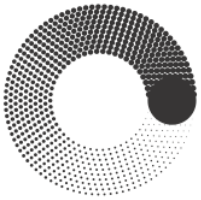
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



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_____

_____Марков М.Д._____
(Фамилия И.О.)

Дата, подпись _____ _____
(Дата) (Подпись)

Проверил: _____ _____
(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись _____ _____
(Дата) (Подпись)

Замечания: _____

_____

Москва

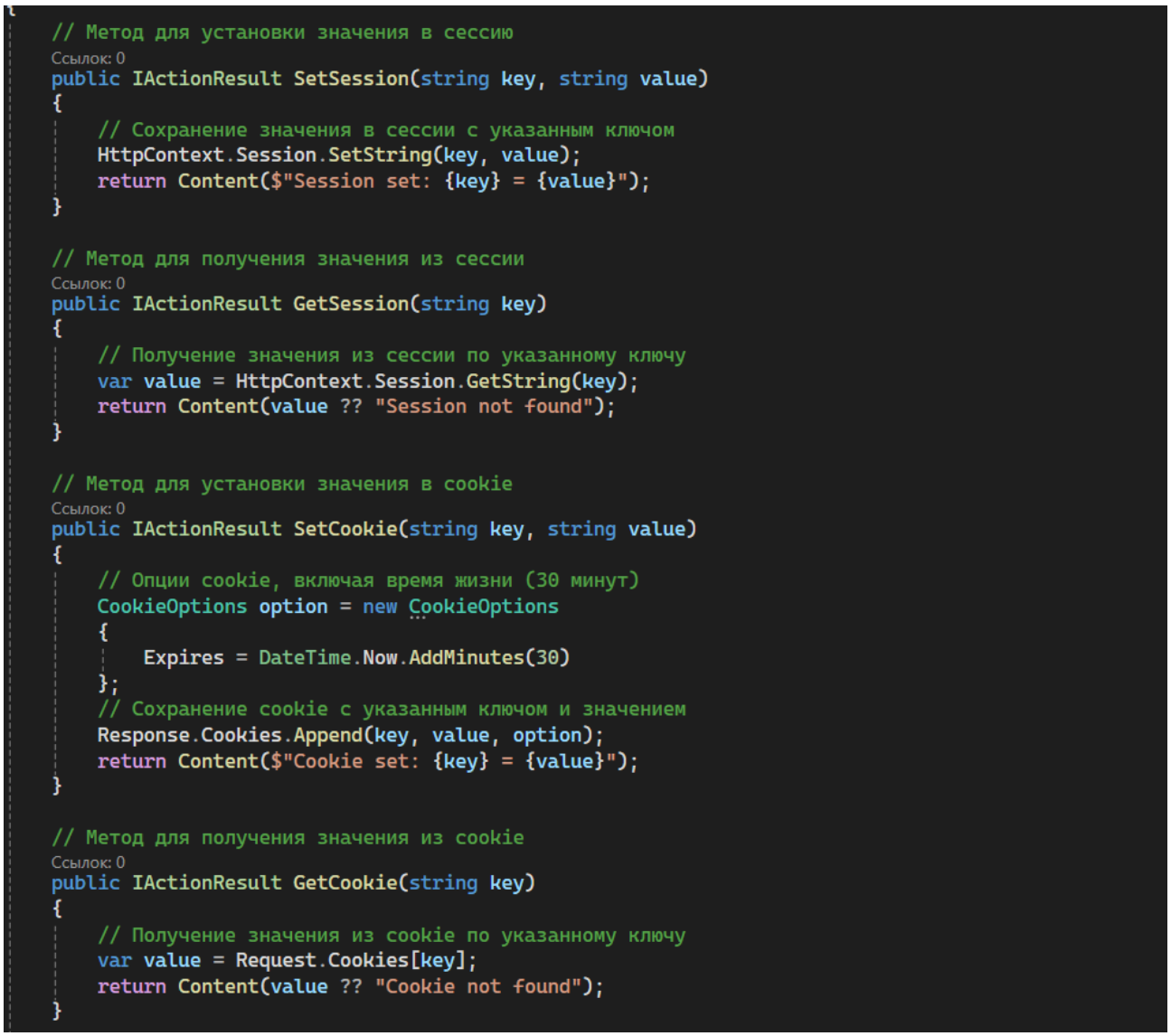
2024

В данной лабораторной работе мы ознакомимся различные способы управления состояниями в веб-приложениях на платформе ASP.NET Core. Состояния позволяют сохранять информацию о пользователе и его действиях как на стороне клиента, так и на стороне сервера. Создадим простое приложение, позволяющее задавать данные сессии, локального хранилища и куки, а также получать из них данные.

