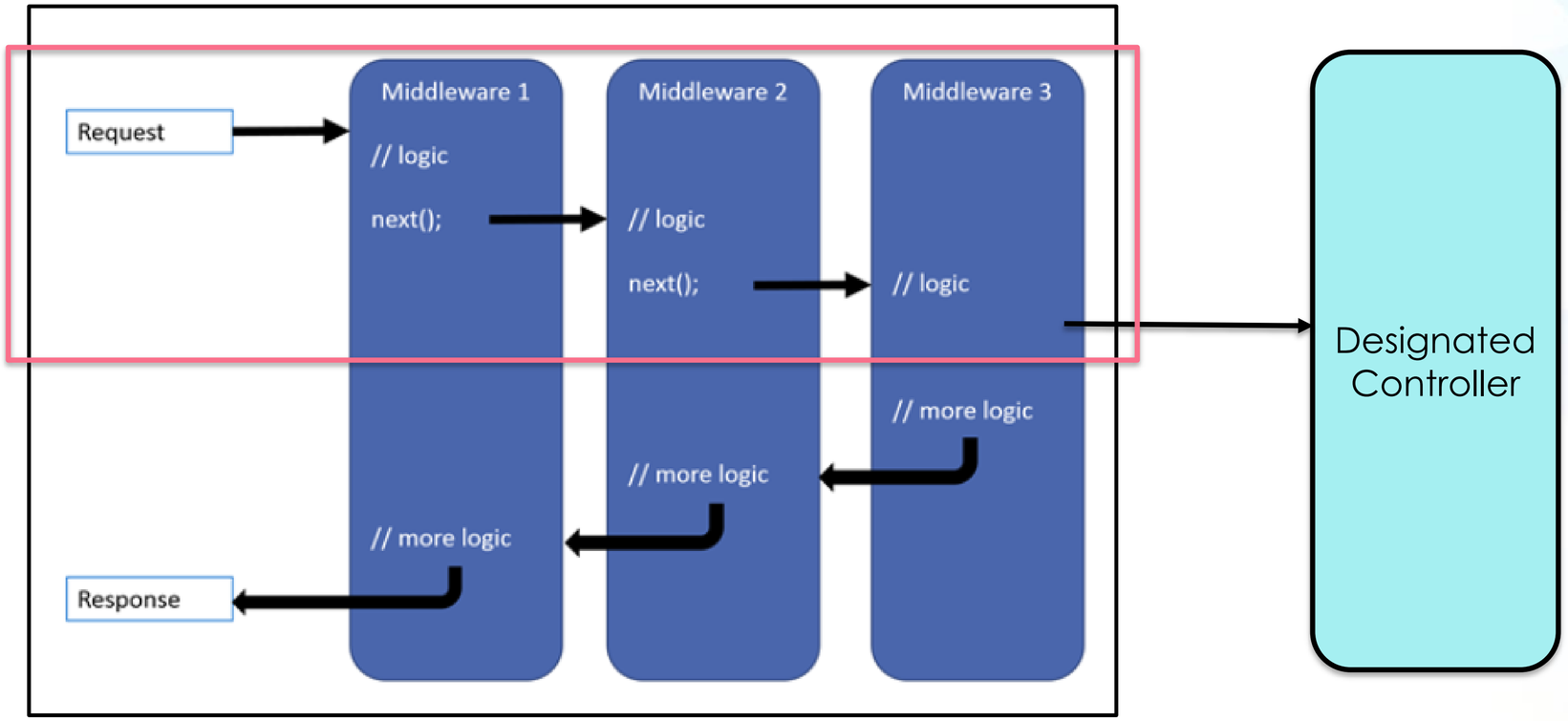
Middleware Pipeline

Each Middleware is run in turn, and it decides how it wishes to handle the current HTTP request and if it should pass the current HTTP request to the next Middleware along the pipeline



