ISSN 2181-3187

MIJOZLAR BILAN ISHLASH

Mo'minov Sarvarbek Ulug'bek o'g'li
Farg'ona davlat unversiteti talabasi
mominov.net@gmail.com
Yusupov Mirsaid Abdulaziz o'g'li
Farg'ona davlat universiteti o'qituvchisi
mirsaidbeky@gmail.com
Onarkulov Maksadjon Karimberdiyevich
Farg'ona davlat universiteti dotsenti
maxmaqsad@gmail.com

Annotatsiya: Ushbu maqolada ASP.NET Core freymvorkida mijozlar bilan ishlash bo'yicha eng yaxshi amaliyotlar va usullar batafsil ko'rib chiqiladi. Maqolada mijozlar so'rovlarini samarali boshqarish, HTTP kontekstidan to'g'ri foydalanish, asinxron dasturlash, keshlashtirish va mijoz tomonidagi ma'lumotlarni validatsiya qilish kabi muhim mavzular yoritilgan. Shuningdek, mijozlar bilan ishlashda xavfsizlik, unumdorlik va kengayuvchanlikni ta'minlash uchun Clean Architecture va SOLID tamoyillarini qo'llash usullari ham ko'rsatilgan. Maqola ASP.NET Core ilovalarini ishlab chiquvchilar uchun mijozlar bilan ishlashda yuqori samaradorlikka erishish va xatolardan qochish bo'yicha amaliy tavsiyalar beradi.

Kalit so'zlar: ASP.NET Core, mijozlar bilan ishlash, HTTP kontekst, asinxron dasturlash, keshlashtirish, validatsiya, xavfsizlik, Clean Architecture, SOLID tamoyillari, unumdorlik, kengayuvchanlik.

Annotation: This article provides a detailed overview of best practices and methods for handling clients in the ASP.NET Core framework. The article covers important topics such as effective client request management, proper use of HTTP context, asynchronous programming, caching, and client-side data validation. Additionally, it demonstrates methods for applying Clean Architecture and SOLID principles to ensure security, performance, and scalability when working with clients.

ISSN 2181-3187

The article offers practical recommendations for ASP.NET Core developers to achieve high efficiency and avoid errors in client handling.

Keywords: ASP.NET Core, client handling, HTTP context, asynchronous programming, caching, validation, security, Clean Architecture, SOLID principles, performance, scalability.

Аннотация: В данной статье подробно рассматриваются лучшие практики и методы работы с клиентами в среде ASP.NET Core. В статье освещаются такие важные темы, как эффективное управление клиентскими запросами, правильное использование HTTP-контекста, асинхронное программирование, кэширование и проверка данных на стороне клиента. Кроме того, демонстрируются методы применения принципов Clean Architecture и SOLID для обеспечения безопасности, производительности и масштабируемости при работе с клиентами. Статья предлагает практические рекомендации для разработчиков ASP.NET Core по достижению высокой эффективности и избежанию ошибок при обработке клиентских взаимодействий.

Ключевые слова: ASP.NET Core, работа с клиентами, HTTP-контекст, асинхронное программирование, кэширование, валидация, безопасность, Clean Architecture, принципы SOLID, производительность, масштабируемость.

Kirish

Mijozlar bilan samarali ishlash har qanday veb-ilovaning muvaffaqiyati uchun hal qiluvchi ahamiyatga ega. Bu nafaqat foydalanuvchi tajribasini yaxshilaydi, balki tizimning unumdorligi, xavfsizligi va kengayuvchanligiga ham bevosita ta’sir ko‘rsatadi. ASP.NET Core freymvorki mijozlar bilan ishlashni osonlashtiradigan va optimallashtiradigan ko‘plab vositalar va mexanizmlarni taklif etadi. Biroq, ushbu vositalardan to‘g‘ri foydalanish va eng yaxshi amaliyotlarga rioya qilish muhimdir.

Ushbu maqolada ASP.NET Core da mijozlar bilan ishlashning asosiy jihatlari, jumladan, HTTP kontekstidan to‘g‘ri foydalanish, asinxron dasturlash yordamida unumdorlikni oshirish, keshlashtirish strategiyalari, mijoz tomonidan yuborilgan

ISSN 2181-3187

ma’lumotlarni validatsiya qilish usullari va xavfsizlikni ta’minlash choralari ko‘rib chiqiladi. Shuningdek, Clean Architecture va SOLID tamoyillari kabi arxitektura yondashuvlarining mijozlar bilan ishlash jarayonini qanday yaxshilashi mumkinligi tahlil qilinadi. Maqolaning maqsadi ASP.NET Core ishlab chiquvchilariga mijozlar bilan ishlashda yuqori samaradorlikka erishish, keng tarqalgan xatolardan qochish va ishonchli, kengayuvchan ilovalar yaratish bo‘yicha amaliy bilim va ko‘nikmalar berishdan iborat.