В файле program включим использование сессий и устанавливаем время жизни 30 минут. Это означает что данные сессии будут храниться на стороне сервера на протяжении 30 минут с последнее запроса пользователя, если в течении этого времени пользователь не совершит никаких действий, данные будут автоматически очищены.

```
6
7   var builder = WebApplication.CreateBuilder(args);
8
9   // Добавление служб в контейнер
10  builder.Services.AddControllersWithViews();
11
12  // Добавление службы сессий
13  builder.Services.AddSession(options =>
14  {
15      options.IdleTimeout = TimeSpan.FromMinutes(30); // Время жизни сессии
16      options.Cookie.HttpOnly = true;
17      options.Cookie.IsEssential = true;
18  });
19
20  var app = builder.Build();
21
22  // Настройка конвейера обработки HTTP-запросов
23  if (app.Environment.IsDevelopment())
24  {
25      app.UseDeveloperExceptionPage();
26  }
27  else
28  {
29      app.UseExceptionHandler("/Home/Error");
30      app.UseHsts();
31  }
32
33  app.UseHttpsRedirection();
34  app.UseStaticFiles();
35
36  app.UseRouting();
37
38  app.UseAuthorization();
39
40  // Использование сессий
41  app.UseSession();
42
43  app.MapControllerRoute(
44      name: "default",
45      pattern: "{controller=Session}/{action=Index}/{id?}");
46
47  app.Run();
48
```

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

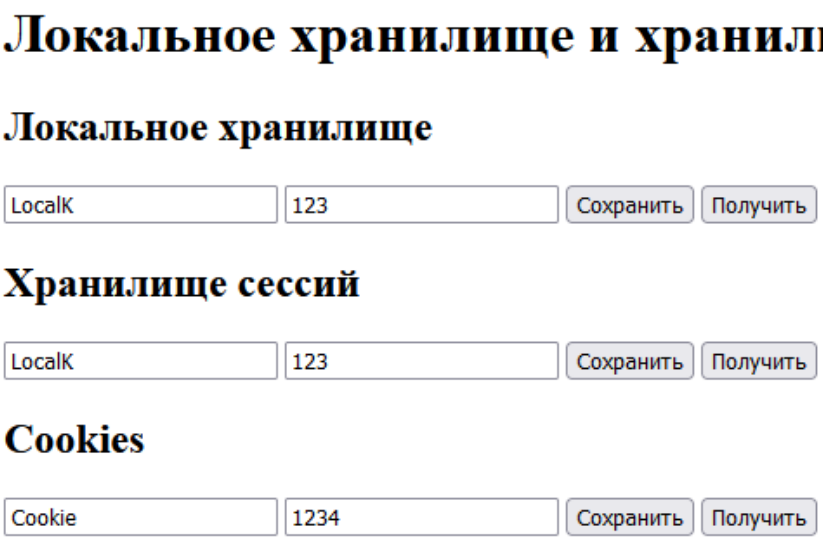


```
// Метод для установки значения в сессию
Ссылок: 0
public IActionResult SetSession(string key, string value)
{
    // Сохранение значения в сессии с указанным ключом
    HttpContext.Session.SetString(key, value);
    return Content($"Session set: {key} = {value}");
}

// Метод для получения значения из сессии
Ссылок: 0
public IActionResult GetSession(string key)
{
    // Получение значения из сессии по указанному ключу
    var value = HttpContext.Session.GetString(key);
    return Content(value ?? "Session not found");
}

// Метод для установки значения в cookie
Ссылок: 0
public IActionResult SetCookie(string key, string value)
{
    // Опции cookie, включая время жизни (30 минут)
    CookieOptions option = new CookieOptions
    {
        Expires = DateTime.Now.AddMinutes(30)
    };
    // Сохранение cookie с указанным ключом и значением
    Response.Cookies.Append(key, value, option);
    return Content($"Cookie set: {key} = {value}");
}

// Метод для получения значения из cookie
Ссылок: 0
public IActionResult GetCookie(string key)
{
    // Получение значения из cookie по указанному ключу
    var value = Request.Cookies[key];
    return Content(value ?? "Cookie not found");
}
```