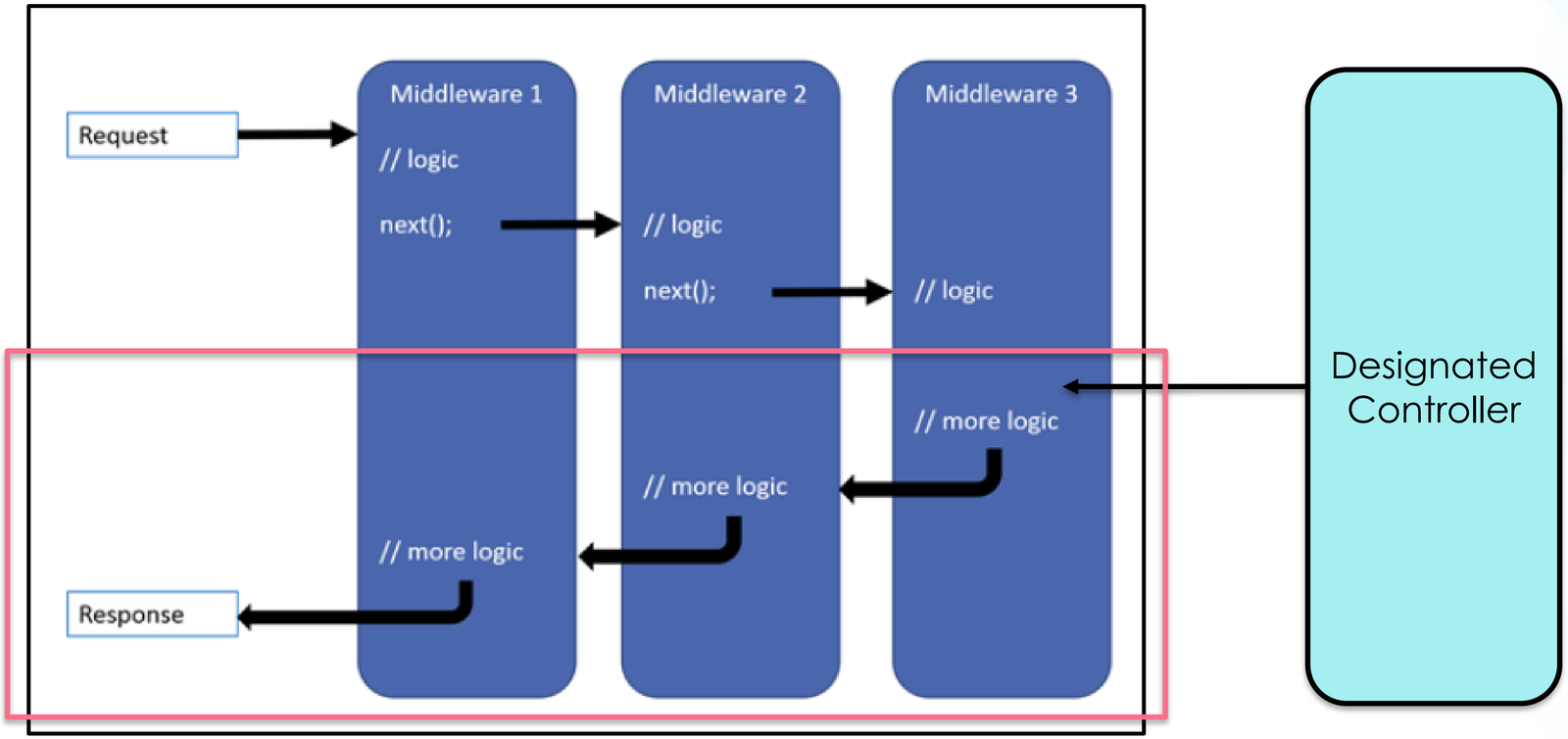
Middleware Pipeline

The request can only reach the designated controller after the final Middleware in the pipeline had the chance to handle the request



Middleware Pipeline

Each Middleware also has a chance to handle the response (from controller) such as logging the request/response time of the .NET application



Examples of Middleware in ASP.NET

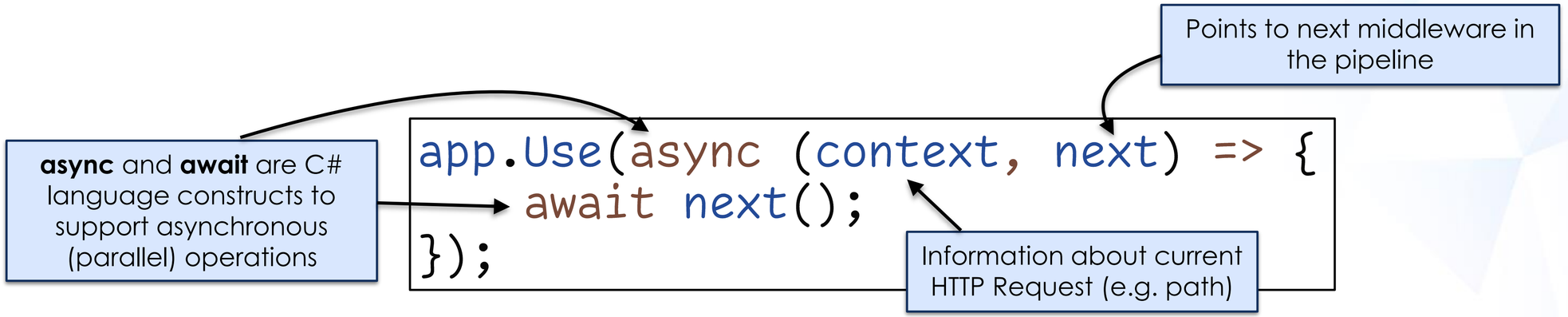
Middleware	Purpose	Name
HTTPS Redirection	Redirect a web browser to use HTTPS requests instead of HTTP requests	app.UseHttpsRedirection()
Files Retrieval	Fetch static files such as JavaScript code (.js), style-sheets (.css) and images (.png, .jpg etc)	app.UseStaticFiles()
Routing	Route requests to the appropriate controllers and action methods	app.UseRouting()

