Im Kontext einer Webanwendung ist Datenspeicherung unerlässlich.

Sie können ja schon Daten in einer Datenbank speichern und von dort abfragen. Datenbanken sind allerdings für große und dauerhafte Datenbestände ausgelegt.

Zur Speicherung von kleineren, temporären Informationen gibt es

# **Cookies**

## **Cookies**

werden **von Browser verwaltet.** Wenn man also z.B. dieselbe Seite in zwei verschiedenen Browsern öffnet, setzt jeder Browser das entsprechende Cookie für sich selbst.

**Zweck**: Personalisieren, Marketing

**Format**: Name=Wert – z.B. **Username=Anna** (Key-Value)

dazu kann man noch optionale Parameter setzen:

Expires – Ablaufsdatum des Cookies – expires=Tue, 29-Mar-2014 19:30:42 GMT;

Domain – für welche Domäne ist das Cookie gültig

Path – für welchen Pfad gilt das Cookie

HttpOnly – verhindert den Zugriff auf das Cookie von JavaScript aus

Max-Age – maximales Alter des Cookies in Sekunden – z.B. 2592000 (= 30 Tage)

**Einschränkungen**:

Bis 4096 Bytes groß.

Pro Domain bis 50 Cookies.

Bis 3000 Cookies insgesamt möglich

**Speicherort:** Client (Browser-Cache)

**Cookie Umsetzung clientseitig (JavaScript)**





**Http-Request: Client fordert von Server eine Seite, setzt auf dieser das Cookie und sendet die Seite mit den Cookies im Request-Header per POST zum Server zurück**

GET http://localhost:59037

Host: localhost:59037

Connection: Upgrade

Pragma: no-cache

Cache-Control: no-cache

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36

Upgrade: websocket

Origin: http://localhost:5011

Sec-WebSocket-Version: 13

Accept-Encoding: gzip, deflate, br, zstd

Accept-Language: de-DE,de;q=0.9,pl;q=0.8,en;q=0.7

Cookie: bgColor=yellow; BesuchsZaehler=3

dnt: 1

Sec-WebSocket-Key: VNu68/pFCJv1BAFmnEZ0eg==

Sec-WebSocket-Extensions: permessage-deflate; client\_max\_window\_bits

Cookie setzen

function setBgColorInCookie(combo, days) {

var bgColor = combo.value;

document.cookie = "bgColor=" + bgColor;

}

Cookie auslesen

function getCookieValue(name) {

var cookies = document.cookie.split('; ');

for (var cookie of cookies) {

const [key, value] = cookie.split('=');

if (key === name) {

return value;

}

}

return null;

}

**Cookie Umsetzung serverseitig (ASP.NET Core, C#)**



**Http-Response: Server fordert den Client an, ein Cookie zu erstellen**

****

HTTP/1.1 200 OK

Content-Type: text/html; charset=utf-8

Date: Sun, 17 Nov 2024 08:03:44 GMT

Server: Kestrel

Set-Cookie: BesuchsZaehler=1; path=/

Transfer-Encoding: chunked

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title> - CookieSessionApplicationTest</title>

....

</head>

<body>

......

</body>

</html>

Serverseitiges Auslesen

int.TryParse(HttpContext.Request.Cookies["BesuchsZaehler"], out

int besuchsZaehler);

besuchsZaehler++;

Serverseitige Aufforderung zum Setzen

HttpContext.Response.Cookies.Append("BesuchsZaehler", besuchsZaehler.ToString());

Auslesen in View aus dem Header – Anweisung „Set-Cookie“

Context.Response.Headers?.SetCookie[0]?.Split(';')[0].Split('=')[1]

Serverseitiges Auslesen in View aus dem Cookie (Cookie ist zu dem Zeitpunkt noch nicht aktualisiert)

Context.Request.Cookies.FirstOrDefault(x => x.Key == "BesuchsZaehler").Value

**Technische Infos zu Cookie**

<https://learn.microsoft.com/de-de/aspnet/web-api/overview/advanced/http-cookies>

**Aufgabe**

Erstellen Sie auf einer Webseite ein Inputfeld und speichern Sie den eingegebenen Inhalt in einem Cookie.  
Wenn der Benutzer die Seite neu lädt, soll der Wert aus dem ausgelesenen Cookie neben dem Inputfeld angezeigt werden.  
Wie Sie das Cookie speichern und auslesen (client- oder serverseitig), bleibt es Ihnen überlassen.