Node.js ile Genişletme

- Node.js, HTTP sunucusu oluşturmak için kullanılabilir, ancak varsayılan çerçeve sınırlıdır. Geliştiriciler, web sunucusu oluşturma sürecinde yönlendirme, dosya ayrıştırma, kimlik doğrulama ve veritabanına bağlanma gibi işlevler için harici kütüphanelere ve paketlere başvururlar.
- Örneğin, hava durumu verilerini almak için bir HTTP isteği gönderilebilir ve alınan veriler manuel olarak ayrıştırılabilir.

XML Verilerinin İşlenmesi

- XML verilerini bir dize olarak işlemek, yapıyı göz ardı eder ve hatalı XML verileri içerebilir. Dize eşleştirme, XML verilerinin yapısına karşı daha az toleranslıdır.
- xml2js Node.js paketi, XML öğelerini JavaScript nesnesine dönüştürmek için kullanılabilir. Bu paket, başka bir dilde yazılmış bir XML ayrıştırma kütüphanesi gerektirmeden yalnızca JavaScript kullanır.

Paket Yönetimi

- Node.js paketlerini yönetmek için npm uygulaması kullanılır. Örneğin, npm install xml2js komutu ile xml2js modülü ve gerekli bağımlılıkları indirilir.
- parseString fonksiyonu, XML ağacını işledikten sonra geri çağırma fonksiyonunu çağırır ve sonuç JavaScript değişkeni, XML parçasının içeriğini temsil eder.

Node.js ve Web Çerçeveleri

- **Node.js**, JavaScript'i sunucuda çalıştıran bir çalışma ortamıdır; bir çerçeve değildir. Çerçeveler, belirli bir ortamda uygulama geliştirmek için kullanılan yapısal bir iskelet gibidir.
- Node.js ile çalışmak için bir web çerçevesine ihtiyaç vardır. Node.js ile çalışan çerçeveler "node web framework" veya "node framework" olarak adlandırılır.

MVC ve REST API Yaklaşımları

- **MVC (Model-View-Controller)**: Uygulamayı üç bileşene ayıran bir mimari modeldir:
    - **Model**: Uygulamanın verilerini yönetir ve veritabanı ile etkileşimde bulunur.
    - **View**: Modelden aldığı verileri sunar.
    - **Controller**: Kullanıcıdan gelen verileri işler ve model ile etkileşimde bulunur.
- **REST API**: Farklı web hizmetlerinin birbirleriyle iletişim kurmasını sağlar. RESTful API'ler, istemci ve sunucu kodlarının bağımsız olmasını ve iletişimin durumsuz olmasını gerektirir.

Popüler Node Web Çerçeveleri

- **Express.js**: Yönlendirme ve ara yazılım için kullanılan popüler bir Node web çerçevesidir. Kolay bir öğrenme eğrisi vardır ve MVC mimarisini uygulamak için uygundur.
- **Koa**: Express'in tasarımcıları tarafından oluşturulmuş daha yeni bir çerçevedir. Daha küçük ve daha ifade edici bir yapı sunar.
- **Socket.io**: Gerçek zamanlı veri alışverişi için idealdir ve WebSocket kullanarak istemci ve sunucu arasında iki yönlü iletişim sağlar.
- **Hapi.js**: Güvenlik özellikleri ile bilinen bir açık kaynak Node web çerçevesidir. API sunucuları ve HTTP-proxy uygulamaları geliştirmek için kullanılır.
- **NestJS**: Dinamik ve ölçeklenebilir kurumsal uygulamalar geliştirmek için uygundur ve TypeScript ile uyumludur.

Express Nedir?

- Express, Node.js çalışma zamanı ortamına dayanan bir web uygulama çerçevesidir. Düşük seviyeli detayları soyutlayarak uygulamanızı daha iyi organize etmenize ve daha hızlı geliştirmenize yardımcı olur.
- Express, API oluşturmak ve sunucu tarafı render'ı (SSR) için yaygın olarak kullanılır.

API ve Sunucu Tarafı Render'ı (SSR)

- API kullanımı, veritabanı katmanıyla etkileşim kurmak için bir HTTP arayüzü kurmayı içerir. Veriler, istemciye JSON formatında gönderilir.
- SSR'de, Express, istemciden gelen verileri kullanarak dinamik olarak HTML, CSS ve/veya JavaScript oluşturur. Bu işlem, res.render metodu ile gerçekleştirilir.

Express ile Çalışma Adımları 1. Express'i bir bağımlılık olarak tanımlayın (package.json dosyasında). 2. Gerekli modülleri indirmek için npm install komutunu çalıştırın. 3. Express modülünü içe aktarın ve bir Express uygulaması oluşturun. 4. Yeni bir rota işleyici oluşturun. 5. Belirli bir port numarasına HTTP sunucusunu başlatın.

package.json Dosyası

- package.json dosyası, bir Node.js modülünün içeriği hakkında bilgi depolar. İçinde modül adı, versiyonu, açıklaması, ana dosyası ve bağımlılıkları gibi bilgiler bulunur.
- Express'i bağımlılık olarak tanımlamak için, dependencies kısmında Express modülünü ve versiyonunu listeleyin.

