



Java Spring Boot Bootcamp

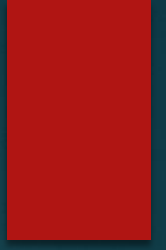
HAZIRLAYAN

MEHMET ALTAN

Spring Boot ile Rest API'lere Giriş

- ▶ API, "Application Programming Interface" (Uygulama Programlama Arayüzü) kelimelerinin kısaltmasıdır. Bir yazılım bileşeninin veya hizmetinin, diğer yazılım bileşenleriyle veya hizmetleriyle etkileşim kurmasını sağlayan bir set kurallar ve protokoller bütünüdür. API'lar, farklı yazılım sistemlerinin birbiriyle iletişim kurmasına ve veri alışverişi yapmasına olanak tanır. Bu sayede farklı uygulamalar birlikte çalışabilir, veri paylaşabilir veya birbirlerini tamamlayıcı işlevler sunabilir.

- ▶ Veri Alışverişi: API'lar, bir uygulamanın diğer bir uygulamanın verisini okumasına veya yazmasına olanak tanır. Örneğin, bir hava durumu uygulamasının dış hizmetlerden hava durumu verilerini çekmesi veya sosyal medya uygulamalarının kullanıcı verilerini yönetmesi gibi.
- ▶ İşlevsellik Paylaşımı: Uygulamalar, belirli işlevleri veya hizmetleri diğer uygulamaların kullanımına sunabilir. Örneğin, ödeme işlemleri sağlayıcısı, e-ticaret sitelerinin ödeme işlemlerini kendi sistemlerini kullanarak gerçekleştirmelerine olanak sağlar.



- ▶ Entegrasyon: Bir uygulama, başka bir uygulamanın özelliklerini veya içeriğini entegre edebilir. Örneğin, haritaların entegre edilmesi veya e-posta gönderme işlevselliğinin entegre edilmesi gibi.
- ▶ Hizmet Odaklı Mimariler: Büyük sistemler genellikle birden fazla hizmetin birleşimi olarak inşa edilir. Bu hizmetler birbirleriyle API'lar aracılığıyla iletişim kurar.

Sık kullanılan API Örnekleri

- ▶ Java I/O API: Giriş/Çıkış işlemleri için kullanılır. Dosya okuma/yazma, ağ işlemleri gibi I/O işlemlerini yönetir. `java.io` ve `java.nio` paketlerinde bulunur.
- ▶ Java JDBC API: Veritabanı işlemleri yapmak için kullanılır. JDBC (Java Database Connectivity), veritabanlarına bağlanma, sorgulama ve güncelleme işlemlerini kolaylaştırır. `java.sql` paketinde yer alır.
- ▶ Java Math API: Matematiksel işlemler yapmak için kullanılır. Matematik fonksiyonları, rastgele sayı üretimi gibi işlevleri içerir. `java.lang.Math` sınıfında bulunur.
- ▶ Java Reflection API: Çalışma zamanında sınıf ve nesne bilgilerine erişim sağlar. Sınıfın yöntemleri, alanları, arabirimleri gibi bilgileri incelemeye ve kullanmaya olanak tanır.

REST (Representational State Transfer)

- ▶ Web tabanlı servislerin mimarisini ve tasarım prensiplerini tanımlayan bir yaklaşımdır. REST, Roy Fielding'in 2000 yılında yayınladığı "Architectural Styles and the Design of Network-based Software Architectures" adlı tezinde tanımlanmıştır.

