FOYDALANUVCHI AUTENTIFIKATSIYASI VA AVTORIZATSIYASIDA CORS VA DOMENLARARO SO'ROVLAR BILAN ISHLASH.

Yusupov Mirsaid Abdulaziz o'g'li
Farg'ona Davlat Universiteti
Amaliy matematika va informatika kafedrasi o'qituvchisi
mirsaidbeky@gmail.com
Metinboyeva Fotimaxon Mirqo'zi qizi
Farg'ona Davlat Universiteti 2-kurs talabasi
f6149133@gmail.com

Annotatsiya: *CORS (Cross-Origin Resource Sharing) — bu web ilovalari orasidagi domenlararo so'rovlarni boshqarish mexanizmidir. Web brauzerlari xavfsizlik nuqtai nazaridan, bir domenning boshqa domenlar bilan resurslarni almashishiga ruxsat bermaydi. CORS yordamida domenlararo so'rovlar amalga oshirilishi mumkin. Ushbu maqolada CORS qanday ishlashi, qanday qilib konfiguratsiya qilish, xavfsizlik masalalari va web ilovalari uchun afzalliklari va kamchiliklari haqida batafsil ma'lumot beriladi. Maqola shuningdek, CORS bilan ishlashda uchraydigan umumiy xatoliklar va ularni hal qilish usullari haqida ham fikr yuritadi.*

Kalit so'zlar: *CORS, Domenlararo so'rovlar, Xavfsizlik, Web ilovalar, HTTP so'rovlar, So'rovlarni boshqarish, Xatoliklar, Cross-origin resurslar, Ruxsatlar, Server konfiguratsiyasi.*

Аннотация: *CORS (Cross-Origin Resource Sharing) — это механизм управления кросс-доменными запросами между веб-приложениями. Веб-браузеры по умолчанию блокируют доступ к ресурсам с других доменов по соображениям безопасности. С помощью CORS можно разрешить*

выполнение таких запросов. В статье подробно рассматривается, как работает CORS, как его настроить, проблемы безопасности, а также преимущества и недостатки использования CORS для веб-приложений. Также рассматриваются распространенные ошибки и способы их решения при работе с CORS.

Ключевые слова: *CORS, Кросс-доменные запросы, HTTP-запросы, Безопасность, Кросс-доменные ресурсы, Кросс-доменные ресурсы, Управление запросами, Ошибки, Веб-приложения, Разрешения, Конфигурация сервера.*

Annotation: *CORS (Cross-Origin Resource Sharing) is a mechanism for managing cross-origin requests between web applications. By default, web browsers block access to resources from different domains for security reasons. CORS allows these requests to be made. This article explores how CORS works, how to configure it, security concerns, and the advantages and disadvantages of using CORS for web applications. It also discusses common errors encountered when working with CORS and their solutions.*

Keywords: *CORS, Cross-origin requests, HTTP requests, Security, Cross-origin resources, Request management, Errors, Web applications, Permissions, Server configuration.*

Web ilovalarining rivojlanishi bilan birga, internetda resurslar almashish ham tobora kengayib bormoqda. Web brauzerlari tomonidan taqdim etilayotgan xavfsizlik choralari sababli, bir domenning boshqa domenlarga so'rovlar yuborishiga odatda ruxsat berilmaydi. Bu tamoyil "Same-Origin Policy" (SOP) deb ataladi. Biroq, web ilovalarining ko'plab hollarda turli domenlardan resurslarni olish zarurati tug'iladi. Bunday holatlarda, CORS (Cross-Origin Resource Sharing) mexanizmi yordamga keladi. CORS, domenlararo so'rovlarni amalga oshirishni

ta'minlash uchun ishlatiladi, lekin bu jarayon xavfsizlikka oid ba'zi masalalarni ham keltirib chiqaradi.

Ushbu maqolada CORS mexanizmining qanday ishlashi, qanday qilib uni sozlash mumkinligi, xavfsizlik jihatlari va web ilovalari uchun afzalliklari bilan birga, CORS bilan ishlashda uchraydigan muammolar va ularni hal qilish usullari ham ko'rib chiqiladi.

