REST SERVİS VERSİYONLAMA

REST Servis Versiyonlama Neden Yapılır?

Uygulamanın gelişmesi ile birlikte yeni isterlerin çıkması ve bu isterlerin eski isterler için hazırlanan servislerin karşılayamamasından kaynaklanmaktadır. Bunun için aşağıdaki örneği inceleyebiliriz;

```
public class StudentV1 {

    private String name;

    public StudentV1(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

```
public class StudentV2 {

    private String name;
    private String surname;

    public StudentV2(String name, String surname) {
        this.name = name;
        this.surname = surname;
    }

    public String getName() {
        return name;
    }

    public String getSurname() {
        return surname;
    }
}
```

1- URL Path Versiyonlama

URL üzerindeki path variable olarak tanımlanır. Burada versiyon farkı /v1, /v2 yapılarak belirlenir.

```
@RestController
@RequestMapping("/api")
public class UrlPathVersioningController {

    @GetMapping("/v1/students")
    public ResponseEntity<List<StudentV1>> getStudentsV1() {
        return ResponseEntity.ok(Arrays.asList(new StudentV1("Yusuf"), new
StudentV1("Metin")));
    }

    @GetMapping("/v2/students")
    public ResponseEntity<List<StudentV2>> getStudentsV2() {
        return ResponseEntity.ok(Arrays.asList(
                new StudentV2("Yusuf", "Alnıaçık"), new StudentV2("Metin",
"Alnıaçık")));
    }
```

```
}
```

Porstman de http://localhost:8080/api/v1/students denersiniz aşağıdaki çıktıyı alırsınız

```
 [
     {
         "name": "Yusuf"
     },
     {
         "name": "Metin"
     }
 ]
```

Porstman de http://localhost:8080/api/v2/students denersiniz aşağıdaki çıktıyı alırsınız.

```
[
      {
          "name": "Yusuf",
          "surname": "Alnıaçık"
      },
      {
          "name": "Metin",
          "surname": "Alnıaçık"
      }
  ]
```

2-URL Parametresi ile Versiyonlama

URL parametresi, QueryString parametresi olarak da bilinir. Bu yöntemde versiyon farklılığını URL üzerindeki **?version=1** ve **?version=2** yapılarak belirlenir.

```
@RestController
@RequestMapping("/api")
public class UrlParameterVersioningController {

    @GetMapping(value = "/students", params = "version=1")
    public ResponseEntity<List<StudentV1>> getStudentsV1() {
        return ResponseEntity.ok(Arrays.asList(new StudentV1("Yusuf"), new
StudentV1("Metin")));
    }

    @GetMapping(value = "/students", params = "version=2")
    public ResponseEntity<List<StudentV2>> getStudentsV2() {
        return ResponseEntity.ok(Arrays.asList(
                new StudentV2("Yusuf", "Alnıaçık"), new StudentV2("Metin",
"Alnıaçık")));
    }
}
```

Versyon 1 için; http://localhost:8080/api/students?version=1


```
[
    {
        "name": "Yusuf"
    },
    {
        "name": "Metin"
    }
]
```


Versiyon 2 için; http://localhost:8080/api/students?version=2


```
[
    {
        "name": "Yusuf",
        "surname": "Alnıaçık"
    },
    {
        "name": "Metin",
        "surname": "Alnıaçık"
    }
]
```


3-Başlık İçerisindeki Accept Alanını Kullanılarak Versiyonlama

```
@RestController
@RequestMapping("/api")
public class AcceptPartInHeaderController {

    @GetMapping(value = "/students", produces = "application/v1+json")
    public ResponseEntity<List<StudentV1>> getStudentsV1() {
        return ResponseEntity.ok(Arrays.asList(new StudentV1("Yusuf"), new
StudentV1("Metin")));
    }

    @GetMapping(value = "/students", produces = "application/v2+json")
    public ResponseEntity<List<StudentV2>> getStudentsV2() {
        return ResponseEntity.ok(Arrays.asList(
                new StudentV2("Yusuf", "Alnıaçık"), new StudentV2("Metin",
"Alnıaçık")));
    }
}
```

İki örnek incelendiğinde URL kısımlarında herhangi bir farklılık yoktur. Versiyon farklılığını Header altındaki Accept alanını kullanarak sağlarız.

4- Custom Header Kullanarak Versiyonlama

Header kısmına kendi belirlediğimiz key değeri ile de versiyonlama yapılabilir. Kendi oluşturduğumuz key lerin başına X harfi koyulması tavsiye edilir.

```
@RestController
@RequestMapping("/api")
public class CustomHeaderController {

    @GetMapping(value = "/students", headers = "X-API-VERSION=1")
    public ResponseEntity<List<StudentV1>> getStudentsV1() {
        return ResponseEntity.ok(Arrays.asList(new StudentV1("Yusuf"), new
StudentV1("Metin")));
    }

    @GetMapping(value = "/students", headers = "X-API-VERSION=2")
    public ResponseEntity<List<StudentV2>> getStudentsV2() {
        return ResponseEntity.ok(Arrays.asList(
                new StudentV2("Yusuf", "Alnıaçık"), new StudentV2("Metin",
"Alnıaçık")));
    }
}
```

Versiyon farklılığını Header altındaki X-API-VERSION alanını kullanarak sağlarız.

Kaynaklar;

https://metinalniacik.medium.com/rest-api-versiyonlama-9f1c89bfbdee#:~:text=%C4%B0lk%20olarak%2C%20REST%20API%20lar%C4%B1%20neden%20versiyonlanmas%C4%B1%20gerekti%C4%9Finden%20bahsedelim.,olu%C5%9Fturulan%20yap%C4%B1%20ile%20uyumlu%20olmamas%C4%B1d%C4%B1r.

https://koraypeker.com/2018/08/08/rest-mimarisi-ve-restful-servisler/

https://devnot.com/2016/rest-mimarisi-ve-restful-servisler/