Ana prensipler;

- ▶ Kaynaklar (Resources): Her kaynak, benzersiz bir URI (Uniform Resource Identifier) ile temsil edilir. Bu kaynaklar genellikle veritabanı tabloları gibi somut varlıkları veya soyut kavramları temsil edebilir.
- ▶ HTTP Metodları: REST, HTTP protokolünün metodlarını (GET, POST, PUT, DELETE vb.) kullanarak kaynaklar üzerinde işlem yapar. Örneğin, GET metoduyla bir kaynağın okunması veya DELETE metoduyla bir kaynağın silinmesi gibi.
- ▶ Temsili Durum (Representational State): Veriler JSON veya XML gibi formatlarda temsil edilir. İstemci, kaynağın durumunu temsil eden verileri alır ve işler.
- ▶ HATEOAS (Hypermedia As The Engine Of Application State): İstemcinin bir kaynağa erişirken kaynağın durumunu (state) anlayabileceği bağlantılar (linkler) içermesi gerektiği prensibidir. Bu şekilde istemci, bir kaynağın üzerinde hangi eylemleri gerçekleştirebileceğini bilebilir.

Restful API

- ▶ RESTful API, iki bilgisayar sisteminin internet üzerinden güvenli bir şekilde bilgi alışverişi yapmak için kullandığı bir arabirimdir. Çoğu iş uygulaması, çeşitli görevleri gerçekleştirmek için diğer dahili ve üçüncü taraf uygulamalarla iletişim kurmak zorundadır.


- ▶ Bağımsızlık: İstemciler ve sunucular birbirinden bağımsızdır ve her biri değişiklikler yapabilir.
- ▶ İnteroperabilite: Farklı platformlardan ve dillerden gelen istemcilerin servisi kullanabilmesi.
- ▶ İsteğe Dayalı (Stateless): Sunucu, istemci durumunu (state) tutmaz. Her istek, kendisiyle ilgili tüm gerekli bilgileri içerir.
- ▶ Katmanlılık: Sunucu ve istemci arasında farklı katmanlar olabilir, bu sayede esneklik sağlanır.
- ▶ Doğal Olarak Güvenli (Naturally Secure): REST, HTTPS gibi güvenlik protokollerine kolayca entegre edilebilir. İstemci kaynaklara (veri tabanı vb.) doğrudan erişime sahip olmadığından daha güvenli bir yapı tasarlanabilir.
- ▶ Geniş Kullanılabilirlik: Web tarayıcıları gibi standart HTTP istemcileri kullanarak erişim kolaydır.


HTTP Temel Kavramlar

- ▶ Request (İstek): İstemci tarafından sunucuya gönderilen taleptir. Talep, metod, başlık (header) ve gerektiğinde bir gövde (body) içerebilir.
- ▶ Response (Yanıt): Sunucunun istemciye geri döndürdüğü cevaptır. Cevap, başlık (header) ve gerektiğinde bir gövde (body) içerebilir.
- ▶ Header (Başlık): İstek ve yanıtların meta verilerini içerir. Örneğin, "Content-Type" başlığı, gövde içeriğinin türünü belirtir.
- ▶ Body (Gövde): İstek veya yanıtın içeriğini taşır. Örneğin, JSON veya XML verisi içerebilir.
- ▶ Status Code (Durum Kodu): Sunucunun işlem sonucunu ifade eden üç haneli koddur. Örneğin, "200 OK" başarılı bir yanıtı, "404 Not Found" kaynağın bulunamadığını belirtir.


Temel HTTP Metodları

- ▶ Sunucudan belirli bir işlem talep etmek için kullanılır.
 - ▶ GET: Kaynağı okumak veya veri almak için kullanılır.
 - ▶ POST: Yeni bir kaynak oluşturmak veya veri göndermek için kullanılır.
 - ▶ PUT: Mevcut bir kaynağı güncellemek için kullanılır.
 - ▶ DELETE: Bir kaynağı silmek için kullanılır.
 - ▶ PATCH: Bir kaynağın belirli alanlarını güncellemek için kullanılır.

- 
- URI (Uniform Resource Identifier) tasarımı, web tabanlı servislerin ve kaynakların tanımlanması için kullanılan bir yöntemdir. Doğru ve etkili bir URI tasarımı, servisin erişilebilirliğini, anlaşılabilirliğini ve gelecekteki değişikliklere uyum sağlamasını kolaylaştırır.


