SESSION MANAGEMENT IN .NET MVC

DATA AND SECURITY BUNDLE

WHAT WE'RE COVERING

Stateless Programming

Sessions -What are they and how they are used?

Using Sessions in ASP.NET Core

TempData

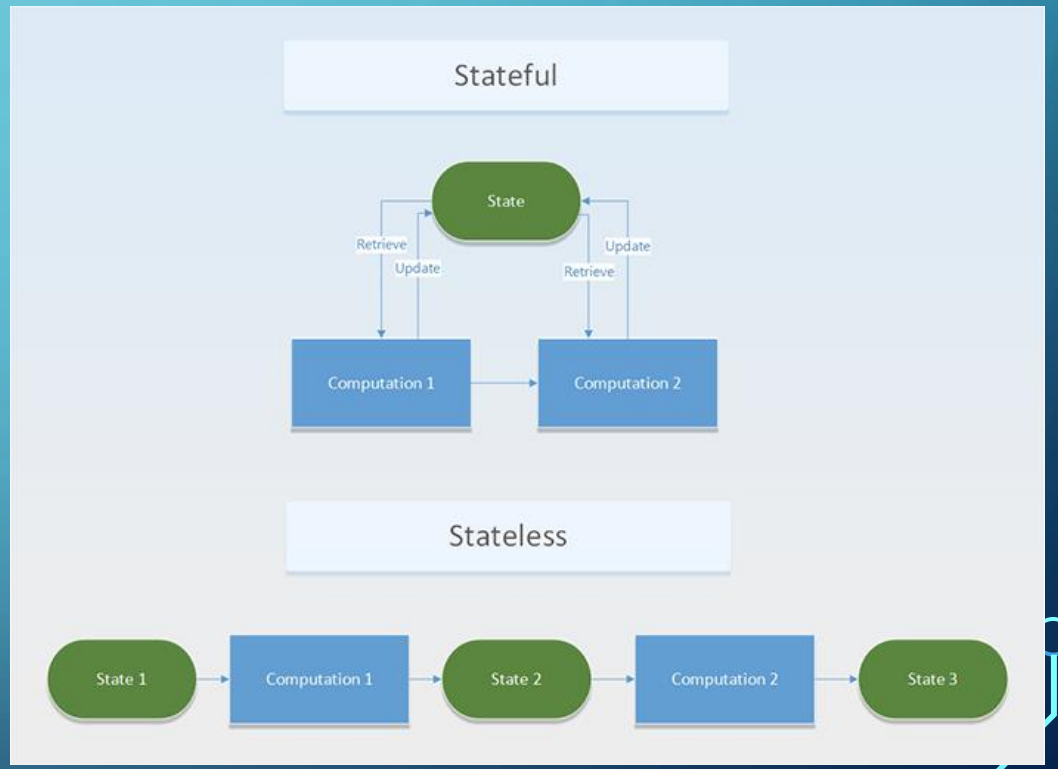
Cookies

Session Management

STATELESS PROGRAMMING

Unlike desktop and console applications, most websites and web applications are designed to treat each request and response from the user as a new interaction and to not automatically retain data between page refreshes.

This is what we refer to as a stateless system, as it does not save the state of the system based upon your recent actions when processing new requests.

