Hibernate: *Java geliştiriciler için geliştirilmiş bir ORM (Object/Relational Mapping) kütüphanesidir.Nesne yönelimli modellere göre veritabanı ile olan ilişkiyi sağlayarak, veritabanı üzerinde yapılan işlemleri kolaylaştırmakla birlikte kurulan yapıyı da sağlamlaştırmaktadır.*

1) hibernate.cfg.xml ne işe yarıyor?
Database'e bağlantı nesnesini oluşturup, veri tabanına hakimiyet sağlıyoruz.

2) @Entity nedir?
Veritabanı tablosu oluşturacağız demek.

3) @Id ve @GeneratedValue nedir?
Id fieldı primary key olarak işaretler, generated value da auto-increment özelliği ekler.

4) SessionFactory nedir?
Hibernate ile veritabanını bağlama işlemlerini yönetmek için kullanılıyor.

4) Session nedir?
Hibernate framework'ü içerisindeki veri tabanı işlemlerini gerçekleştiren ana araçtır. SQL temelli sorguları oluşturmak, komutlandırmak ve çalıştırmak için kullanılır. Prepared statement yerine geçer.

5) Transaction nedir? (commit() - rollback())
Yapılan session olayını commit ile birlikte tamamlar, rollback ise yapılan session olayını geri al demektir.

6) HibernateUtil ne iş yapıyor?
hibernate.cfg.xml dosyasının içini açıyor, içindeki bilgilere göre SessionFactory nesnesini kuruyor, Session Factory üzerinden almış olduğumuz yetkiyle Sessionları hareket ettiriyoruz. Singleton design pattern kullanılıyor.

7) @Column nedir?
Database tablosundaki sütuna, fielda özellik vermemizi sağlıyordu.
@Column(length = 200)
Private String title —> normalde default değer 255ken, uzunluğu 200 yapıyoruz.
@Column(columnDefinition = "text") —> varchar yerine text türünde ayarlıyoruz.

8) @Table (name="User", schema ="jdbc.hibernate") nedir?
Veri tabanı içinde tablonun adı User olacak.

9) List<User> ls = sesi.createQuery("from User").getResultList(); ne işe yarar?
User Tablosundaki bütün verileri çeker. .setMaxResult() —> kaç tane sonuç getireceğini belirliyor

10) tr.commit(); ve sesi.close(); ne işe yarıyor?
Commit yaptığımız işlemi onaylıyor, Close sessioni kapatıyor.

11) foreign key nedir?
İki tablo joinlendiğinde bir tablonun primary keyin diğer tablonun foreign keyi olarak adlandırılır.

12) @OnetoOne nedir?
Tablolar arasında 1 e 1 bir ilişki kurar.

13) @OnetoMany nedir?
Tablolar arasında 1 e çok bir ilişki kurar. Bir ara tablo yaratılır.

14) @NamedQuery nedir?
İsteğimize göre o sınıfa ait sorgularımızı yazıyoruz ve her sorguya isim veriyoruz.

15) Procedure nedir?
Parametre alan SQL sorgularıdır, Search yaparken kullanılabilir. Java kısmında CALL methoduyla çağırılır. CREATE PROCEDURE ile bir prosedür oluşturularak BEGIN – END bloğu arasında istenen işlemler, kodlar yürütülebilir.

16) View nedir?
Tablolardaki verilerin bir veya birden fazla tablodan istenildiği şekilde bir arada sunulmasını sağlayan tanımlanmış sorgulardır. Aynı tablolar gibi satırlar ve sütunları içeren sanal tablolar olarak da tanımlanabilir. Tablolardan fakı ise tablolar verileri tutarak veri tabanında yer kaplarlar fakat view veri depolamaz, yer tutmazlar. Sadece SQL sorgusu veri tabanın veri sözlüğünde tutulmaktadır.

14) @NamedQuery nedir?
İsteğimize göre o sınıfa ait sorgularımızı yazıyoruz ve her sorguya isim veriyoruz.

Servlet: *Java EE içerisinde yer alan Servlet sunucu (server) – istemci (client) tabanlı uygulamaların haberleşmesini yönetmek için kullanılan sınıf, arayüz ve paket topluluğudur.*

1) @WebServlet nedir?
Sınıfı bir Servlet olarak işaretler, name ve value alır. Name sınıfın adıyla aynı olur baş harfi küçük, Value html kısmında formun actionıyla aynı isimde olmalıdır.

