

# Web Storage and Cookies

You can save data on the client  
– even between sessions!

# Why?

- HTTP is a stateless protocol
- A web server treats each HTTP request as an independent request and does not retain user values from previous requests
- But you may use cookies or local storage on the client to preserve application and session state between requests

# What Are Cookies?

- Cookies are pieces of data, normally stored in text files, that websites place on visitors' computers to store a range of information, usually specific to that visitor
- Cookies allow you to login on one page, then move around to other pages and stay logged in
- They allow you to set preferences for the display of a page, and for these to be remembered the next time you return to it
- Cookies can also be used to watch the pages you visit between sites, which allows advertisers to build up a picture of your interests
  - This is known as 'behavioural advertising'
- Cookies are incredibly useful
- But they can also be used to manipulate your web experience in ways you might not expect, or like

# EU legislation on cookies

- The ePrivacy directive – more specifically Article 5(3) – requires prior informed consent for storage or for access to information stored on a user's terminal equipment
- In other words, you must ask users if they agree to most cookies and similar technologies
- Consent is not required if the cookie is:
  - **user-input** cookies (session-id) such as first-party cookies to keep track of the user's input when filling online forms, shopping carts, etc., for the duration of a session or persistent cookies limited to a few hours in some cases
  - **authentication** cookies, to identify the user once he has logged in, for the duration of a session
  - **user-centric security** cookies, used to detect authentication abuses, for a limited persistent duration
  - **multimedia content player** cookies, used to store technical data to play back video or audio content, for the duration of a session
  - **load-balancing** cookies, for the duration of session
  - **user-interface customisation** cookies such as language or font preferences, for the duration of a session (or slightly longer)
  - **third-party social plug-in content-sharing** cookies, for logged-in members of a social network.

# The different types of cookies

- By lifespan, a cookie is either a:
  - **session cookie** which is erased when the user closes the browser *or*
  - **persistent cookie** which remains on the user's computer/device for a pre-defined period of time
- By domain to which it belongs, there are either:
  - **first-party cookies** which are set by the web server of the visited page and share the same domain
  - **third-party cookies** stored by a different domain to the visited page's domain

# Cookies limitation

- Because cookies are sent with every request, their size should be kept to a minimum
- Ideally, only an identifier should be stored in a cookie with the actual data stored on the server
- Most browsers restrict cookies to 4096 bytes
- Only a limited number of cookies are available for each domain
- Users can clear the cookies on their computer at any time
  - Even if you store cookies with long expiration times, a user might decide to delete all cookies, wiping out any settings you might have stored in cookies

# Cookies in ASP.Net

- Both the HttpRequest and the HttpResponse objects expose a collection called Cookies
- You can access the HttpRequest and HttpResponse object as the Request/Response property of your Controller class
- Any cookies that you want to send to the browser must be added to this collection
- When creating a cookie, you specify a Name and Value
- Each cookie must have a unique name so that it can be identified later when reading it from the browser

# Writing Cookies

- You can add cookies to the Cookies collection in a number of ways


```
Response.Cookies["userName"].Value = "patrick";  
Response.Cookies["userName"].Expires = DateTime.Now.AddDays(1);  
  
HttpCookie aCookie = new HttpCookie("lastVisit");  
aCookie.Value = DateTime.Now.ToString();  
aCookie.Expires = DateTime.Now.AddDays(1);  
Response.Cookies.Add(aCookie);
```



# Reading Cookies

- To read a cookie in an action method in a controller

To get a single value



```
if(Request.Cookies["userName"] != null)
{
    string cookieValueFromReq = Request.Cookies["userName"].Value

    HttpCookie aCookie = Request.Cookies["userName"];
}
```

# Cookies with more than one value

- To create a cookie with subkeys, you can use a variation of the syntax for writing a single cookie

```
Response.Cookies["userInfo"]["userName"] = "patrick";  
Response.Cookies["userInfo"]["lastVisit"] = DateTime.Now.ToString();  
Response.Cookies["userInfo"].Expires = DateTime.Now.AddDays(1);  
  
HttpCookie aCookie = new HttpCookie("userInfo");  
aCookie.Values["userName"] = "patrick";  
aCookie.Values["lastVisit"] = DateTime.Now.ToString();  
aCookie.Expires = DateTime.Now.AddDays(1);  
Response.Cookies.Add(aCookie);
```

