Rest Servislerde Versiyonlama

Bu yazımızda Rest Servislerde versiyonlama neden ve nasıl yapılır konusu üzerinde duracağız. Eğer uygulamanızı kullanan çok fazla müşteriniz varsa ve ihtiyaçlarımız giderek büyüyüp yeni özelliklere ve güncellemelere ihtiyaç duyuyorsa versiyonlama yapılabilir. Yeni bir versiyonlama yapılırken müşterilere yeni versiyona geçileceği önceden haber verilir ve daha sonra eski versiyon sistemden kaldırılıp yeni versiyon üzerinden devam etmeleri beklenir. Burada bir örnek vermem gerekirse yakın zaman önce HepsiBurada satıcı paneli sistemi v1 üzerinden yönetiliyordu. Sonra yeni bir versiyona geçeceklerini ve belli bir süre sonra v2 üzerinden devam edeceklerini ve tanınan süre bittikten sonra v1 kaldıracaklarını belirtiler.
Şu an sistemlerinde v2 kullandıklarını aşağıdaki iki örnek URI üzerinden de görebiliriz.

“ https://merchant.hepsiburada.com/v2/login?returnUrl=/ ” → Giriş Sayfası
“ https://merchant.hepsiburada.com/v2/dashboard ” → AnaSayfa

Uygulamalar versiyonlama sayesinde bütünü bozmadan yapılan güncellemelerle sistemlerinin geliştirmelere uygun hale getirilirler.
Versiyonlama yapmayı önlemek ve sürdürebilirlik sorunu yaşamamak için servislerimizin geriye uyumlu olacak (backwards compatibility) şekilde oluşturulması gerektiği söylenir ama bazı durumlarda versiyonlama ihtiyacı duyulabiliyor. Bu durumlar nelerdir kısaca onlara bakabiliriz.
Aşağıda verdiğim örnekte uygulamamıza ürün yüklemek için sahip olmamız gereken bilgiler arasında ürün id, ürün fiyatı, başlığı, açıklaması, kategori adı ve stok durumu bilgileri olsun.

[POST] http://url/products
{
 “productId”: 1234,
 “productPrice”: 450.00,
 “productTitle”: “new product title”
 “productDescription”: “new product description”
 “productCategoryName”: “electronics”
 “productStockCondition”: “some stock condition”
}

Yapılan yeni düzenleme ile ürün stok bilgilerini birbirlerinden ayrı ve daha düzenli almak için ürün stok id, stok miktarı ve stok gtin ayrı olarak alabiliriz.

[POST] http://url/products
{
 “productId”: 1234
 “productPrice”: 450.00,
 “productTitle”: “new product title “
 “productDescription”: “new product description “
 “productCategoryName”: “electronics “
 “productStockId”: “S12134”
 “productStockQuantity”: 25
 “productStockGtin”: 8692111111222
 “productStockConditionDetail”: “new stock condition details “
}

İki istek aynı hizmet üzerinden desteklenebilirdi ama her yapılan eklemede karışıklık artacağı için versiyonlama sistemi yapılması daha iyi bir yöntem olarak gözüküyor. Rest Servislerde versiyonlama yapılırken kullanılan farklı yöntemler vardır ve bunları yukarıdaki örnek üzerinden sırasıyla inceleyebiliriz.

URI ile Versiyonlama (URI Versioning)

En çok kullanılan versiyonlama sistemidir. İstemci kullanacağı versiyon bilgisini URI içerisinde gönderir.

Örnek Kullanım Şekli:

http://url/v1/products

http://url/v2/products

Avantajları:
- Versiyonlama çabukça fark edilir.
- Kullanımı basittir.
- İstemciler her defasında aynı verilere ulaşmaya çalıştığı sürece kaynakları önbelleğe almaya uygundur.

Dezavantajları:
- REST kurallarına tam olarak uymaz. Kaynakları görüntülemek için URI kullanılmalıdır.
- Versiyon değeri değiştiğinde tüm kaynaklar etkilenir.

Request Parametresi ile Versiyonlama (Request Parameter Versioning)

İstemcinin kullanmak istediği versiyon bilgisini parametre olarak belirtiyoruz.

Örnek Kullanım Şekli:

http://url/products?v=1

http://url/products?v=2

Avantajları:
- API versiyonu yapmak basittir, son sürüm değeri konulur.

Dezavantajları:
- Sorgu parametrelerine istek yapmak zor olabilir.

Header ile Versiyonlama (Headers Versioning)

İstemcinin kullanmak istediği versiyon bilgisini header ile göndermesini isteriz.

Örnek Kullanım Şekli:
http://url/products/products/header
headers=[PRODUCT-API-VERSION=1]

http://url/products/products/header
headers=[PRODUCT-API-VERSION=2]

Avantajları:

- Kullanımı kolay bir yöntemdir. Başlıkların düzgün bir şekilde gönderilmesi için dokümantasyon ihtiyacı olabilir.

Dezavantajları:

- URI veya Request Parametre gibi kolayca fark edilmez.
- Önbelleğe almak zor olabilir çünkü URL ile istek başlıklarını da dikkate almamız gerekir.

Media Type ile Versiyonlama (Media Type Versioning)

Content Negotiation veya Accept Header adıyla da anılır.

Örnek Kullanım Şekli:

http://url/products/produces
headers[Accept=application/vnd.my.app-v1+json]

http://url/products/produces
headers[Accept=application/vnd.my.app-v2+json]

Avantajları:

- URI yapısında versiyon bilgisi yoktur. Kaynak seviyesinde versiyonlama yapılır ve daha temiz ve kolay bir gösterim sağlar.

Dezavantajları:

- Karmaşık API yapısı vardır.
- Her kaynak için medya türünün bilinmesi gerekir.
- Browser üzerinden API görüntüleme olanağı yoktur, bazı Postman gibi araçlara ihtiyaç duyar.
- Test etmek zordur.

Kaynaklar
https://www.springboottutorial.com/spring-boot-versioning-for-rest-services
https://ademcatamak.medium.com/web-api-versiyonlama-1b446095b46b
https://www.narwhl.com/2015/03/the-ultimate-solution-to-versioning-rest-apis-content-negotiation/