Yönlendirme (Routing)

- Yönlendirme, sunucu tarafı betimlemesinde önemli bir unsurdur. Sunucuya gelen farklı yollar (routes) için istekler (GET, POST, PUT, DELETE) işlenmelidir.
- Uygulama düzeyinde veya yönlendirici düzeyinde yönlendirme yapılabilir. Uygulama düzeyinde, her yol için ayrı yöntemler kullanarak istekleri işlemek daha basittir.

Ara Yazılım (Middleware)

- Ara yazılım, istek ve yanıt nesnelerine ve bir sonraki işlevi çağırmaya erişimi olan fonksiyonlardır. Bir Express uygulaması birden fazla ara yazılım içerebilir ve bunlar birbirine zincirlenebilir.
- Ara yazılım türleri: uygulama düzeyi, yönlendirici düzeyi, hata işleme, yerleşik ve üçüncü taraf. Örneğin, uygulama düzeyindeki ara yazılım, kimlik doğrulama gibi işlemler için kullanılabilir.

Şablon Oluşturma (Template Rendering)

- Şablon oluşturma, sunucunun HTML şablonundaki dinamik içeriği doldurma yeteneğidir. Örneğin, React bileşenlerini sunucudan render etmek için express-react-views kullanılır.
- Şablon motoru, belirli bir dizindeki JSX dosyalarını arar ve dinamik içerik ile birlikte yanıtı oluşturur.

Kimlik Doğrulama (Authentication)

- Kimlik doğrulama, bir kişinin iddia ettiği kimliği doğrulamak için kimlik bilgilerini kontrol etme sürecidir. Bu, uygulamaların güvenliğini sağlamak için temel bir adımdır.
- Uygulamanın arka ucu, bu doğrulama sürecini yönetir.

Üç Popüler Kimlik Doğrulama Yöntemi 1. **Session-based Authentication**

- Kullanıcı, kimlik bilgileriyle giriş yapar ve sunucu, benzersiz bir şifrelenmiş **session ID** oluşturur. Bu ID, veritabanında ve tarayıcıda çerez olarak saklanır.
- Kullanıcı çıkış yaptığında veya belirli bir süre sonra, session ID yok edilir.

2. **Token-based Authentication**
    - Kullanıcı kimlik bilgilerini sağladığında, bir **token** alır. Bu token, kullanıcının kimliğini doğrulamak için kullanılır.
    - Token, genellikle **JSON Web Tokens (JWT)** formatındadır ve üç bölümden oluşur: **Header**, **Payload** ve **Signature**.
    - Token, her HTTP isteğiyle birlikte gönderilir ve bu sayede yetkilendirme sağlanır.
3. **Passwordless Authentication**
    - Geleneksel şifrelerin yerine, biyometrik veriler, e-posta ile gönderilen sihirli bağlantılar veya mobil cihazlara gönderilen tek kullanımlık kodlar gibi yöntemler kullanılır.
    - Bu yöntem, **public key** ve **private key** şifrelemesine dayanır. Kullanıcı kaydolduğunda, cihazı bir anahtar çifti oluşturur.

Kimlik Doğrulama Süreci

- Kimlik doğrulama, bir kullanıcının kimliğini doğrulamak için **credentials** (kimlik bilgileri) elde etme ve bu bilgileri kullanarak kimliği doğrulama sürecidir.

- Kullanıcıların tanımlanması ve erişim haklarının sağlanması amacıyla yapılır.

Token Tabanlı Kimlik Doğrulamanın Avantajları

- **Scalability**: Token'lar yalnızca istemci tarafında saklanır, bu da çok sayıda kullanıcıyı yönetmeyi kolaylaştırır.
- **Flexibility**: Token'lar birden fazla sunucu arasında kullanılabilir ve farklı web siteleri ve uygulamalar için kimlik doğrulama sağlar.
- **Security**: Kullanılan **JWT** (JSON Web Token) imzalanabilir ve şifrelenebilir, bu da onların değiştirilmesini ve okunmasını zorlaştırır.

API Sunucusu Kurulumu

- **POST API**: Kullanıcı adı ve şifre gönderilerek oturum açmak için kullanılır ve bir web token döner.
- **GET API**: Sadece kimliği doğrulanmış kullanıcıların erişebileceği çalışan bilgilerini almak için kullanılır.

Kod Örneği

- **express()** fonksiyonu ile bir web sunucusu modülü oluşturulur.
- GET API, **Authorization** başlığını okuyarak kullanıcıyı doğrular. Eğer başlık yoksa, 401 durum kodu ile "No Token" mesajı döner.
- JWT, **jsonwebtoken.verify()** fonksiyonu ile doğrulanır ve kullanıcı bilgileri kontrol edilir.