1) req.getParameter() nedir ?
Dataları yakalamak için request.getParameter() yaparız.

2) session.setAttribute() nedir?
Kullanıcı istek gönderdiğinde sunucuda (serverda) tutulan değer set edilir.

3) Session nedir?
Kullanıcıya ait bilgilerin sunucuda tutulduğu alan, oturum.

4) removeAttribute() nedir?
Session silmek için kullanılır.

5) invalidate() nedir?
Tüm sessionları silmek için kullanılır.

6) Cookie nedir?
Kullanıcıya ait bilgilerin kullanıcıda (istemcide) tutulduğu alan, çerez.

7) sendRedirect nedir?
Kullanıcıyı başka bir adrese yönlendirmek için kullanılır.
response.sendRedirect("http://www.google.com”);

8) RequestDispatcher nedir?
Gelen isteği başka bir Servlet’a aktarmak için RequestDispatcher kullanılır.
RequestDispatcher requestDispatcher = request.getRequestDispatcher(“/helloServlet");
requestDispatcher.forward(request, response);

***sendRedirect sayfayı yönlendirdiğinde adresi değiştirirken RequestDispatcher adresi değiştirmez.**

9) requestDispatcher.forward(request, response) ne işe yarar?
Yönlendirilen Servlet’a data gönderen kısım.

10) FilterServlet ne işe yarıyor?
Servletların utf-8e uygun olarak filtlerenmesini sağlıyor, setCharacterEncoding ile.
filterChain.doFilter(req,res) —> Çalışmazsa Chrome’a null gönderilir, beyaz bir sayfa çıkardı.

11) FilterServlet ne işe yarıyor?
Listernerlar kullanıcının ya da o an programdaki kişinin oluşturmuş olduğu Session varsa onu dinler ve arka planda alıp bunu kullanır.
contextInitialized —> Listenerlar ayakta mı?
contextDestroyed —> Listenerlar öldü mü?

12) Enumeration<String> enumeration = req.getHeaderNames(); ne işe yarıyor?
Bir sayfayı ziyaret ettiğinde o sayfaya gelen bilgiler ulaşıyordu. Hangi tarayıcıdan girdi? Hangi dili kullanıyor? Hangi ülkeden gibi bilgiler.

Spring (Kurulum)

Spring Boot Dev Tools: Uygulama geliştirirken her değişiklikte, değişikliğin çıktılara yansıması için kodun tekrar derlenmesi; yani bizim uygulamamızı tekrar çalıştırmamız gerekir. İş yükünü azaltmak adına burada Spring Boot Dev Tools devreye giriyor. Her kaydetme işleminde kodu tekrar derliyor ve çıktılarımızı

güncelliyor.

Lombok: Lombok ile daha temiz ve daha az kod yazmış oluyoruz. Bizim yerimize getter setter işlemlerini yapıyor.

Thymeleaf: A modern server-side Java template engine for web. Allows HTML to be correctly displayed in browsers and as static prototypes.

Spring Web: RESTful dahil, Spring MVC kullanarak web uygulamaları yazmamızı sağlıyor. Gömülü container olarak Apache Tomcat kullanıyor.

Spring Data JPA: Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Spring Data JDBC: Persist data in SQL stores with plain JDBC using Spring Data.

H2: Küçük (2mb) ayak izi ile JDBC API ve R2DBC erişimini destekleyen hızlı bir bellek içi veritabanı sağlar. Gömülü ve sunucu modlarının yanı sıra tarayıcı tabanlı bir konsol uygulamasını destekler.

Spring Data Elasticsearch (Access+Driver): Spring Data Elasticsearch ile, RESTful arama ve analiz motoru.

Validation: Bean Validation.

Spring: *Spring Framework Java için geliştirilmiş, açık kaynak olan bir uygulama geliştirme framework'üdür.*

1) @Controller nedir?
Sınıfın içindeki methodları birer servlet olarak işaretler.

