**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

# МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

***Факультет Информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки 09.03.02 «Информационные системы и технологии»**

# ЛАБОРАТОРНАЯ РАБОТА № \_8\_\_

Дисциплина: BackEnd-разработка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема: *Изучение состояний в веб-приложение на основе ASP.NET*

*Core*\_\_\_\_\_\_\_\_\_\_

**Выполнил(а): студент(ка) группы \_\_221-3711\_\_\_\_\_\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_\_\_Костоваров А. С.

(Фамилия И.О.)

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

## Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_*

(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

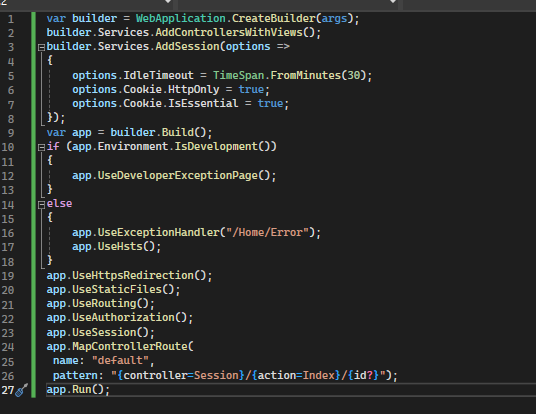
## Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Москва**

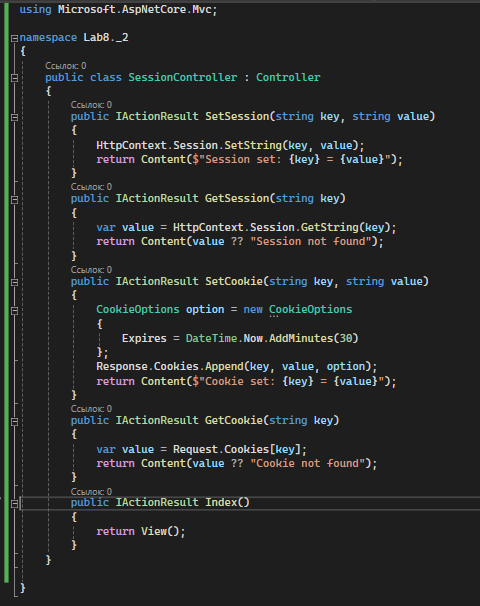
**2024**

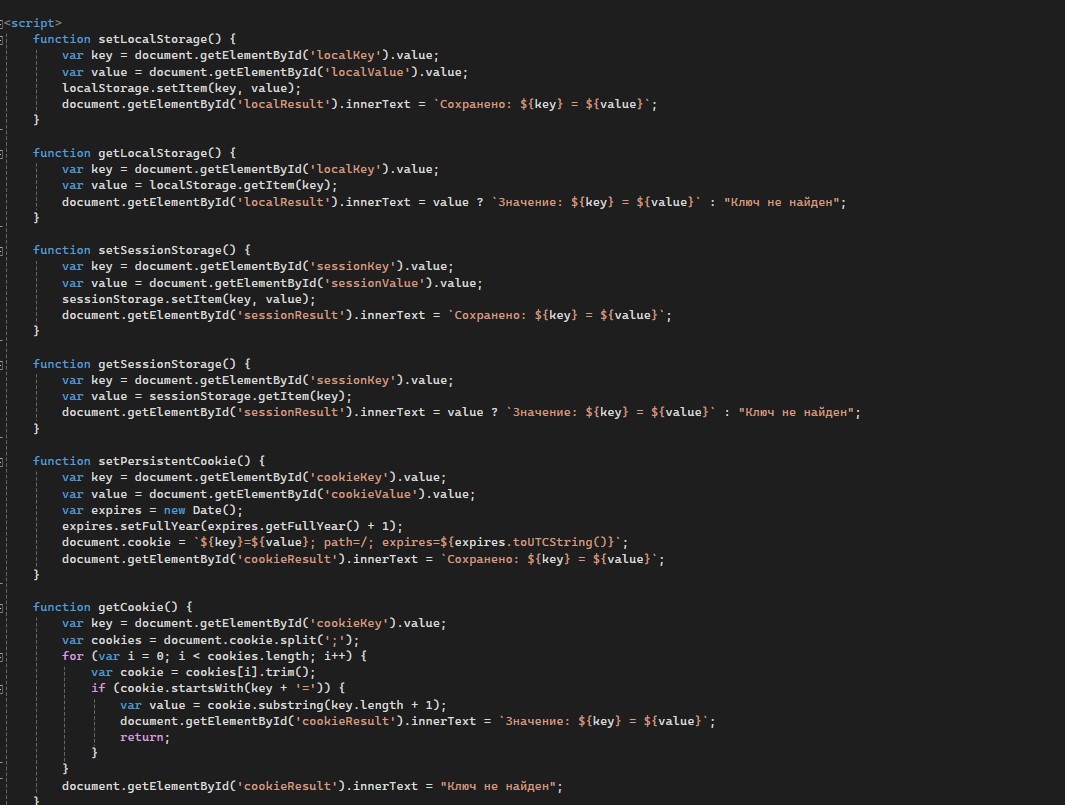
В этой лабораторной работе мы рассмотрим различные способы управления состояниями в веб-приложениях на платформе ASP.NET Core. Управление состояниями позволяет сохранять информацию о пользователе и его действиях как на клиентской, так и на серверной сторонах. Мы создадим простое приложение, которое позволит задавать и извлекать данные из сессий, локального хранилища и cookies.

