Node.Js ile RestFull API olusturma

JAFAR JAFARLI
20360859116
Bilgisayar Mühendisliği

Giriş

- **Node.js Nedir?**
- **Kullanım Alanları**
- **RESTful API Nedir?**
- **RESTful API'nin Avantajları**
- **RESTful API'nin Dezavantajları**
- **RESTful API Oluşturma**

Node.js Nedir?

- **Tanım ve Tarihçe**
  - Node.js, 2009 yılında Ryan Dahl tarafından geliştirilmiş, sunucu tarafında JavaScript çalıştırmamızı sağlayan bir çalışma ortamıdır.
  - V8 JavaScript motoru üzerinde çalışır ve Google Chrome tarayıcısı için geliştirilmiştir.
  - Node.js'nin asıl amacı, yüksek performanslı ve ölçeklenebilir ağ uygulamaları geliştirmeyi kolaylaştırmaktır.
- **Kullanım Alanları**
  - **Web Sunucuları**: Yüksek trafikli ve hızlı yanıt süreleri gerektiren web uygulamaları.
  - **API Sunucuları**: RESTful API'lerin oluşturulması ve yönetilmesi.
  - **Gerçek Zamanlı Uygulamalar**: Chat uygulamaları, online oyunlar ve gerçek zamanlı bildirim sistemleri.
  - **Microservices (Mikroservisler)**: Büyük uygulamaların daha küçük, yönetilebilir parçalara ayrılması.

RESTful API Nedir?

- **Tanım**
  - REST (Representational State Transfer), 2000 yılında Roy Fielding tarafından tanımlanmış bir mimari tarzdır. RESTful API, bu prensiplere uygun olarak tasarlanmış web servisleridir.
  - RESTful API'ler, HTTP protokolü kullanılarak istemci (client) ve sunucu (server) arasında veri alışverişi yapılmasını sağlar.

Temel Prensipler

- **Kaynak (Resource) Temelli**: RESTful API'lerde her şey bir kaynak olarak ele alınır ve bu kaynaklar URI (Uniform Resource Identifier) ile tanımlanır. Örneğin, bir kullanıcı kaynağı ***/users*** URI'si ile erişilebilir.
- **Stateless (Durumsuz) Olma**: Her isteğin bağımsız olması ve sunucunun önceki isteklerle ilgili herhangi bir bilgi tutmaması gerektiğini belirtir. Bu, her isteğin gerekli tüm bilgileri içermesi gerektiği anlamına gelir.
- **HTTP Metotları Kullanımı**: CRUD işlemleri (Create, Read, Update, Delete) için HTTP metotları kullanılır:
  - **GET**: Kaynakları okuma.
  - **POST**: Yeni bir kaynak oluşturma.
  - **PUT**: Mevcut bir kaynağı güncelleme.
  - **DELETE**: Mevcut bir kaynağı silme.
- **HATEOAS (Hypermedia as the Engine of Application State)**: Kaynakların, diğer kaynaklara nasıl erişileceğini belirten bağlantılar (hyperlink) içermesi gerektiği prensibi. Bu, istemcinin, API'yi keşfetmesini ve kullanmasını kolaylaştırır.

Avantajları

- **Esneklik ve Ölçeklenebilirlik**: RESTful API'ler, esnek yapıları sayesinde çeşitli istemciler (web tarayıcıları, mobil uygulamalar vb.) tarafından kullanılabilir ve ölçeklenebilir uygulamalar geliştirilmesine olanak tanır.
- **Bağımsızlık ve Modülerlik**: İstemci ve sunucu bağımsız olarak geliştirilebilir ve güncellenebilir. Bu, geliştirme sürecini hızlandırır ve bakımını kolaylaştırır.
- **HTTP Protokolünün Gücünden Yararlanma**: HTTP'nin yerleşik özellikleri (güvenlik, önbellekleme, yönlendirme vb.) RESTful API'lerde doğrudan kullanılabilir.

