**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»**

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

**Выполнил: студент группы: 231-339**

Карапетян Нвер Каренович\_\_\_\_\_\_\_\_\_\_\_

(Фамилия И.О.)



**Дата, подпись:** 05.05.25 \_\_\_\_\_\_\_\_\_\_\_\_

(Дата)(Подпись)

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

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

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

(Дата)(Подпись)

**Москва**

**2025**

Цель:

Познакомиться с основами работы с состояниями в веб-приложениях на платформе ASP.NET Core для управления информацией на клиентской и серверной сторонах.

Задачи:

* Изучить и применить работу с состояниями на стороне сервера (например, сессии).
* Исследовать и применить работу с состояниями на стороне клиента (например, cookies, localStorage, sessionStorage).
* Реализовать примеры использования различных типов состояний для хранения и передачи данных между клиентом и сервером.
* Протестировать работу приложения при использовании различных состояний.

Ход работы

В веб-приложениях HTTP-протокол по умолчанию без сохранения состояния (stateless), то есть каждый запрос обрабатывается независимо. Для реализации берущихся «в памяти» данных используются механизмы управления состоянием:

* **Серверная сессия (Session)**. Сервер хранит данные, а клиент получает лишь идентификатор сессии в виде cookie (ASP.NET\_SessionId), по которому данные извлекаются.
* **Cookies.** Небольшие пары «ключ–значение», хранящиеся на стороне клиента и автоматически отправляемые с каждым запросом к тому же домену.
* **localStorage / sessionStorage**: JavaScript-хранилища в браузере. Данные в localStorage живут пока их не удалят вручную, в sessionStorage — до закрытия вкладки.

Использование состояний позволяет сохранять пользовательские настройки, корзины покупок, токены аутентификации и др., не передавая их в каждом запросе явно.

Серверная сессия

Для работы с сессиями на стороне сервера в ASP.NET Core используется ISession, подкрепленный кэшем (IDistributedCache). Данные хранятся в памяти приложения, а клиенту выдаётся лишь идентификатор сессии в cookie.

Листинг 1. Фрагмент из Program.cs с настройкой работы с сессиями.

// Распределенный кэш (требуется для хранения сессий)

builder.Services.AddDistributedMemoryCache();

// Регистрация сессий с таймаутом 20 минут

builder.Services.AddSession(options =>

{

    options.IdleTimeout = TimeSpan.FromMinutes(20);

    options.Cookie.HttpOnly = true;     // защита от JS-доступа

    options.Cookie.IsEssential = true;  // обязательно для работы

});

var app = builder.Build();

// Подключение middleware сессий

app.UseSession();

В контроллере StateController.cs реализованы методы, работающие с серверными сессиями:

Листинг 2. Методы контроллера StateController для работы с серверными сессиями.

// Установить значение в сессию

[HttpGet("/session/set/{value}")]

public IActionResult SessionSet(string value)

{

    HttpContext.Session.SetString("SessionKey", value);

    return Ok($"На серверную сессию записано значение: {value}");

}

// Получить значение из сессии

[HttpGet("/session/get")]

public IActionResult SessionGet()

{

    var value = HttpContext.Session.GetString("SessionKey") ?? "<не задано>";

    return Ok($"С серверной сессии прочитано значение: {value}");