Middleware

- We shall explore two ways of writing a Middleware in .NET
  - Inline Middleware
  - Middleware Class
- No real differences between them - though a Middleware Class is deemed more reusable
- An Inline Middleware can be easily converted to a Middleware Class

Inline Middleware

C# language construct of a barebone Inline Middleware



Using Inline Middleware

Use app.Use(...) to insert an Inline Middleware into a .NET application

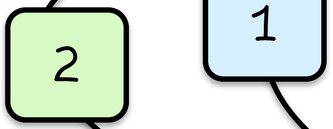
```
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();

// our middleware 1
app.Use(async (context, next) => {
    await next();
});

// out middleware 2
app.Use(async (context, next) => {
    await next();
});

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

.NET's pre-defined Middleware

Inserting our custom Middleware into the application's pipeline

Execution Flow

Note the execution flow of these two Middleware




```
// our middleware 1
app.Use(async (context, next) =>
{
    Debug.WriteLine("Middleware 1: Entry");
    await next();
    Debug.WriteLine("Middleware 1: Exit");
});

// out middleware 2
app.Use(async (context, next) =>
{
    Debug.WriteLine("Middleware 2: Entry");
    await next();
    Debug.WriteLine("Middleware 2: Exit");
});
```

Output

```
Middleware 1: Entry
Middleware 2: Entry
Middleware 2: Exit
Middleware 1: Exit
```

MIDDLEWARE EXAMPLES

Estimate Latency

This Middleware approximates the time taken for a Request and Response cycle to complete

```
app.Use(async (context, next) =>
{
    // capture start time
    long startTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();

    await next(context);

    // capture end time
    long endTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();