Asosiy qism

ASP.NET Core freymvorkida mijozlar bilan samarali ishlash veb-ilovalarning unumdorligi, xavfsizligi va kengayuvchanligi uchun muhim ahamiyat kasb etadi. Ushbu bo‘limda mijozlar so‘rovlarini boshqarish, HTTP kontekstidan foydalanish, asinxron dasturlash, ma’lumotlarni validatsiya qilish, keshlashtirish, xavfsizlikni ta’minlash va arxitektura yondashuvlari kabi asosiy jihatlar batafsil ko‘rib chiqiladi.

Mijoz so‘rovlarini qabul qilish va qayta ishlash

ASP.NET Core ilovalari mijozlardan keladigan HTTP so‘rovlarini qabul qilish va ularga javob qaytarish uchun bir qator mexanizmlarga ega. Eng keng tarqalgan yondashuvlar bu kontrollerlar (Controllers) va minimal API (Minimal APIs) lardir. Kontrollerlar an’anaviy MVC (Model-View-Controller) arxitekturasining bir qismi bo‘lib, so‘rovlarni marshrutlash, biznes mantiqni chaqirish va javoblarni shakllantirish uchun ishlatiladi. Minimal APIlar esa ASP.NET Core 6.0 da taqdim etilgan bo‘lib, kamroq kod bilan tezkor va yengil APIlar yaratish imkonini beradi.

Mijoz so‘rovini qabul qilganda, ASP.NET Core uni marshrutlash (routing) mexanizmi orqali tegishli ishlov beruvchiga (handler) yo‘naltiradi. Bu ishlov beruvchi kontrollerdagi aksiya (action) metodi yoki minimal API dagi delegat bo‘lishi mumkin. So‘rovni qayta ishlash jarayonida so‘rov tanasi (request body), sarlavhalar (headers), so‘rov satri parametrlari (query string parameters) va marshrut parametrlari (route parameters) kabi ma’lumotlardan foydalanish mumkin. ASP.NET Core bu ma’lumotlarni avtomatik ravishda modellar bilan bog‘lash (model binding) imkoniyatini taqdim etadi, bu esa ishlab chiquvchilar ishini ancha osonlashtiradi.

ISSN 2181-3187

HTTP Kontekstidan (HttpContext) foydalanish

HttpContext obyekti joriy HTTP so‘rovi va javobi bilan bog‘liq barcha ma’lumotlarni o‘zida saqlaydi. Unga **Request**, **Response**, **User**, **Session** (agar sozlan_gan bo‘lsa), **Items** (so‘rov davomida ma’lumot almashish uchun) kabi xususiyatlar orqali kirish mumkin. **HttpContext** ga odatda kontrollerlarda **ControllerBase.HttpContext** xususiyati orqali yoki boshqa joylarda **IHttpContextAccessor** interfeysi yordamida kiriladi.

Microsoft Learn hujjatlarida ta’kidlanishicha, **HttpContext** ni fon rejimida ishlaydigan potoklarda (background threads) to‘g‘ridan-to‘g‘ri ishlatish xavfli, chunki u potoklar uchun xavfsiz (thread-safe) emas. Agar fon rejimida **HttpContext** ma’lumotlariga ehtiyoj bo‘lsa, kerakli ma’lumotlarni so‘rovni qayta ishlash jarayonida nusxalash va fon potokiga uzatish yoki **IHttpContextAccessor** dan ehtiyotkorlik bilan foydalanish tavsiya etiladi. Shuningdek, **HttpContext** ni maydon (field) sifatida saqlab qo‘yish ham tavsiya etilmaydi; buning o‘rniga **IHttpContextAccessor** ni saqlash va kerak bo‘lganda undan joriy kontekstni olish kerak (Microsoft Learn, n.d.).

Mijoz uzilib qolganda, **HttpContext.RequestAborted** bekor qilish tokeni (cancellation token) ishga tushadi. Uzoq davom etadigan so‘rovlarni bekor qilish uchun ushbu tokendan foydalanish kerak.