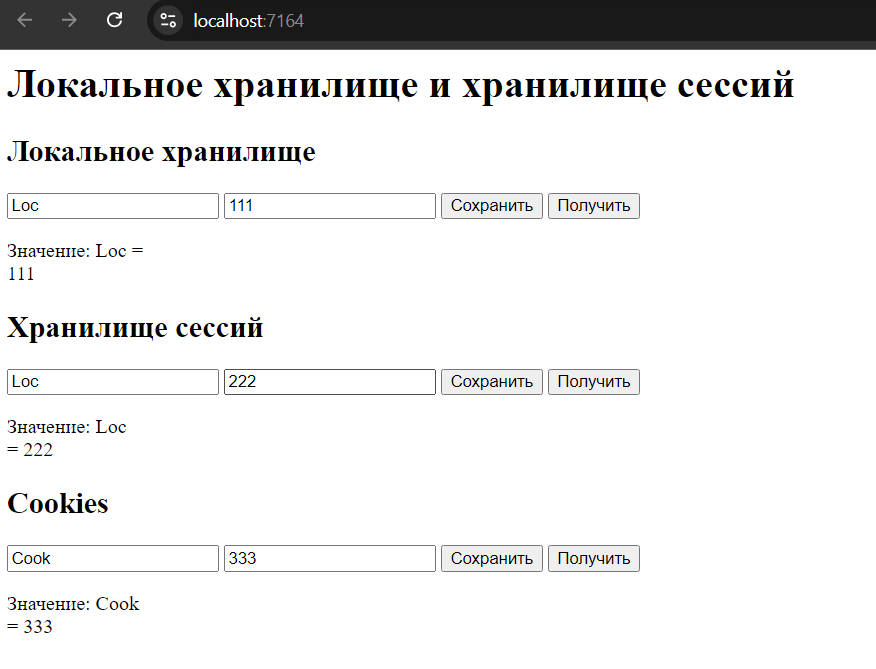
В файле Program.cs мы активируем использование сессий и зададим время жизни сессии в 30 минут. Это означает, что данные сессии будут сохраняться на сервере в течение 30 минут после последнего запроса пользователя. Если в течение этого времени пользователь не выполнит никаких действий, данные будут автоматически удалены.



Далее создадим контроллер для управления сессиями у куки файлами, с возможностью задавать их и получать.



Для сессий данные хранятся на сервере, в то время как идентификатор сессии сохраняется в cookie на стороне клиента. Сессия включает в себя пару значений: ключ сессии и соответствующее значение, которое она хранит. Также существует метод для установки cookies, который позволяет сохранять небольшие фрагменты данных на стороне клиента. Эти cookies отправляются на сервер с каждым запросом и также имеют ключ и значение. По ключу можно получить данные как из сессий, так и из cookies.