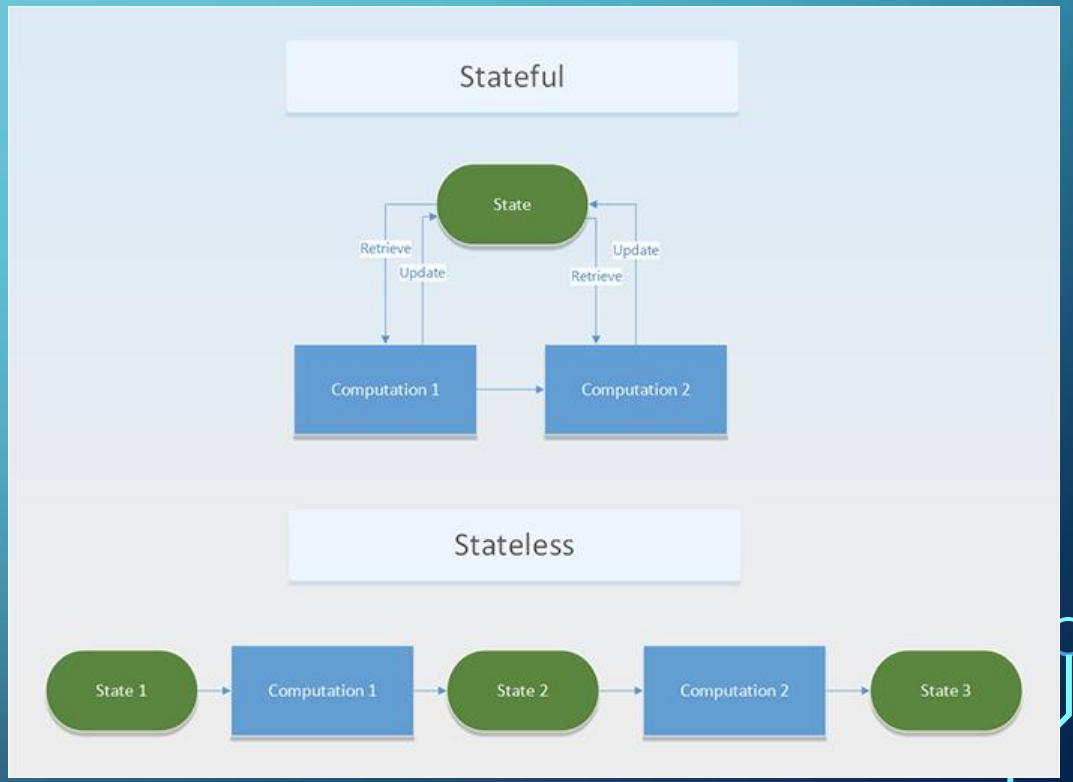


STATELESS PROGRAMMING

Each request and response generates a brand-new HTML page for each new interaction and sends it to your browser on each request. Even if you are staying on the same page.

With each interaction, any values and variables created while processing the request are discarded once it is finished.

This is different to other types of systems which will automatically store variables and values that will then be reused for each interaction.



STATELESS PROGRAMMING

This means that anything you have done on screen, prior to the latest request to the website servers, will be lost once the request is processed.

Because of this, we need alternative systems to retain this data between interactions so that if there is any data that needs to be reused, it can be retrieved and reapplied to the page once it reloads.

WEB SESSIONS

One way to do this is by using web sessions and session data, which can be saved and passed between the client and server to keep track of this data.

Once it is needed, it can be retrieved and put back where it is required to give a feel of working within a stateful system.

WHAT ARE SESSIONS?

The term session refers to the sequence of requests and responses between a server and the same user

Session data is used work around the statelessness of the HTTP Protocol

Session data is a set of data saved on the server that is associated to a client's user session in their device.

They can be configured to be long or short lived and can even be set to be disposed of once you close your browser.

WHERE ARE THEY STORED?

As the web-server operates and handles interactions with its users, it may have systems or code in place to store certain values as session data in memory on the server-machine.

The client machine will then get a copy of a session ID which acts as a reference to this data. This will identify which session data is associated with them.

Each time a user initiates a session with the server they will be assigned a new session ID which will be used until it expires, or a new ID is assigned.