}

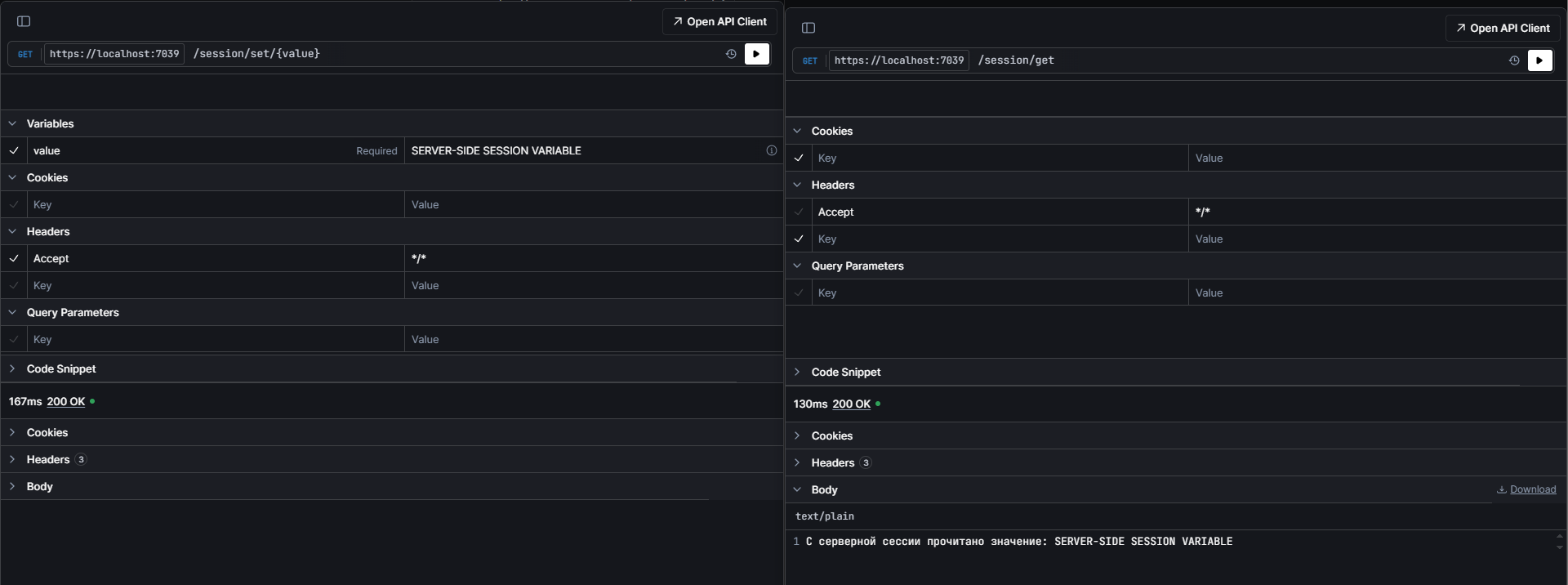


Рисунок . Результаты работы с сессионными методами контроллера StateController.

При этом куки с ключом .AspNetCore.Session автоматически устанавливается клиенту и связывает запросы с данными на сервере.

Cookies

В том же контроллере реализованы методы, работающие с куки:

Листинг 3. Методы контроллера StateController для работы с куками.

// Установить куки

[HttpGet("/cookie/get/{value}")]

public IActionResult CookieSet(string value)

{

    Response.Cookies.Append("MyCookie", value, new CookieOptions

    {

        HttpOnly = false,

        Expires = DateTimeOffset.UtcNow.AddMinutes(10)

    });

    return Ok($"Cookie установлен: {value}");

}

// Прочитать куки

[HttpGet("/cookie/set")]

public IActionResult CookieSet()

{

    if (Request.Cookies.TryGetValue("MyCookie", out var value))

        return Ok($"Cookie = {value}");

    return Ok($"Cookie не найден");