- 
- ▶ Açıklık (Clarity): URI'ler, kaynağın neyi temsil ettiğini anlamak için açık ve anlaşılır olmalıdır. İnsanlar ve sistemler URI'leri okuduğunda kaynağın doğasını hızla anlayabilmelidir.
 - ▶ Düzenlilik (Consistency): URI'ler benzer kaynaklar için benzer olmalıdır. Aynı türdeki kaynaklar aynı modeli izlemelidir. Örneğin, `"/users/{id}"` ve `"/orders/{id}"` gibi.
 - ▶ Basitlik (Simplicity): URI'ler mümkün olduğunca basit ve sade olmalıdır. Fazla karmaşıklık, anlaşılabilirliği azaltabilir.


- ▶ Öngörülebilirlik (Predictability): URI yapısı, gelecekteki değişikliklere uyum sağlama yeteneğine sahip olmalıdır. Değişiklikler, URI yapısını zorlamadan yapılabilir olmalıdır.
- ▶ Mevcut Standartlara Uygunluk: URI'ler genellikle küçük harfle yazılır ve boşluk yerine tire veya alt çizgi kullanılır. Ancak, standart uygulamaları ve en iyi uygulama yönergelerini takip etmek önemlidir.
- ▶ Anlamlı Kategorizasyon: URI'ler, kaynakları mantıklı bir şekilde kategorize etmelidir. Örneğin, `"/products"` ve `"/categories"` gibi.

- 
- ▶ Sözdizimi ve Dil Kullanımı: Dil seçenekleri veya farklı biçimler (örneğin JSON veya XML) gibi faktörleri ele alırken dil ve biçim tanımlayıcılarını dahil etmek mümkündür.
 - ▶ Otomatik Tamamlama (Auto-Completion) ve İpucular: Kullanıcı dostu URI'ler, otomatik tamamlamayı ve hızlıca önerileri sunmayı destekler.
 - ▶ Versiyonlama: Eğer API'de farklı sürümler bulunacaksa, versiyonlama URI tasarımına dahil edilebilir. Örneğin, `"/v1/users"` ve `"/v2/users"` gibi.

URI Standartları

- ▶ HTTP metodları (GET, POST, PUT, DELETE, PATCH) API işlem türünü belirtir.
- ▶ Uç nokta isimleri (**resource_type**) genelde çoğul olarak kullanılır. (örneğin, books, authors, members, reservations).
- ▶ İç içe geçmiş veya ilişkili bir kaynağın (nested_resource_type) türünü belirtir (örneğin, bir üyenin rezervasyonları için reservations).
- ▶ Uç nokta adreslerimizde genelde yüklem değil isim kullanılır. Örn:
 - ▶ localhost/api/personnels/list
 - ▶ localhost/api/personnels/

- 
- ▶ GET /api/books: Tüm kitapları listeler.
 - ▶ GET /api/books/123: ID'si 123 olan bir kitabın detaylarını getirir.
 - ▶ POST /api/authors: Yeni bir yazar ekler.
 - ▶ PUT /api/members/456: ID'si 456 olan bir üyenin bilgilerini günceller.
 - ▶ GET /api/members/789/reservations: ID'si 789 olan bir üyenin rezervasyonlarını listeler.
 - ▶ DELETE /api/reservations/987: ID'si 987 olan bir rezervasyonu iptal eder.

- 
- Bu yapı, kaynakları ve işlemleri açık bir şekilde belirleyerek API'nizi tasarlamana yardımcı olur. Daha karmaşık senaryolarda veya daha fazla ilişkisel yapıda daha fazla düzeyli URI yapısı kullanabilirsiniz, ancak genel prensip aynı kalır: kaynakları tanımlamak ve ilişkilendirmek için açık ve tutarlı bir yapı kullanmak.