CORS (Cross-Origin Resource Sharing) – bu domenlararo resurslarni almashish imkonini beruvchi mexanizm bo'lib, asosan HTTP protokoli orqali ishlaydi. CORS asosan web-brauzerlarda qo'llaniladi, chunki ular "Same-Origin Policy" (SOP) siyosatiga amal qiladi. SOP siyosati web-brauzerga, bir domen tomonidan so'rov yuborilganda, faqat o'sha domenning resurslariga kirish imkonini beradi. CORS esa, boshqa domenlar bilan resurslarni almashishga ruxsat berish uchun maxsus HTTP sarlavhalarini (headers) ishlatadi.

CORSning ishlash prinsipi:

CORSning ishlash prinsipi quyidagicha:

Preflight Request (Oldindan tekshirish so'rovi): Agar so'rov mutlaqo yangi yoki xavfli bo'lsa (masalan, PUT yoki DELETE so'rovlari), brauzer avval preflight so'rovini yuboradi. Bu so'rovni server qabul qilib, kerakli ruxsatlarni berishi lozim.

Access-Control-Allow-Origin: Server so'rovni qabul qilganda, u Access-Control-Allow-Origin sarlavhasi bilan javob beradi, bu esa brauzerga boshqa domenlardan so'rovlarni qabul qilishga ruxsat berishni bildiradi. Access-Control-Allow-Methods va Access-Control-Allow-Headers: Agar server boshqa so'rovlar uchun ruxsat berishni xohlaydigan bo'lsa, u bu so'rovlar uchun mos metodlar (GET, POST, PUT, DELETE) va sarlavhalarni (headers) aniqlash uchun yuqoridagi sarlavhalarni qo'shishi mumkin.

Xavfsizlik:

CORSning xavfsizlik bilan bog'liq bo'lgan eng asosiy muammosi, serverlarning noto'g'ri sozlashlari natijasida nomaqbul domenlardan kelgan so'rovlarga ruxsat berishdir. Bu esa XSS (Cross-Site Scripting) va CSRF (Cross-Site Request Forgery) kabi hujumlarga olib kelishi mumkin. CORSni noto'g'ri sozlash, ayniqsa, ilova foydalanuvchilari va ularning ma'lumotlari uchun jiddiy xavf tug'diradi.

CORSni sozlash:

CORSni serverda sozlash uchun quyidagi usullarni ishlatish mumkin:

Nginx va Apache kabi web serverlarda CORSni sozlash.

Node.js va Express.js yordamida CORSni boshqarish uchun maxsus middleware (cors modulini o'rnatish) ishlatish.

Serverda ruxsat berilgan domenlarni aniqlash va faqat shu domenlarga so'rov yuborishga ruxsat berish.

Shuningdek, CORSda xavfsizlikni ta'minlash uchun faqat zarur domenlar va HTTP metodlarini ruxsat berish, foydalanuvchilardan kerakli autentifikatsiya va avtorizatsiya jarayonlarini talab qilish zarur.

CORS (Cross-Origin Resource Sharing) turli web ilovalarida va xizmatlarida ishlatiladigan muhim mexanizm bo'lib, quyidagi masalalarda keng qo'llaniladi:

1. API xizmatlari va mikroxizmatlar arxitekturasi

Web ilovalari ko'pincha turli domenlardan API (Application Programming Interface) xizmatlariga so'rov yuboradi. Masalan, bir web ilovasi bir domen (frontend) va backend server boshqa domen (API server)da joylashgan bo'lishi

mumkin. Bu holda, frontend va backend o'rtasida CORS mexanizmi orqali so'rovlar amalga oshiriladi. CORS yordamida, frontend o'z domenidan boshqa domenlarda joylashgan resurslarga so'rov yuborish imkoniyatini oladi.

Misol: Web ilovasi foydalanuvchi ma'lumotlarini olish uchun boshqa domendagi APIga so'rov yuboradi.

2. Web-sahifalar orasidagi resurslarni almashish

Bir web ilovasi yoki sahifa boshqa bir web ilovasidan yoki sahifasidan resurslarni olish zarurati tug'iladi. Masalan, bir domenning sahifasi boshqa domenlarda joylashgan tasvirlar, JavaScript kutubxonalarini yoki CSS fayllarini yuklashi mumkin. CORS bu jarayonda xavfsizlikni ta'minlab, ruxsat etilgan domenlardan resurslarni olishga imkon beradi.