}

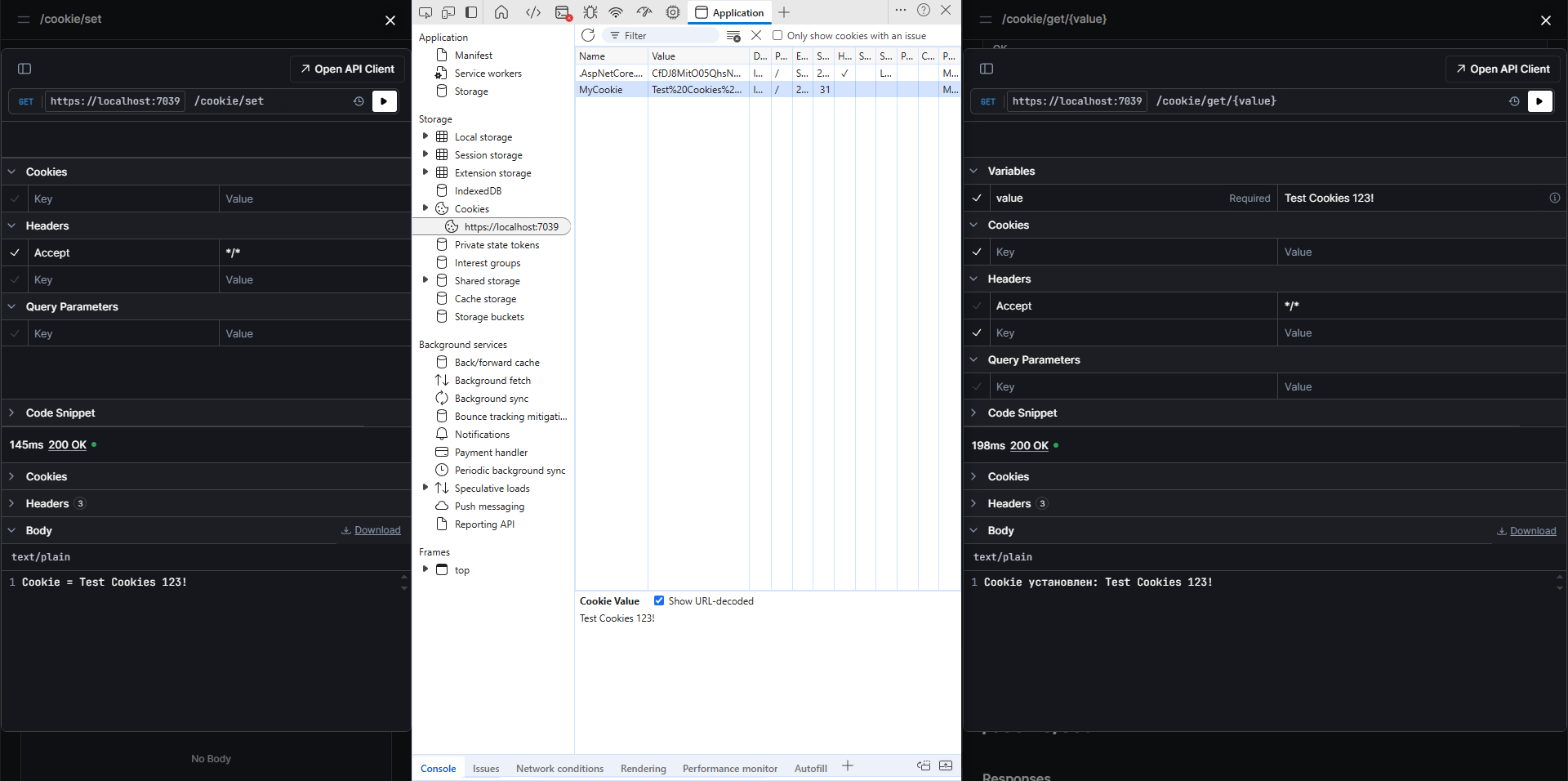


Рисунок . Пример работы с куками через методы контроллера StateController.

Клиентские состояния (localStorage / sessionStorage)

Чтобы продемонстрировать клиентские хранилища, в проект добавлена статическая страница wwwroot/index.html, которая загружается без контроллеров:

Листинг 4. Index.html.

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8" />

    <title>Клиентские состояния</title>

    <script src="/js/app.js" defer></script>

</head>

<body>

    <h1>Клиентские состояния</h1>

    <button id="saveLocal">Сохранить в localStorage</button>

    <button id="loadLocal">Прочитать из localStorage</button>

    <button id="saveSession">Сохранить в sessionStorage</button>

    <button id="loadSession">Прочитать из sessionStorage</button>

    <h3 id="out"></h3>

</body>

</html>

Также в проекте был добавлен JavaScript-файл wwwroot/js/app.js, в котором написана логика и функциональность кнопок, которые сохраняют и читают данные из localStorage и sessionStorage:

Листинг 5. app.js.

const out = document.getElementById("out");

document.getElementById("saveLocal").onclick = () => {

    const value = prompt("Значение для localStorage: ");

    localStorage.setItem("localStorageKey", value);

    out.textContent = "localStorage успешно сохранен"

};

document.getElementById("loadLocal").onclick = () => {

    out.textContent = "localStorage: " + localStorage.getItem("localStorageKey");

};

document.getElementById("saveSession").onclick = () => {

    const value = prompt("Значение для sessionStorage: ");

    sessionStorage.setItem("sessionStorageKey", value);

    out.textContent = "sessionStorage успешно сохранен";

