Web Uygulamalarında JWT Token, Authentication Authorization ve Redux

Web uygulamalarında authentication (kimlik doğrulama) ve authorization (yetkilendirme) genellikle JWT (JSON Web Token), cookies ve session gibi yöntemlerle gerçekleştirilir. Bir projede bu işlemleri backend tarafında genellikle bir authentication middleware veya servis kullanarak yönetilir.

Cookies ve session gibi yapılar, kullanıcının bilgilerini tutmak için kullanılır ve bu nedenle genellikle authentication (kimlik doğrulama) ve authorization (yetkilendirme) süreçlerinde önemli bir rol oynarlar.

- Cookies: Cookies, tarayıcıda saklanan küçük metin dosyalarıdır. Web sunucusu, tarayıcıya bir cookie göndererek, tarayıcı tarafından saklanacak bilgileri belirtir. Özellikle kullanıcının oturum durumunu ve bazı oturum verilerini saklamak için sıklıkla kullanılır. Authentication sürecinde, kullanıcı giriş yaptığında sunucu tarafından tarayıcıya gönderilen bir cookie, kullanıcının oturum açıkken kimlik doğrulama bilgilerini tutar. Bu cookie, tarayıcı tarafından sunucuya her istek gönderildiğinde sunucu tarafından kontrol edilir ve kullanıcının oturum durumunu belirler.
- Session: Session, sunucu tarafında tutulan geçici depolama alanıdır. Sunucu, her bir kullanıcı için bir oturum başlatır ve oturum boyunca kullanıcının etkileşimlerini takip eder. Kullanıcı bir oturum başlattığında, sunucu oturumla ilgili bir kimlik belirler ve bu kimliği kullanarak kullanıcıya ilişkilendirilmiş bilgileri (örneğin, oturum verileri, kullanıcı bilgileri) saklar. Authentication sürecinde, kullanıcı kimlik doğrulama işlemi başarıyla tamamlandığında, sunucu oturum içinde kullanıcının yetkilendirme bilgilerini saklar. Bu sayede, her istek gönderildiğinde sunucu, oturum kimliğini kullanarak kullanıcının yetkilendirme durumunu kontrol edebilir.

Bu yapılar authentication ve authorization süreçlerinde kullanılırken, çoğu modern web uygulaması JSON Web Token (JWT) gibi token tabanlı bir yaklaşımı tercih eder. Bu, sunucu tarafında oturum yönetimi ve oturum verisi saklama yükünü azaltır ve ölçeklenebilirlik açısından daha avantajlıdır. Ancak, cookies ve session gibi yapılar hala birçok durumda yaygın bir şekilde kullanılmaktadır, özellikle klasik web uygulamalarında ve bazı özel durumlarda.

bir kullanıcı giriş yaptıktan ve bir token oluşturulduktan sonra, sunucuda bir session oluşturulmamış olsa bile token içindeki bilgiler hala kullanılabilir.

JSON Web Token (JWT), kullanıcı hakkında belirli bilgileri (örneğin, kullanıcı kimliği, rolleri vb.) içeren bir veri yapısıdır. Bu bilgiler, kullanıcı oturum açıkken veya oturum kapalıyken sunucu tarafından üretilen ve kullanıcıya verilen bir token içinde kodlanır. Token, kullanıcının her bir isteğiyle birlikte sunucuya gönderilir. Sunucu, bu token'ı doğrular ve token içindeki bilgilere erişebilir.

Bu nedenle, kullanıcı giriş yaptıktan sonra bir oturum (session) açılmamış olsa bile, sunucu, gelen her istekle birlikte token'ı doğrulayarak kullanıcı hakkındaki bilgilere erişebilir. Bu şekilde, oturum yönetimine gerek kalmadan, sunucu tarafından token doğrulama işlemiyle kullanıcı kimliği ve yetkilendirme bilgileri elde edilir ve bu bilgilere dayanarak kullanıcının isteklerini işleyebilirsiniz. Bu token tabanlı yaklaşım, modern web uygulamalarında yaygın olarak kullanılır ve oturum (session) yönetimine göre daha ölçeklenebilir bir yapı sunar.

Redux

Redux, React uygulamalarında global bir state yönetim kütüphanesidir. React uygulamalarında, bileşenler arasında paylaşılan verilerin yönetilmesini kolaylaştırır. Authentication ve JWT token yönetimi gibi karmaşık işlemler için Redux sıklıkla kullanılır.

JWT token, kullanıcının kimlik doğrulamasını sağlayan bir mekanizmadır. Redux, React uygulamalarında JWT token'ın tutulması ve yönetilmesi için kullanılabilir. Redux, JWT token'ı uygulama boyunca kullanılabilir bir global state içinde saklar ve gerektiğinde bileşenlere sağlar.

Redux'un JWT token yönetimindeki görevleri :

1. Token Saklama: Redux, JWT token'ı saklayarak, kullanıcı oturumu boyunca token'ın erişilebilir olmasını sağlar. Bu, token'ın her istekte sunucuya gönderilmesi gerektiğinde kolaylık sağlar.
2. Token Güncelleme: Kullanıcı oturumu sırasında token'ın süresi dolabilir veya güncellenebilir. Redux, token'ın güncellenmesi durumunda bu değişikliği yönetir ve güncel token'ı saklar.
3. Token Temizleme: Kullanıcı oturumu sonlandığında veya token geçersiz hale geldiğinde, Redux token'ı temizleyerek oturumu sonlandırır ve kullanıcının oturumu kapatır.
4. Yetkilendirme Kontrolü: Redux, kullanıcı oturumu boyunca token'ı saklayarak, bileşenlerin yetkilendirme kontrollerini gerçekleştirmesine olanak tanır. Bileşenler, Redux'tan token'ı alarak yetkilendirme kontrollerini yapabilir ve kullanıcıya göre belirli işlevleri veya içeriği görüntüleyebilir veya gizleyebilir.

"Beni hatırla" özelliği genellikle cookie üzerinden yönetilir. Kullanıcı "Beni hatırla" seçeneğini işaretlediğinde, genellikle uzun ömürlü bir cookie oluşturulur. Bu cookie, kullanıcının tarayıcısında saklanır ve belirli bir süre boyunca geçerlidir.

Bu cookie genellikle kullanıcının oturum bilgilerini (örneğin, kullanıcı adı veya e-posta) ve bir oturum kimliği (session ID) gibi bilgileri içerir. Kullanıcı "Beni hatırla" seçeneği işaretlendiğinde, tarayıcı kapatılsa bile bu cookie hala saklanır. Bu sayede, kullanıcı sonraki ziyaretlerinde tekrar giriş yapmadan otomatik olarak oturum açılabilir.

Ancak, JWT token tabanlı authentication yöntemlerinde, genellikle "Beni hatırla" özelliği için token'ın süresini uzun tutarak benzer bir etki elde edilir. Kullanıcı "Beni hatırla" seçeneğini işaretlediğinde, token'ın süresi uzatılır veya token'ın süresi sınırsız yapılır. Bu sayede, kullanıcı oturumu kapatılsa bile tarayıcıyı tekrar açtığında otomatik olarak oturum açabilir.