Spring Boot ve HATEOAS

HATEOAS Nedir?

- Açılımı “Hypertext As The Engine Of Application State” şeklindedir
- Uygulamaların REST API’lerinin harici herhangi bir dokümana ihtiyaç olmadan, **kendi başına keşfedilebilir olması**nı hedefler
- REST çağrıları sonucu dönen resource **content içerisindeki link’ler** vasıtası ile diğer resource içeriklerine nasıl erişileceği hakkında bilgi sağlar

HATEOAS Nedir?

- Uygulama ve kullanıcılar bu **linkleri izleyerek** ilgili resource içeriklerine erişim imkanı kazanırlar
- HATEOAS yaklaşımındaki ana fikir istemcilerin, sunucu ile dönülen bu linkler vasıtası ile etkileşime geçmesi, bu etkileşimlerin gerekiyorsa “**state transition**”larla da sonuçlanmasıdır

Spring Boot ve HATEOAS

- RESTful API geliştirenler için Spring **HATEOAS'ın otomatik konfigürasyonu**nu yapar
- Aktive etmek için hateoas starter'ı **pom.xml**'e eklenmelidir

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-hateoas</artifactId>
</dependency>
```

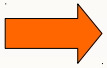
HATEOAS'a Uygun REST Servis Cevabı

```
@RestController
public class PetClinicRestController {
     @Autowired
     private PetClinicService petClinicService;
     @RequestMapping(method = RequestMethod.GET, value = "/owner/{id}")
     public ResponseEntity<?> getOwner(@PathVariable("id") Long id) {
          try {
               Owner owner = petClinicService.findOwner(id);

               ControllerLinkBuilder linkToController =
                    ControllerLinkBuilder.linkTo(PetClinicRestController.class);
               Link self = linkToController.slash("/owner/" + id).withSelfRel();
               Link create = linkToController.slash("/owner").withRel("create");
               Link update = linkToController.slash("/owner/" + id).withRel("update");
               Link delete = linkToController.slash("/owner/" + id).withRel("delete");

               Resource<Owner> resource = new Resource<Owner>(owner,
                                                  self,create,update,delete);
               return ResponseEntity.ok(resource);
          } catch (OwnerNotFoundException ex) {
               return ResponseEntity.notFound().build();
          }
     }
     ...
}
```

HATEOAS'a Uygun REST Servis Cevabı

- Spring content tipi olarak **application/hal+json** mime tipinde içerik dönmektedir

GET /owner/1



```
{
  "firstName": "Kenan",
  "lastName": "Sevindik",
  "_links": {
    "self": {"href": "http://localhost:8080/owner/1"},
    "create": {"href": "http://localhost:8080/owner"},
    "update": {"href": "http://localhost:8080/owner/1"},
    "delete": {"href": "http://localhost:8080/owner/1"}
  }
}
```

JSON ve HAL

- JSON notasyonunda normalde **hyperlink elemanları** ile ilgili herhangi bir **kabiliyet mevcut değil**dir
- **Hypertext Application Language** (HAL), JSON içeriğinde hyperlink elemanlarını tanımlayabilmek için bir takım ortak standartlar belirlemektedir

İletişim

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