Asinxron dasturlash (Asynchronous Programming)

ASP.NET Core ilovalarining unumdorligini oshirishda asinxron dasturlash muhim rol o‘ynaydi. Ko‘plab mijozlarga bir vaqtning o‘zida xizmat ko‘rsatish uchun ilova potoklarni (threads) bloklamasligi kerak. Ma’lumotlar bazasiga murojaat qilish, fayllar bilan ishlash yoki tashqi xizmatlarga so‘rov yuborish kabi I/O operatsiyalari asinxron tarzda bajarilishi kerak. Buning uchun **async** va **await** kalit so‘zlaridan foydalaniladi.

Microsoft Learn hujjatlarida qayd etilishicha, barcha I/O operatsiyalari asinxron bo‘lishi kerak. **HttpRequest.Body** yoki **HttpResponse.Body** bilan ishlaganda sinxron o‘qish yoki yozishdan qochish kerak, chunki bu potoklar pulining (thread pool) tugashiga olib kelishi mumkin. Buning o‘rniga **ReadAsync**, **WriteAsync** kabi

ISSN 2181-3187

asinxron metodlardan foydalanish lozim. Xuddi shunday, form ma’lumotlarini o‘qish uchun **HttpContext.Request.Form** o‘rniga **HttpContext.Request.ReadFormAsync** dan foydalanish tavsiya etiladi (Microsoft Learn, n.d.).

Asinxronlikni bloklaydigan **Task.Wait()** yoki **Task<TResult>.Result** chaqiruvlaridan qochish kerak. Shuningdek, umumiy kod yo‘llarida blokirovkalarni (locks) ishlatmaslik kerak. **Task.Run()** ni chaqirib, darhol uni **await** qilish ham ortiqcha, chunki ASP.NET Core allaqachon ilova kodini oddiy potoklar puli potoklarida ishga tushiradi.

Ma’lumotlarni validatsiya qilish (Data Validation)

Mijoz tomonidan yuborilgan ma’lumotlarni validatsiya qilish ilovaning barqarorligi va xavfsizligi uchun juda muhimdir. ASP.NET Core ma’lumotlarni validatsiya qilish uchun bir necha usullarni taklif etadi:

1. **Model validatsiyasi (Model Validation):** Ma’lumotlar atributlari (Data Annotations) yordamida (masalan, **[Required]**, **[StringLength]**, **[Range]**) yoki **IValidatableObject** interfeysini amalga oshirish orqali modellarni validatsiya qilish mumkin. Kontroller aksiya metodlarida **ModelState.IsValid** xususiyatini tekshirish orqali validatsiya natijalarini olish mumkin.
2. **FluentValidation kabi tashqi kutubxonalar:** Murakkabroq validatsiya qoidalari uchun FluentValidation kabi kutubxonalardan foydalanish mumkin. Ular yanada moslashuvchan va o‘qilishi oson validatsiya logikasini yaratishga yordam beradi.

Mijoz tomonida (client-side) validatsiya ham muhim, chunki u serverga keraksiz so‘rovlar yuborilishining oldini oladi va foydalanuvchi tajribasini yaxshilaydi. Biroq, mijoz tomonidagi validatsiya hech qachon server tomonidagi validatsiyaning o‘rnini bosa olmaydi, chunki mijoz tomonidagi validatsiyani chetlab o‘tish mumkin.

Keshlashtirish (Caching)

ISSN 2181-3187

Keshlashtirish tez-tez murojaat qilinadigan ma’lumotlarni vaqtincha saqlash orqali ilova unumdorligini sezilarli darajada oshirishi mumkin. ASP.NET Core bir necha turdagi keshlashtirishni qo‘llab-quvvatlaydi:

- **Xotirada keshlashtirish (In-memory caching): IMemoryCache** interfeysi orqali amalga oshiriladi. Ma’lumotlar ilovaning o‘z xotirasida saqlanadi. Bu bitta serverli ilovalar uchun mos keladi.
- **Taqsimlangan keshlashtirish (Distributed caching): IDistributedCache** interfeysi orqali amalga oshiriladi. Ma’lumotlar Redis yoki SQL Server kabi tashqi kesh omborida saqlanadi. Bu ko‘p serverli (load-balanced) muhitlar uchun zarur.
- **Javoblarni keshlashtirish (Response caching):** HTTP javoblarini keshlaydi, bu esa keyingi bir xil so‘rovlarga tezroq javob berish imkonini beradi. **[ResponseCache]** atributi yordamida sozlanadi.
- **Output caching:** ASP.NET Core 7.0 da taqdim etilgan bo‘lib, butun HTTP javobini yoki uning qismlarini keshlaydi va yanada moslashuvchan boshqaruvni ta’minlaydi.