Код приложения:

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

builder.Services.AddSession(options =>

{

options.IdleTimeout = TimeSpan.FromMinutes(30);

options.Cookie.HttpOnly = true;

options.Cookie.IsEssential = true;

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseSession();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Session}/{action=Index}/{id?}");

app.Run();

using Microsoft.AspNetCore.Mvc;

namespace Lab8.\_2

{

public class SessionController : Controller

{

public IActionResult SetSession(string key, string value)

{

HttpContext.Session.SetString(key, value);

return Content($"Session set: {key} = {value}");

}

public IActionResult GetSession(string key)

{

var value = HttpContext.Session.GetString(key);

return Content(value ?? "Session not found");

}

public IActionResult SetCookie(string key, string value)

{

CookieOptions option = new CookieOptions

{

Expires = DateTime.Now.AddMinutes(30)

};

Response.Cookies.Append(key, value, option);

return Content($"Cookie set: {key} = {value}");

}

public IActionResult GetCookie(string key)

{

var value = Request.Cookies[key];

return Content(value ?? "Cookie not found");

}

public IActionResult Index()

{

return View();

}

}

<html>

<head>

<h1>Локальное хранилище и хранилище сессий</h1>

<div>

<h2>Локальное хранилище</h2>

<input type="text" id="localKey" placeholder="Ключ" />

<input type="text" id="localValue" placeholder="Значение" />

<button onclick="setLocalStorage()">Сохранить</button>

<button onclick="getLocalStorage()">Получить</button>

<p id="localResult"></p>

</div>

<div>

<h2>Хранилище сессий</h2>

<input type="text" id="sessionKey" placeholder="Ключ" />

<input type="text" id="sessionValue" placeholder="Значение" />

<button onclick="setSessionStorage()">Сохранить</button>

<button onclick="getSessionStorage()">Получить</button>

<p id="sessionResult"></p>

</div>

<div>

<h2>Cookies</h2>

<input type="text" id="cookieKey" placeholder="Ключ" />

<input type="text" id="cookieValue" placeholder="Значение" />

<button onclick="setPersistentCookie()">Сохранить</button>

<button onclick="getCookie()">Получить</button>

<p id="cookieResult"></p>

</div>

</head>

<body>

<script>

function setLocalStorage() {

var key = document.getElementById('localKey').value;

var value = document.getElementById('localValue').value;

localStorage.setItem(key, value);

document.getElementById('localResult').innerText = `Сохранено: ${key} =

${value}`;

}

function getLocalStorage() {

var key = document.getElementById('localKey').value;

var value = localStorage.getItem(key);

document.getElementById('localResult').innerText = value ? `Значение: ${key} =

${value}` : "Ключ не найден";

}

function setSessionStorage() {

var key = document.getElementById('sessionKey').value;

var value = document.getElementById('sessionValue').value;

sessionStorage.setItem(key, value);

document.getElementById('sessionResult').innerText = `Сохранено: ${key} =

${value}`;

}

function getSessionStorage() {

var key = document.getElementById('sessionKey').value;

var value = sessionStorage.getItem(key);

document.getElementById('sessionResult').innerText = value ? `Значение: ${key}

= ${value}` : "Ключ не найден";

}

function setPersistentCookie() {

var key = document.getElementById('cookieKey').value;

var value = document.getElementById('cookieValue').value;

var expires = new Date();

expires.setFullYear(expires.getFullYear() + 1);

document.cookie = `${key}=${value}; path=/; expires=${expires.toUTCString()}`;

document.getElementById('cookieResult').innerText = `Сохранено: ${key} =

${value}`;

}

function getCookie() {

var key = document.getElementById('cookieKey').value;

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

for (var i = 0; i < cookies.length; i++) {

var cookie = cookies[i].trim();

if (cookie.startsWith(key + '=')) {

var value = cookie.substring(key.length + 1);

document.getElementById('cookieResult').innerText = `Значение: ${key}

= ${value}`;

return;

}

}

document.getElementById('cookieResult').innerText = "Ключ не найден";

}

</script>

</body>