

REST API Tasarım Kılavuzu



Önemli REST API Pratikleri: API Versiyonu

- API versiyonu request URI'da net biçimde belirtilmelidir
 - GET /api/**v2**/owners

Önemli REST API Pratikleri: Hata Kodları

- Hataları istemci tarafına iletmek için HTTP statü kodları kullanılmalıdır
 - 4xx -> istemci tarafından kaynaklanan hatalar
 - 5xx -> sunucu tarafından kaynaklanan hatalar

Pratikleri: Hata Detay Bilgisi

- Hatalar ile ilgili istemci tarafına daha detaylı bilgi iletmek için statü kodlarının yanı sıra error payload'da döndürülmelidir
 - Error Title
 - Error Description
 - Internal Error Code
 - More Information Link

Önemli REST API Pratikleri: İsim vs Fiil

- URI ifadeleri isimlerden oluşmalıdır, isimler çoğul olmalıdır
 - GET /owners
 - GET /owners/101
 - POST /owners
- URI ifadelerinde fiil kullanılmamalıdır
 - GET /findOwners
 - POST /createOwner



Önemli REST API Pratikleri: GET

- GET metodu ile erişim Resource'lar üzerinde kesinlikle state değişikliğine neden olmamalıdır
 - GET /owners -> bütün owner kayıtlarını döndürür, paging uygulayabilir
 - GET /owners/101 -> sadece 101 id'li owner kaydını döndürür
 - Owner mevcut değil ise 404 not found statü kodunu döndürmelidir

Pratikleri: GET Sub Resource

- Sub-resource erişimleri parent resource ismi ile birlikte olmalıdır
 - GET /owners/101/pets
 - GET /owners/pets/202

Önemli REST API Pratikleri:

Filter, Sort, Paging, Projection

- GET metodu ile erişimlerde filter, sort, paging ve projection imkanları da sunulmalıdır
 - GET /owners?firstName=K&lastName=S
 - GET /owners?sort=+lastName,-firstName
 - + asc
 - - desc
 - GET /owners?offset=3&limit=50
 - **X-Total-Count** custom response header ile toplam kayıt sayısı da belirtilmelidir
 - GET /owners?fields=id,email

Önemli REST API Pratikleri: POST

- Resource üzerinde state değişikliği için aşağıdaki metotlar kullanılmalıdır
 - POST
 - POST /owners -> yeni bir owner yaratır
 - Başarılı işlem sonucu 201 Created statü kodu dönmelidir
 - Location response header'ında da yaratılan resource URL'i döndürülmelidir
 - POST /owners/101 -> işleme izin vermemeli ve 405 method not allowed hatası döndürmelidir

Önemli REST API Pratikleri: PUT

- PUT
 - PUT /owners -> bütün owner'ların bütün alanlarını günceller
 - PUT /owners/101 -> 101 id'li owner'ın bütün alanlarını günceller
 - Başarılı update sonrası 200 OK veya 204 No Content statü kodlarından herhangi birisi dönmelidir
 - 204 No Content daha çok tercih edilir

Pratikleri: PATCH ve DELETE

- PATCH
 - PATCH /owners -> bütün owner'ların spesifik alanlarını günceller
 - PATCH /owners/101 -> belirli bir owner'ın spesifik alanlarını günceller
- DELETE
 - DELETE /owners -> bütün owner'ları siler
 - DELETE /owners/101 -> belirli bir owner'ı siler
 - Başarılı silme işlemi sonucu 204 no content statü kodu dönmelidir
 - Belirtilen kayıt mevcut değil ise 404 not found statü kodu dönmelidir

Önemli REST API Pratikleri: HTTP Method Override

- Sadece GET ve POST metotlarını destekleyen istemciler için PUT ve DELETE metotlarını da çağırma kabiliyeti sağlanmalıdır
 - **X-HTTP-Method-Override** gibi bir custom request header kullanılabilir

Pratikleri: Serialization Format

- Serialization formatının istemci tarafından belirlenmesi sağlanmalıdır, bunun için **Accept** request header kullanılabilir
 - Accept text/xml, application/json
- Response **Content-Type** header'ı ile sunucudan istemciye resource'un hangi formatta döndüğü bilgisi belirtilmelidir
 - Content-Type application/json

Önemli REST API Pratikleri: Request/Response Model

- Document/Message oriented olmalıdır
 - Request message
 - Response message
- API, response üzerinden kendi kendine keşfedilebilir olmalıdır
 - Resource üzerinde yapılabilecek muhtemel aksiyonlar response ile belirtilmelidir
 - Data + control metadata (HATEOAS)

Pratikleri: Control Metadata

- Resource'un yanı sıra istemci tarafına control metadata'sı da dönmelidir
- Bunun için **HATEOAS** kullanılabilir
 - Dönülen resource üzerinde yapılabilecek işlemler belirtilebilir
 - Self, Create, Update, Delete
 - Paging uygulanan cevaplarda navigasyonel metadata bilgisi de sunulabilir
 - First, Next, Previous, Last

Pratikleri: HEAD ve OPTIONS

- HEAD metodu GET ile aynı olmalı, fakat resource'u dönmek yerine sadece response header'ları döndürmelidir
 - Resource'un durumunu, değişip değişmediğini öğrenmek isteyen istemciler için uygundur
- OPTIONS metodu resource URI üzerinde yapılabilecek işlemleri (HTTP methods) döndürmelidir

İletişim

- **Harezmi** Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- <http://www.java-egitimleri.com>
- info@java-egitimleri.com