    // compute difference
    int duration = (int) (endTime - startTime);
});
```

Enforce Valid Session

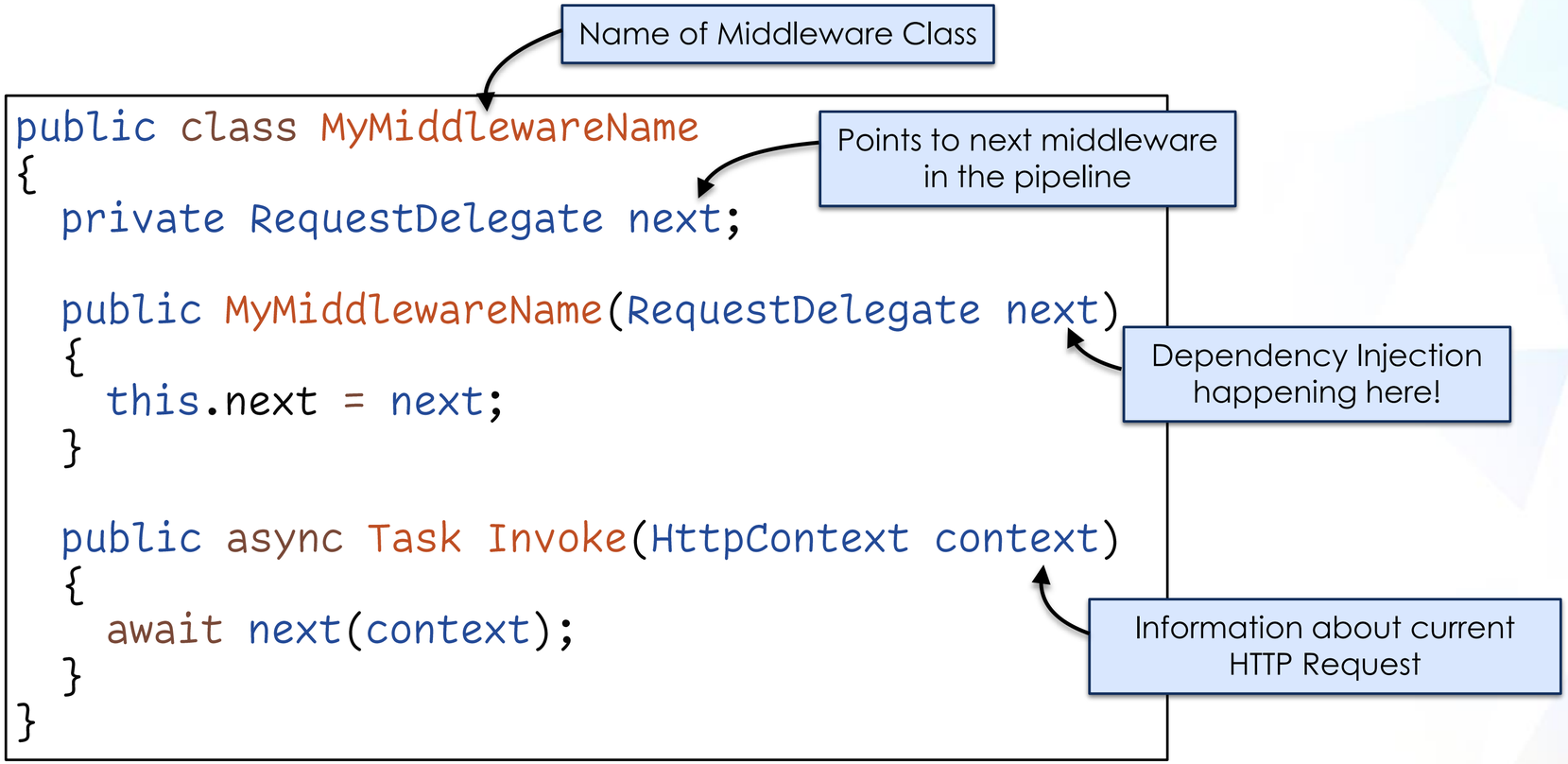
This Middleware directs any web requests without a valid Session-ID cookie to the Login page

```
app.Use(async (context, next) => {
    if (context.Request.Path.StartsWithSegments("/Login/")) {
        // do nothing if user is heading to Login page
        await next(context);
        return;
    }

    string sessionId = context.Request.Cookies["sessionId"];
    if (sessionId == null) {
        // bring user to Login page to get a session
        context.Response.Redirect("/Login/");
    }
    else {
        await next(context);
    }
});
```

MIDDLEWARE CLASSES

Middleware Class

A Middleware Class has the following language construct




```
public class MyMiddlewareName
{
  private RequestDelegate next;

  public MyMiddlewareName(RequestDelegate next)
  {
    this.next = next;
  }

  public async Task Invoke(HttpContext context)
  {
    await next(context);
  }
}
```

Estimate Latency

Implementing the Estimate Latency middleware as a Middleware Class

```
public class CycleTimer
{
    private RequestDelegate next;

    public CycleTimer(RequestDelegate next) {
        this.next = next;
    }

    public async Task Invoke(HttpContext context) {
        // capture start time
        long startTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();

        await next(context);

        // capture end time
        long endTime = DateTimeOffset.Now.ToUnixTimeMilliseconds();

        // compute difference
        int duration = (int) (endTime - startTime);
    }
}
```

Get current Unix timestamp before control is passed to the middleware after us

Get current Unix timestamp before control is passed to the middleware before us

Enforce Valid Session

Implementing the Enforce Valid Session middleware as a Middleware Class

```
public class LoginChecker
{
    private RequestDelegate next;

    public LoginChecker(RequestDelegate next) {
        this.next = next;
    }

    public async Task Invoke(HttpContext context) {
        if (context.Request.Path.StartsWithSegments("/Login/")) {
            await next(context);
            return;
        }

        string sessionId = context.Request.Cookies["sessionId"];
        if (sessionId == null) {
            // bring user to Login page to get a session
            context.Response.Redirect("/Login/");
        }
        else {
            await next(context);
        }
    }
}
```

If user is trying to visit the Login page, then need not perform the check

Using Middleware Class

Adding our Middleware Classes to the Middleware Pipeline

```
…

app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseAuthorization();

// install our custom middlewares
app.UseMiddleware<CycleTimer>();
app.UseMiddleware<LoginChecker>();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

Inserting our middleware classes into our app's Middleware pipeline

THE END