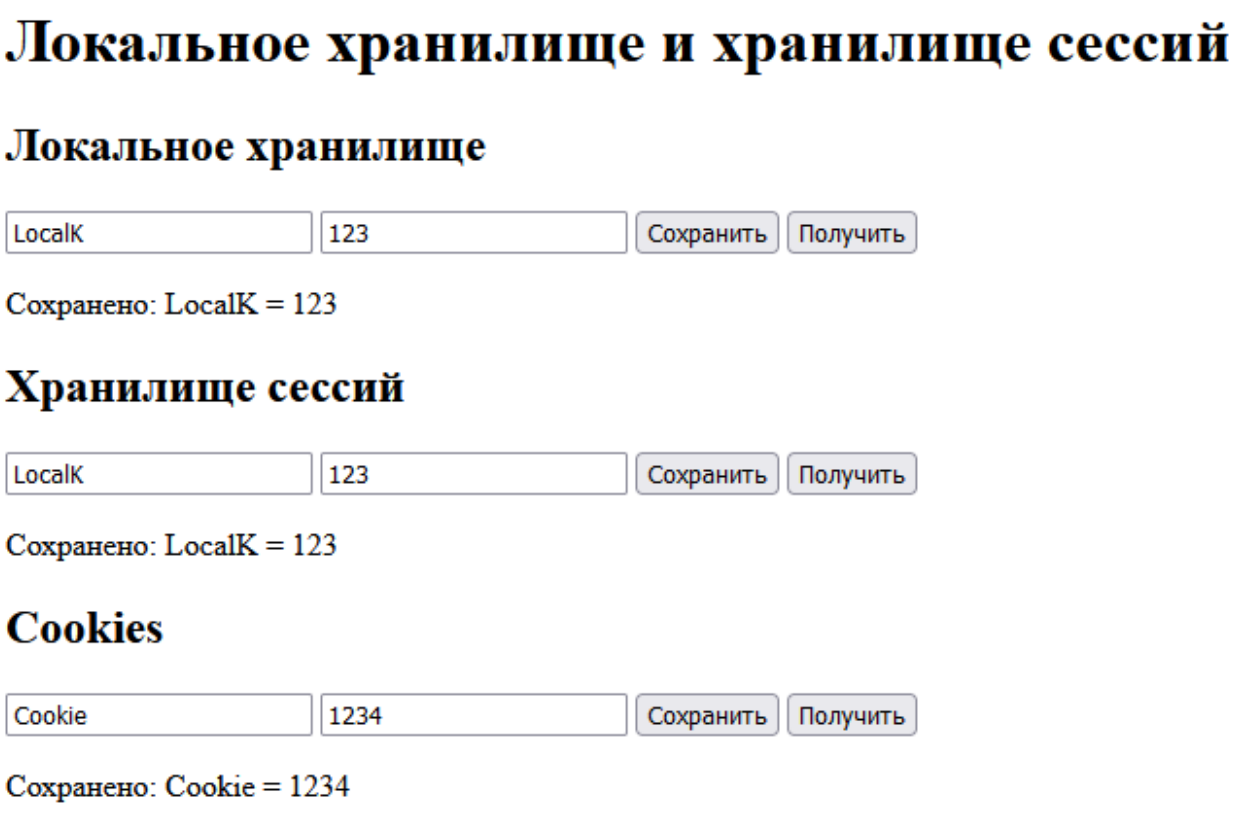
Для сессий данные хранятся на сервере, а идентификатор сессии сохраняется в cookie на стороне клиента. Сессия задается 2 значениями, ключ сессии и само значение которое она хранит, так же имеем метод для того, чтобы задавать куки, что позволяет сохранять небольшие кусочки данных на стороне клиента, которые отправляются на сервер с каждым запросом, они имеют так же ключ и значения, по ключу, есть возможно получить значение как сессии, так и куки.

```
<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";
    }
```

Далее в представлении у нас содержится локальное хранилище, 2 вида - Local Storage и Session Storage. предоставляют способы хранения данных непосредственно в браузере. Local Storage сохраняет данные без срока действия, пока они не будут удалены вручную. Session Storage сохраняет данные только на время сессии (до закрытия вкладки браузера). Так же здесь реализован интерфейс для работы с куки на стороне клиента, можно задать значение и получить их по ключу. Перейдем к тестированию.