Misol: Bir sahifa boshqa bir domenning tasvirini yoki stilini yuklab olishni istaydi.

3. Foydalanuvchi autentifikatsiyasi va avtorizatsiyasi

CORS web ilovalarida autentifikatsiya (foydalanuvchini tekshirish) va avtorizatsiya (foydalanuvchiga ruxsat berish) jarayonlarida ham muhim rol o'ynaydi. Ko'pincha, foydalanuvchi ma'lumotlari bir domenning serverida saqlanadi, lekin boshqa domenlarda joylashgan ilova orqali foydalanuvchi autentifikatsiyasi amalga oshiriladi. CORS yordamida boshqa domenlardan yuborilgan autentifikatsiya so'rovlari ruxsat etiladi.

Misol: Web ilovasida, foydalanuvchi boshqa domen orqali autentifikatsiya qilinadi va ma'lumotlar bilan ishlash uchun ruxsat oladi.

4. Client-Side Web Applications (SPA)

Single-Page Application (SPA) web ilovalari ko'pincha bir domenni ishlatib, serverga so'rovlar yuboradi, ammo serverlar turli domenlarda bo'lishi mumkin.

SPA ilovalarida, frontend va backend o'rtasida muvofiqlikni ta'minlash uchun CORSdan foydalanish zarur. Bu, frontend ilovasi va backend resurslar o'rtasida domenlararo so'rovlar amalga oshirilishini ta'minlaydi.

Misol: Angular, React yoki Vue.js bilan qurilgan SPA ilovasi backend APIdan ma'lumot olish uchun CORSni ishlatadi.

5. Mobil ilovalar va web xizmatlar

Mobil ilovalar (masalan, Android yoki iOS) ko'pincha serverlardan ma'lumot olish uchun web APIlarini ishlatadi. Agar mobil ilova va API serverlari turli domenlarda bo'lsa, CORS mexanizmi yordamida mobil ilovaga boshqa domenlardan so'rov yuborishga ruxsat berish mumkin.

Misol: Mobil ilova o'zining backend serveridan yoki uchinchi tomon APIlaridan ma'lumot olish uchun CORSni ishlatadi.

6. Ijtimoiy tarmoqlar va ulanishlar

Ijtimoiy tarmoqlarda (masalan, Facebook, Twitter) mavjud bo'lgan integratsiyalar (JavaScript SDK, iframe) boshqa domenlarda joylashgan resurslardan foydalanishni talab qiladi. CORS bu jarayonda xavfsizlikni ta'minlab, bir domenning boshqa domenlardagi resurslarga kirishini boshqaradi.

Misol: Web ilovasi ijtimoiy tarmoqdan foydalanuvchi profilini olish uchun so'rov yuboradi.

7. Third-Party JavaScript kutubxonalari va xizmatlari

Web ilovalari ko'pincha boshqa kompaniyalar tomonidan taqdim etilgan JavaScript kutubxonalari yoki xizmatlari bilan integratsiya qiladi. Masalan, Google Maps, Stripe, yoki PayPal kabi xizmatlar ko'pincha boshqa domenlardan olingan resurslarga ehtiyoj sezadi. CORS bu jarayonda boshqa domenlardan JavaScript kutubxonalarini yoki resurslarni ishlatishga imkon beradi.

Misol: Web ilovasi Google Maps API-dan xarita ko'rsatkichlarini yuklaydi va boshqa domenlardan xizmat oladi.

Autentifikatsiya qanday ishlaydi?

Foydalanuvchi tizimga kiradi: Foydalanuvchi o'zining foydalanuvchi nomi va parolini kiritadi.

Autentifikatsiya serverga yuboriladi: Frontend ilovasi foydalanuvchi ma'lumotlarini backend serveriga yuboradi.

Ma'lumotlarni tekshirish: Backend server foydalanuvchi nomi va parolini bazadagi ma'lumotlar bilan solishtiradi. Agar ular to'g'ri bo'lsa, autentifikatsiya muvaffaqiyatli bo'ladi.