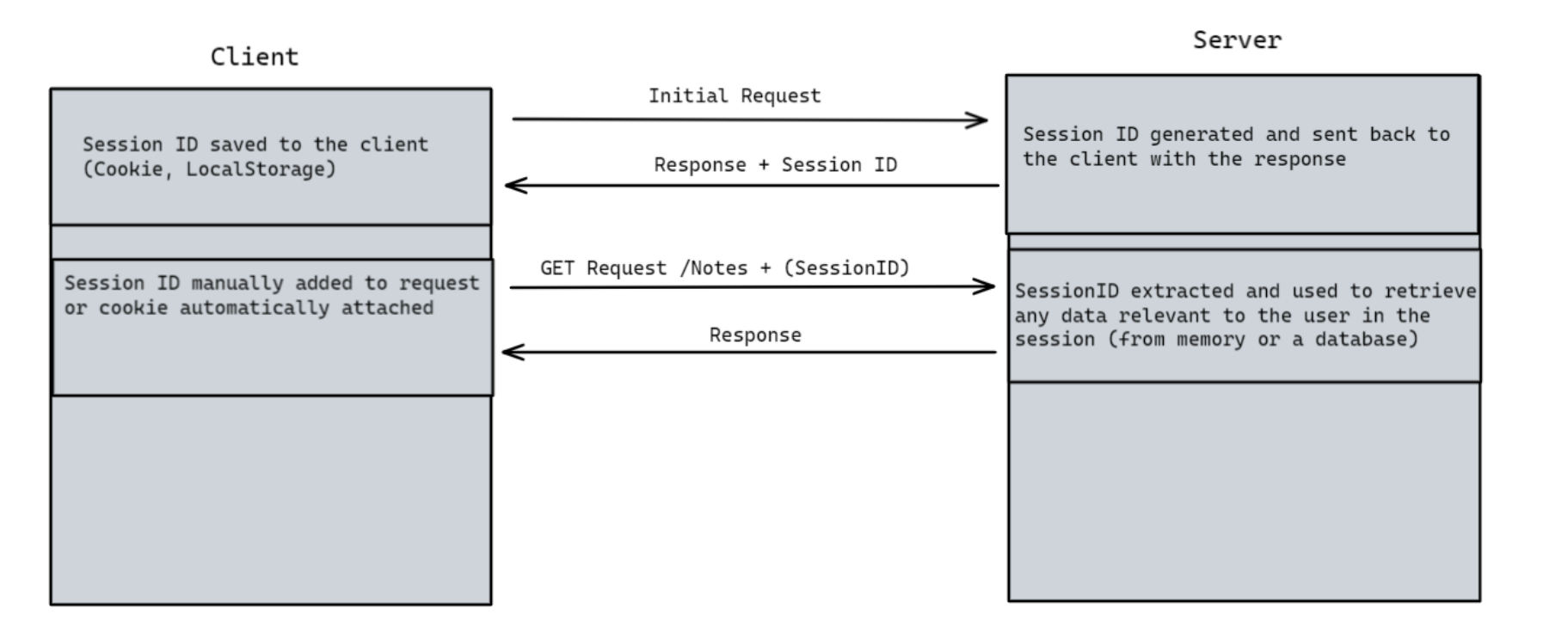
HOW THEY WORK?

For every request it processes, the server's default behavior is to treat each request as if it has not interacted with the current user before.

However, if a request comes in with a session ID provided in the request details, it will look at the stored session data to see if there is any relevant data for that user that has been previously stored in the data.

If there is, it will retrieve the data and use it for the request.

SESSION EXAMPLE



COOKIES

Another alternative for storing user data is by using cookies.

Cookies were first used in about 1995 to store data related to web usage on devices. They were the first system used in websites to help achieve data persistence in website interactions.

Cookies can be set to be Persistent or Session based, depending on your intended usage.

COOKIES

Persistent cookies normally have a set expiry based upon a timeframe. This can be anything from minutes to years.

These are normally stored on the hard disk in the client device as a file.

On every request to the site that the cookie is created for, the entire contents of the cookie is attached to each request so that its data can be applied to the web pages of the site.

COOKIES

Session based cookies can have a short term, time-based expiry but will normally also be set to automatically expire with the closing of a web browser.

These are stored on the client device in memory only and will be automatically lost once the session closes by closing the browser.

The data in these will also be added to each request to the web server.

SESSION STORAGE & LOCAL STORAGE

Session and Local storage are newer technologies than Cookies which allow for the storage of larger amounts of data.

Unlike cookies however, they are not sent to the server with each request. Instead, they are used only on the client side and usually rely on being implemented through JavaScript in your HTML

- Local Storage is stored in your hard disk until deleted
- Session Storage is cleared at the end of a page session
    - A page session is limited to the tab that opened the page, and is not accessible across tabs/windows

CONFIGURING SESSIONS

- The session and cookies are configured in Program.cs
- The Configuration needs to be added to ‘Services’ Section.
- Usage of the Session is added to the ‘Configure’ Section

```
builder.Services.AddDistributedMemoryCache();
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromSeconds(60);
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});
```

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

app.UseRouting();

app.UseAuthorization();

app.UseSession();
```

USING SESSIONS

- Session values can be set using the following syntax (requires using Microsoft.AspNetCore.Http;)

```
HttpContext.Session.SetString("name", "example");
```

- Session values can be accessed in several ways, one is demonstrated below:

```
// returns the value or null
string name = HttpContext.Session.GetString("name");

if(!String.IsNullOrEmpty(name))
{
    // use the data retrieved from the session
}
```

USING SESSIONS IN A VIEW

- Sessions can be manipulated from within a View
- Adding the AspNetCore.Http using statement allows access to .GetString()

```
@using Microsoft.AspNetCore.Http;

@{
    ViewData["Title"] = "Home Page";
    string name = Context.Session.GetString("name");
}

<div class="text-center">
    <h1 class="display-4">Welcome @Context.Session.GetString("name")</h1>
</div>
```