};

document.getElementById("loadSession").onclick = () => {

    out.textContent = "sessionStorage: " + sessionStorage.getItem("sessionStorageKey");

};

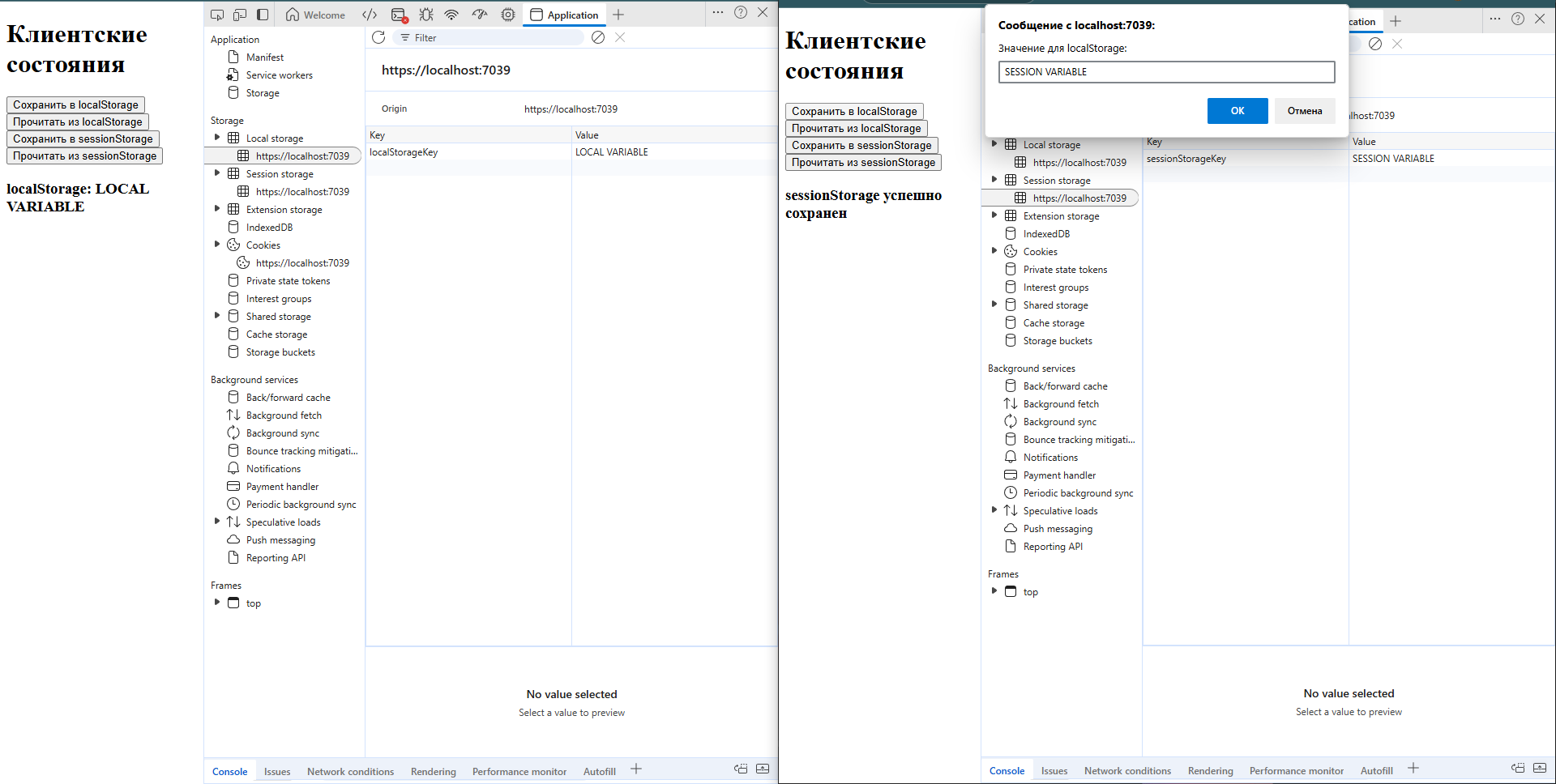


Рисунок . Результаты работы с localStorage и sessionStorage.

При сохранении в **localStorage** данные сохраняются между перезапусками вкладки/браузера. В **sessionStorage** — живут до закрытия вкладки.