Microsoft Learn hujjatlarida ta’kidlanishicha, tez-tez ishlatiladigan katta obyektlarni keshlashtirish qimmat ajratmalarning oldini oladi. Biroz eskirgan ma’lumotlar qabul qilinadigan bo‘lsa, ma’lumotlar bazasidan yoki masofaviy xizmatdan olingan tez-tez murojaat qilinadigan ma’lumotlarni keshlashtirishni ko‘rib chiqish kerak (Microsoft Learn, n.d.).

Xavfsizlik (Security)

Mijozlar bilan ishlashda xavfsizlikka alohida e’tibor berish kerak. ASP.NET Core xavfsizlikni ta’minlash uchun bir qator vositalarni taklif etadi:

- **Autentifikatsiya (Authentication):** Foydalanuvchining shaxsini tasdiqlash. ASP.NET Core Identity, JWT tokenlari, OAuth 2.0 kabi turli autentifikatsiya sxemalarini qo‘llab-quvvatlaydi.
- **Avtorizatsiya (Authorization):** Autentifikatsiyalangan foydalanuvchining ma’lum resurslarga kirish huquqlarini tekshirish. Rolga

ISSN 2181-3187

asoslangan (role-based) va siyosatga asoslangan (policy-based) avtorizatsiyani qo‘llab-quvvatlaydi.

- **CSRF (Cross-Site Request Forgery) hujumlaridan himoya:** Antiforgery tokenlari yordamida amalga oshiriladi.
- **XSS (Cross-Site Scripting) hujumlaridan himoya:** Ma’lumotlarni to‘g‘ri kodlash (encoding) orqali oldini olinadi.
- **HTTPS dan foydalanish:** Ma’lumotlarni uzatishda shifrlashni ta’minlaydi.
- **Kiruvchi ma’lumotlarni validatsiya qilish:** Yuqorida aytib o‘tilganidek, bu ham xavfsizlikning muhim qismidir.

Arxitektura yondashuvlari

Mijozlar bilan ishlashni samarali tashkil etish uchun to‘g‘ri arxitektura yondashuvlarini tanlash muhim. Medium maqolasida (Singh, 2024) Clean Architecture yoki Onion Architecture kabi yondashuvlardan foydalanish tavsiya etiladi. Bu yondashuvlar mas’uliyatlarni ajratish (separation of concerns) va bog‘liqliklarni (dependencies) boshqarish orqali ilovani yanada tartibli, sinovdan o‘tkazilishi oson va qo‘llab-quvvatlanishi oson qiladi. Odatda Presentation (Taqdimot), Application (Ilova), Domain (Domen) va Infrastructure (Infratuzilma) kabi qatlamlardan foydalaniladi.

Bog‘liqliklarni kiritish (Dependency Injection - DI): ASP.NET Core o‘rnatilgan DI konteyneriga ega. DI xizmatlarning hayot aylanishini (service lifetimes) va bog‘liqliklarini samarali boshqarishga yordam beradi. Bu kodning bog‘liqligini kamaytiradi va uni sinovdan o‘tkazishni osonlashtiradi.

SOLID tamoyillari: Yagona mas’uliyat (Single Responsibility), Ochiq/Yopiq (Open/Closed), Liskov almashtirish (Liskov Substitution), Interfeyslarni ajratish (Interface Segregation) va Bog‘liqlik inversiyasi (Dependency Inversion) tamoyillariga rioya qilish yanada barqaror va kengayuvchan kod yozishga yordam beradi (Singh, 2024).

Mijozlar bilan aloqa uchun HttpClientFactory dan foydalanish

ISSN 2181-3187

Agar ilovangiz boshqa HTTP xizmatlariga mijoz sifatida murojaat qilsa, **HttpClient** dan to‘g‘ri foydalanish muhim. **HttpClient** nusxalarini to‘g‘ridan-to‘g‘ri yaratish va yo‘q qilish o‘rniga **HttpClientFactory** dan foydalanish kerak. **HttpClientFactory** HTTP ulanishlarini pulllash (pooling) orqali unumdorlik va ishonchlilikni optimallashtiradi va soketlarning tugab qolishi kabi muammolarning oldini oladi (Microsoft Learn, n.d.).