# Cookie Options

- *Domain*
  - The domain you want to associate with cookie
- *Path*
  - Cookie Path
- *Expires*
  - The expiration date and time of the cookie
- *HttpOnly*
  - Gets or sets a value that indicates whether a cookie is accessible by client-side script or not
- *Secure*
  - Transmit the cookie using Secure Sockets Layer (SSL) that is, over HTTPS only

# Cookies from JavaScript

- JavaScript can create, read, and delete cookies with the `document.cookie` property

```
document.cookie = "username=John Doe";
```

- The cookie is deleted when the browser is closed unless you add an expiry date in UTC time

```
document.cookie = "username=John Doe; expires=Thu, 18 Dec 2018 12:00:00 UTC";
```

- With a path parameter, you can tell the browser what path the cookie belongs to. By default, the cookie belongs to the current page

```
document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";
```

# A Function to Get a Cookie

```
function getCookie(cname) {  
    var name = cname + "=";  
    var decodedCookie = decodeURIComponent(document.cookie);  
    var ca = decodedCookie.split(';');  
    for(var i = 0; i < ca.length; i++) {  
        var c = ca[i];  
        while (c.charAt(0) == ' ') {  
            c = c.substring(1);  
        }  
        if (c.indexOf(name) == 0) {  
            return c.substring(name.length, c.length);  
        }  
    }  
    return "";  
}
```













# Session state

- Session state is a feature in ASP.NET Core that you can use to save and store user data while the user browses your web app
- Consisting of a dictionary or hash table on the server, session state persists data across requests from a browser
- ASP.NET Core maintains session state by giving the client a cookie that contains the session ID, which is sent to the server with each request
- The server uses the session ID to fetch the session data

# Web Storage

# HTML5 Client-Side Storage

## Browser Support

									
<i>Web Storage - name/value pairs</i> 	4+	3.5+	4+	10.5+	8+	3.2+	2.1+	—	11.5+
<i>IndexedDB</i> 	23+	10+	9+	15+	10+	9+	4.4	—	0
<i>Web SQL Database</i> 	4+	—	3.1+	10.5+	—	3.2+	2.1+	—	11.5+

- Web Storage simply provides a key-value mapping,
  - e.g. `localStorage["name"] = username;`
- Web SQL Database gives you all the power of a structured SQL relational database
- Indexed Database is somewhere in between Web Storage and Web SQL Database
  - Like Web Storage, it's a straightforward key-value mapping, but it supports indexes, so searching objects matching a particular field is fast



# Web Storage

- Supports persistent data storage
  - similar to cookies but with a greatly enhanced capacity (5-10MB per origin) and no information stored in the HTTP request header
- There are two main web storage types:
  - localStorage
    - Permanent storage (may be deleted by user)
    - Data placed in local storage is per origin (protocol+hostname+port number)
  - sessionStorage
    - Limited to the lifetime of the window
- Browsers that support web storage have the global variables 'sessionStorage' and 'localStorage' declared at the window level
- Web storage is being standardized by the World Wide Web Consortium

# How to Use?

- Web Storage only supports string-to-string mappings
  - so you need to serialise and de-serialise other data structures
- You can use `JSON.stringify()` and `JSON.parse()` to store other data types than string

```
// Store an object/array using JSON
localStorage.numbers = JSON.stringify(numbers);
```

```
if (localStorage.hasOwnProperty("numbers")) {
    // Read an object/array using JSON
    numbers = JSON.parse(localStorage.numbers);
}
```

# References & Links

- Introduction to session and application state in ASP.NET Core  
<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/app-state?tabs=aspnetcore2x>
- Cokiepedia  
<https://cookiepedia.co.uk/all-about-cookies>
- Lovgivning og vejledning til cookiebekendtgørelsen  
<https://erhvervsstyrelsen.dk/lovgivning-og-vejledning-til-cookiebekendtgørelsen>