Token yaratish: Autentifikatsiya muvaffaqiyatli bo'lsa, backend server foydalanuvchiga autentifikatsiya tokeni (masalan, JWT) yuboradi. Ushbu token keyinchalik so'rovlar bilan birga yuboriladi va foydalanuvchining autentifikatsiyasi tasdiqlanadi.

Avtorizatsiya qanday ishlaydi?

Foydalanuvchi tizimga kiradi: Foydalanuvchi avval autentifikatsiyadan o'tadi va token oladi.

Token yuboriladi: Frontend ilovasi so'rov yuborishda tokenni Authorization sarlavhasida yuboradi.

Backend server tokenni tekshiradi: Backend server JWT tokenni tekshiradi va foydalanuvchining kimligini aniqlaydi.

Ruxsatlar aniqlanadi: Backend server foydalanuvchining huquqlarini tekshiradi va unga kerakli resurslarga kirishga ruxsat beradi.

Misol: Agar foydalanuvchi admin bo'lsa, u faqat admin paneliga kirish huquqiga ega bo'ladi. Agar foydalanuvchi oddiy foydalanuvchi bo'lsa, admin paneliga kirish ruxsati bo'lmaydi.

Autentifikatsiya va avtorizatsiya CORS bilan qanday bog'lanadi?

CORS (Cross-Origin Resource Sharing) mexanizmi turli domenlar o'rtasida resurslarni olishga ruxsat berish uchun ishlatiladi. Autentifikatsiya va avtorizatsiya jarayonlarida CORS quyidagi rollarni bajaradi:

Foydalanuvchi autentifikatsiyasi: CORS yordamida frontend ilovasi va backend serveri turli domenlarda bo'lsa ham, frontenddan yuborilgan autentifikatsiya so'rovlarini qabul qilish mumkin bo'ladi. Masalan, frontend example.comda joylashgan bo'lsa, backend esa api.example.comda joylashgan bo'lsa, CORS serverga so'rov yuborishga imkon beradi.

Foydalanuvchi avtorizatsiyasi: Agar frontend JWT tokenni yuborishi kerak bo'lsa, Authorization sarlavhasini CORS yordamida server qabul qilishiga ruxsat berish kerak bo'ladi. Aks holda, frontend serverdan ma'lumot olishga muvaffaqiyatli erisha olmaydi.

Foydalanilgan Adabiyotlar

1.Microsoft Documentation. (2023). Cross-Origin Resource Sharing (CORS) in ASP.NET Core. Microsoft Docs

Ushbu hujjat ASP.NET Core ilovalarida CORS mexanizmini qanday sozlash va ishlatish haqida batafsil ma'lumot beradi.

2.JWT.io. (2023). Introduction to JSON Web Tokens. JWT.io

JSON Web Token (JWT) haqida umumiy tushunchalar va token yaratish va foydalanish usullari haqida tushuntirish.

3.M. O'Reilly, A. (2019). ASP.NET Core 3.1 - Building Web Applications. O'Reilly Media.

ASP.NET Core 3.1 bilan web ilovalarini qurish bo'yicha batafsil qo'llanma.

4.Hoffman, R. (2020). Web Security: A Guide for Developers. Packt Publishing.

Web xavfsizligi bo'yicha yozilgan qo'llanma, CORS, XSS, CSRF va boshqa xavfsizlik texnikalarini o'z ichiga oladi.

5.M. St. Laurent, A. (2018). Learning REST APIs with ASP.NET Core. Packt Publishing.

REST API yaratish va autentifikatsiya jarayonlarini ASP.NET Core yordamida amalga oshirish.

6.RFC 7235. (2014). Hypertext Transfer Protocol (HTTP/1.1): Authentication. IETF RFC

HTTP protokolida autentifikatsiya va avtorizatsiya uchun tavsiyalar, shu jumladan Authorization sarlavhasi bilan ishlash.

7.Ben, A. (2019). Mastering JWT Authentication in Modern Web Applications. Packt Publishing

Web ilovalarida JWT autentifikatsiyasini amalga oshirish va xavfsizlikni ta'minlash bo'yicha qo'llanma.