Заполним поля и нажмем сохранить



Наши данные сохранены, теперь попоробуем получить все данные по их ключам

Локальное хранилище и хранилище сессий

Локальное хранилище

LocalK | Значение | Сохранить | Получить

Значение: LocalK = 123

Хранилище сессий

LocalK | Значение | Сохранить | Получить

Значение: LocalK = 123

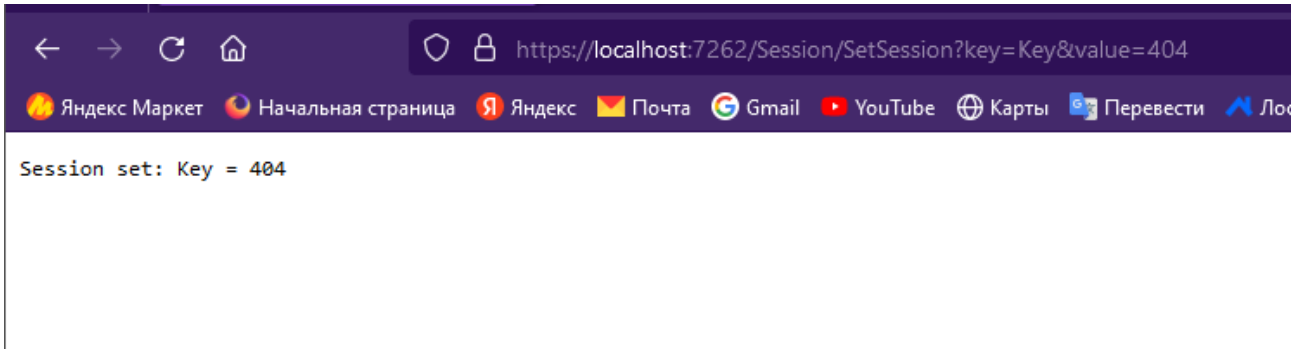
Cookies

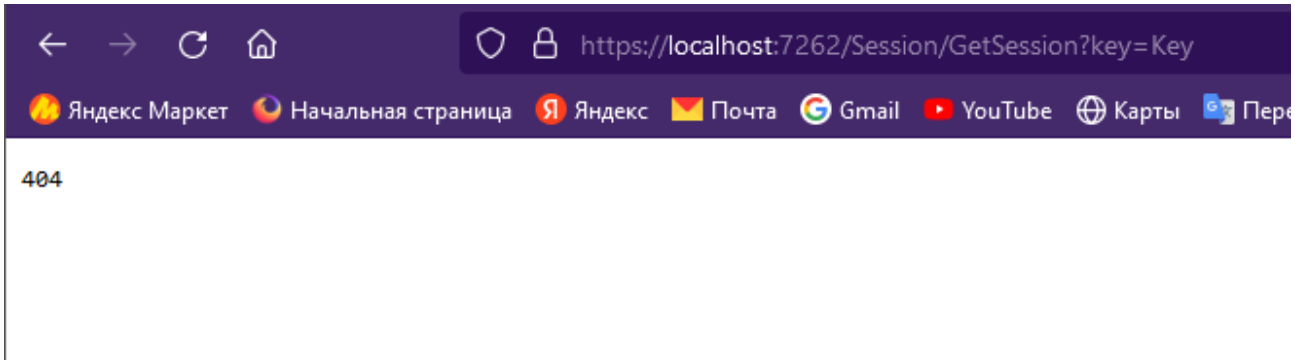
Cookie | Значение | Сохранить | Получить

Значение: Cookie = 1234

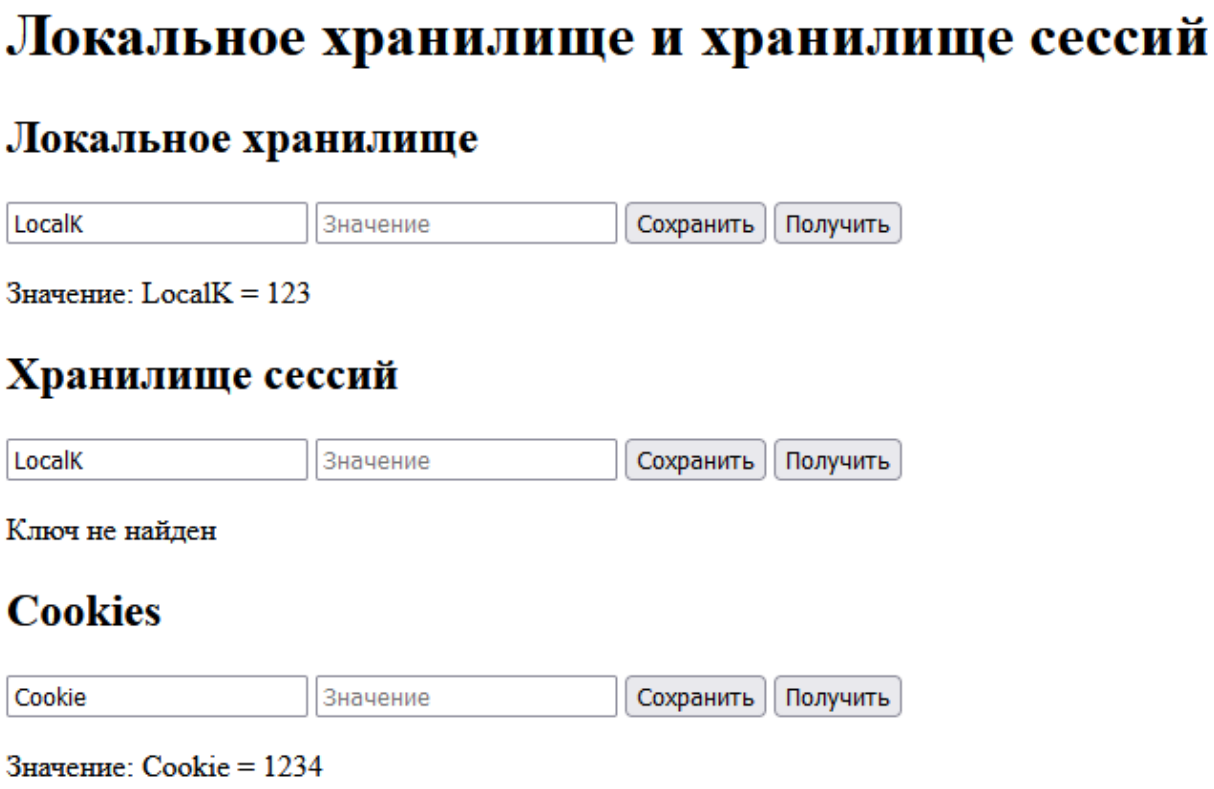
теперь зададим значение сесии через переход на действие в браузере