Dezavantajları

- **Durumsuz (Stateless) Yapının Karmaşıklığı**
  - RESTful API'ler, her isteğin bağımsız olması ve sunucunun herhangi bir durum bilgisi tutmaması gerektiği prensibine dayanır. Bu durum, istemcinin her istekte gerekli tüm bilgileri göndermesini gerektirir, bu da veri trafiğini artırabilir ve istemci tarafında karmaşıklık yaratabilir.
- **Versiyonlama Zorlukları**
  - RESTful API'lerin versiyonlaması, zamanla değişen ve güncellenen API'lerin yönetilmesi gerektiğinde zor olabilir. Versiyonlama stratejileri (URI bazlı, header bazlı, vs.) konusunda karar vermek ve uygulamak karmaşık olabilir.
- **Güvenlik Endişeleri**
  - RESTful API'ler, internet üzerinden erişilebilir olduklarından, çeşitli güvenlik tehditlerine maruz kalabilirler. API güvenliği sağlamak için ek önlemler (OAuth, API anahtarları, SSL/TLS, vb.) almak gereklidir.

RESTful API Oluşturma

- **Adım 1: Node.js ve npm Kurulumu**
- Node.js'nin en son sürümünü Node.js resmi web sitesinden indirip kurulur.

- **Adım 2: Proje Dizini Oluşturma**
- Yeni bir proje dizini oluşturulur ve

Adım 3: Proje Başlatma

- **npm init** komutunu kullanarak proje için bir **package.json** dosyası oluşturulur:

```
mkdir rest-api
cd rest-api
```

```
npm init -y
```

RESTful API Oluşturma

- **Adım 4: Gerekli Modüllerin Kurulumu**
- Express.js ve body-parser modülleri kurulur:
- Geliştirme sürecinde sunucuyu otomatik olarak yeniden başlatmak için nodemon kurulur:

```
npm install express body-parser
```

```
npm install --save-dev nodemon
```

RESTful API Oluşturma

Adım 5: Temel Sunucu Yapısının Oluşturulması

- Proje dizininde bir **index.js** dosyası oluşturulur ve temel sunucu yapılandırması yapılır:

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

app.get('/', (req, res) => {
  res.send('Merhaba, RESTful API!');
});

app.listen(PORT, () => {
  console.log(`Sunucu ${PORT} portunda çalışıyor.`);
});
```

RESTful API Oluşturma

Adım 6: package.json Dosyasını Düzenleme

•**package.json** dosyasını, nodemon kullanarak sunucu başlatacak şekilde düzenlenir. **"scripts"** bölümü yandaki gibi güncellenir:

```
"scripts": {
  "start": "node index.js",
  "dev": "nodemon index.js"
}
```

RESTful API Oluşturma

- **Adım 7: CRUD İşlemlerinin Eklenmesi**

- Kullanıcı veri modelini ve CRUD işlemlerini gerçekleştirmek için gerekli rotalar eklenmeli. Aşağıda örnek olarak kullanıcı yönetimi için temel CRUD işlemleri verilmiştir:

```
// UPDATE - Mevcut bir kullanıcıyı güncelleme
app.put('/users/:id', (req, res) => {
  const userId = parseInt(req.params.id);
  const updatedUser = req.body;
  let user = users.find(u => u.id === userId);
  if (user) {
    user.name = updatedUser.name;
    user.email = updatedUser.email;
    res.send(user);
  } else {
    res.status(404).send({ message: 'Kullanıcı bulunamadı' });
  }
});

// DELETE - Belirli bir kullanıcıyı silme
app.delete('/users/:id', (req, res) => {
  const userId = parseInt(req.params.id);
  users = users.filter(u => u.id !== userId);
  res.status(204).send();
});

app.listen(PORT, () => {
  console.log(`Sunucu ${PORT} portunda çalışıyor.`);
});
```

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

let users = []; // Geçici veri saklama

// CREATE - Yeni bir kullanıcı oluşturma
app.post('/users', (req, res) => {
  const user = req.body;
  user.id = users.length + 1; // Basit id atama
  users.push(user);
  res.status(201).send(user);
});

// READ - Tüm kullanıcıları getirme
app.get('/users', (req, res) => {
  res.send(users);
});

// READ - Belirli bir kullanıcıyı getirme
app.get('/users/:id', (req, res) => {
  const userId = parseInt(req.params.id);
  const user = users.find(u => u.id === userId);
  if (user) {
    res.send(user);
  } else {
    res.status(404).send({ message: 'Kullanıcı bulunamadı' });
  }
});
```

RESTful API Oluşturma

- **Adım 8: API'nin Test Edilmesi**
- Sunucuyu başlatın:

```
npm run dev
```

- API'nizi test etmek için Postman veya cURL gibi araçları kullanabilirsiniz. Örneğin, Postman ile **POST /users** isteği göndererek yeni bir kullanıcı ekleyin.

SORULARINIZ

Dinlediğiniz İçin Teşekkürler