Ushbu eng yaxshi amaliyotlarga rioya qilish ASP.NET Core ilovalarida mijozlar bilan ishlash jarayonini sezilarli darajada yaxshilashi, unumdorlikni oshirishi, xavfsizlikni kuchaytirishi va ilovani qo‘llab-quvvatlashni osonlashtirishi mumkin.

Xulosa

ASP.NET Core da mijozlar bilan samarali ishlash zamonaviy veb-ilovalarni yaratishning muhim tarkibiy qismidir. Ushbu maqolada ko'rib chiqilganidek, mijozlar so'rovlarini to'g'ri qabul qilish va qayta ishlash, HTTP kontekstidan oqilona foydalanish, asinxron dasturlash imkoniyatlarini to'liq ishga solish, ma'lumotlarni sinchkovlik bilan validatsiya qilish, keshlashtirish strategiyalarini qo'llash va xavfsizlik choralarini ko'rish ilovaning umumiy sifati, unumdorligi va ishonchliligiga bevosita ta'sir ko'rsatadi.

Kontrollerlar va minimal APIlar orqali so'rovlarni boshqarish, **HttpContext** bilan ehtiyotkorlik bilan ishlash, ayniqsa fon potoklarida, hamda **async/await** dan keng foydalanish potoklar pulining samarasini oshiradi va ilovaning ko'plab mijozlarga bir vaqtda xizmat ko'rsatish qobiliyatini yaxshilaydi. Model validatsiyasi va tashqi kutubxonalar yordamida ma'lumotlarning to'g'riligini ta'minlash, shuningdek, xotirada va taqsimlangan keshdan unumli foydalanish javob vaqtini qisqartiradi.

Xavfsizlik nuqtai nazaridan, autentifikatsiya, avtorizatsiya, CSRF va XSS hujumlaridan himoyalanish kabi mexanizmlarni joriy etish shart. Bundan tashqari, Clean Architecture va SOLID tamoyillariga asoslangan holda ilovani loyihalash, bog'liqliklarni kiritish (DI) mexanizmidan unumli foydalanish kodning barqarorligi, sinovdan o'tkazilishi osonligi va qo'llab-quvvatlanishini ta'minlaydi. Boshqa

ISSN 2181-3187

xizmatlarga mijoz sifatida murojaat qilganda **HttpClientFactory** dan foydalanish ham resurslarni tejash va ishonchlilikni oshirishga yordam beradi.

Xulosa qilib aytganda, ASP.NET Core ishlab chiquvchilari ushbu maqolada keltirilgan eng yaxshi amaliyotlar va tavsiyalarga amal qilish orqali mijozlar bilan ishlash jarayonini optimallashtirishi, yuqori sifatli, unumdor va xavfsiz veb-ilovalar yaratishi mumkin. Doimiy o'rganish va yangi texnologiyalardan xabardor bo'lish ushbu sohada muvaffaqiyatga erishishning kalitidir.

Foydalanilgan adabiyotlar

1. Karimberdiyevich, O. M., & Abdulaziz o‘g’li, Y. M. (2024). SUN’IY INTELLEKTNING AFZALLIKLARI VA KAMCHILIKLARI. *IZLANUVCHI*, *1*(1), 75-85.

2. Karimberdiyevich, O. M., & Abdulaziz o‘g‘li, Y. M. (2024). NEYRO KOMPYUTERLAR. *YANGI O ‘ZBEKISTON, YANGI TADQIQOTLAR JURNALI*, *1*(5), 19-27.

3. Karimberdiyevich, O. M., & Abdulaziz o‘g‘li, Y. M. (2024). K-YAQIN QO'SHNI ALGORITMI. *IZLANUVCHI*, *1*(1), 122-124.

4. Abdulaziz o'g'li, Y. M. (2025). WPFDA ANIMATSIYA YARATISHNI QO’LLANISHI. *MODERN PROBLEMS IN EDUCATION AND THEIR SCIENTIFIC SOLUTIONS*, *1*(4), 172-175.

5. Abdulaziz o’g’li, Y. M. (2025). MOLIYA VA HISOB–KITOB ILOVALARIDA WPF BILAN ISHLASH. *MODERN PROBLEMS IN EDUCATION AND THEIR SCIENTIFIC SOLUTIONS*, *1*(4), 189-193.

6. Karimberdiyevich, O. M. (2024). NEYROEMULYATORLAR VA ULARNING QO'LLANILISHI. *YANGI O ‘ZBEKISTON, YANGI TADQIQOTLAR JURNALI*, *1*(5), 82-89.