Задали ключ и значение и переходим.





Как видим значение задано и мы можем его получить, теперь закроем приложение и зайдем снова чтобы проверить очистку локального хранилища с данными, которые должны быть удалены после закрытия браузера.



Как видим данные сохраняются корректно.

Выводы о типах состояний:

Сессии:

Преимущества: Удобны для хранения небольших данных на длительный срок. Данные хранятся на стороне клиента и автоматически отправляются с каждым запросом на сервер.

Недостатки: Ограниченный объем хранения и возможные проблемы с безопасностью, так как данные хранятся на стороне клиента.

Cookies:

Преимущества: Удобны для хранения небольших данных на длительный срок. Данные хранятся на стороне клиента и автоматически отправляются с каждым запросом на сервер.

Недостатки: Ограниченный объем хранения и возможные проблемы с безопасностью, так как данные хранятся на стороне клиента.

Local Storage и Session Storage:

Преимущества: Удобны для хранения данных на стороне клиента без необходимости отправлять их на сервер. Local Storage сохраняет данные на неопределенный срок, Session Storage — только на время сессии.

Недостатки: Данные доступны только в браузере и могут быть потеряны при очистке браузера или завершении сессии.\

Код приложения:

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

var builder = WebApplication.CreateBuilder(args);

// Добавление служб в контейнер
builder.Services.AddControllersWithViews();

// Добавление службы сессий
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(30); // Время жизни сессии
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});

var app = builder.Build();

// Настройка конвейера обработки HTTP-запросов
if (app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
}
else
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

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

app.UseRouting();

app.UseAuthorization();

// Использование сессий
app.UseSession();

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

```
app.Run();


using Microsoft.AspNetCore.Mvc;

namespace Lab8.Controllers
{
    public class SessionController : Controller
    {
        // Метод для установки значения в сессию
        public IActionResult SetSession(string key, string value)
        {
            // Сохранение значения в сессии с указанным ключом
            HttpContext.Session.SetString(key, value);
            return Content($"Session set: {key} = {value}");
        }

        // Метод для получения значения из сессии
        public IActionResult GetSession(string key)
        {
            // Получение значения из сессии по указанному ключу
            var value = HttpContext.Session.GetString(key);
            return Content(value ?? "Session not found");
        }

        // Метод для установки значения в cookie
        public IActionResult SetCookie(string key, string value)
        {
            // Опции cookie, включая время жизни (30 минут)
            CookieOptions option = new CookieOptions
            {
                Expires = DateTime.Now.AddMinutes(30)
            };
            // Сохранение cookie с указанным ключом и значением
            Response.Cookies.Append(key, value, option);
            return Content($"Cookie set: {key} = {value}");
        }

        // Метод для получения значения из cookie
        public IActionResult GetCookie(string key)
        {
            // Получение значения из cookie по указанному ключу
            var value = Request.Cookies[key];
            return Content(value ?? "Cookie not found");
        }

        // Метод для отображения главной страницы
        public IActionResult Index()
        {
            return View();
        }
    }
}


@{
    ViewData["Title"] = "Локальное хранилище и хранилище сессий";
}

<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>

<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>
```