2) @RequestMapping nedir?
Her GetMapping veya PostMappingin url'inde öne gelecek olan kelimeyi belirler. Her seferinde yazmak yerine Controller sınıfında bu notasyon ile bir kere yazılarak tekrardan kaçınırız.

3) Model Interface nedir?
Model data çekmez, data gönderir. Datayı view katmanına iletir. (Data taşımada sorumlu açılacak olan index.hml'in içerisine gönderilecek dataları Model ile göndeririz.

4)(“ “) veya (“/“) nedir?
Domaine ilk defa giriş yapıldığında ilk istek boş gider, Mapping'in altındakini yap demektir.

5) Dependencies Injection nedir?
Spring verileri kendi çekip entity'ye göre nesneyi bizim için oluşturuyor. Üretmek istediğimiz classın içinde boş constructor olacak. Random, Math bunlara örnek olabilir. Scanner değil.

6) saveAndFlush(user) nedir?
Userin içerisinde bir id değeri varsa, veritabanı ile senkronize çalışır ve updatelenir ama yoksa ekleme işlemi yapar.

7) @PathVariable nedir?
url'de bulunan değişkeleri ilgili metodlara aktararak ilgili methodun işlemi yapmasını sağlar.

8) redirect:/ ne işe yarar?
Ana domaini açar. Neden? —> Kayıtları yenilemek, güncel kayıtları göstermek ya da model içindeki dataları refreshlemek için kullandığımız bir yöntem. (HTML tarafında, arka tarafta veri kalıyor ama model olan mappinge gönderirsek boş kayıt kalmıyor.)

9) userUpdate = uRepo.findById(uid).get(); Neden get() methodu var?
Java8 ile birlikte gelen özellik, findById Optional türde bir nesne gelir fakat bizim beklentimiz saf değere ulaşmak.

10) Constructor Injection nedir?
Sınıf için gereken bağımlılıklar yapıcıya bağımsız değişkenler olarak sağlanır. Spring 4.3'ten önce, constructor'a bir @Autowired ek açıklaması eklememiz gerekiyordu.

```
final CustomerRepository cRepo;
public SaleController(CustomerRepository cRepo) {
    this.cRepo = cRepo;
}
```

11) UserRepository extends JpaRepository<User,Integer>{}
User, entity. Integer da onun primary keyinin türü. Extend edilen JpaRepositorynin içinde bulunan özellikler: veritabanına giriş, veritabanından veri alış, saveAndFlush, saveAllAndFlush, delete, deletebatch gibi methodlar var ve bunlar generic type <T> bu interfacein içindeki gövdesiz methodların nasıl çalışması gerektiğine karar verebilen bir yönteme sahip.

12) Thymeleaf nedir?
Hem web hem de web dışı ortamlarda çalışabilen bir Java XML / XHTML / HTML5 şablon motorudur. MVC tabanlı web uygulamalarının görünüm katmanında XHTML / HTML5 sunmak için daha uygundur, ancak çevrimdışı ortamlarda bile herhangi bir XML dosyasını işleyebilir. Tam Spring Framework entegrasyonu sağlar.

13) Neden th:block kullanıyoruz?
Tasarımın bozulmaması için.

14) Pageable nedir?
Veritabanında sorgu atıldığı zaman dönen sonuç eğer çok fazla ise kayıtların hepsinin hafızaya çekilmesi ve işlenmesi hem zaman hem de hafıza açısından sorun olacaktır. Bu sorunu aşmak için toplam kayıtlar parça parça çekilip işlenir. Her bir

parçaya page deniyor. Spring kendi içinde bu yapıyı desteklemek için bir Interface/ sınıf kümesi sunuyor. Bu yapı atılan sorgunun sonuçlarını parçalara/sayfalara (page) ayırmak için kullanılıyor.
Pageable pageable = PageRequest.of(0, 10);
Page<Employee> page = employeeRepository.findAll(pageable); —> ilk sayfadan (0) 10 tane çalışan listelemek istedik.

14) @Transactional nedir?
Aynı anda hem delete hem select işlemi varsa Transactional nesnesi yoksa yeni bir tane yaratıyor ve yeni transaction üzerinden yeni bir yol kurmuş oluyoruz ve böyle datamız da ilgili işlemi yapmış oluyor.

15) Projection nedir?
Yazma düzenleme silme tek satırsa JPA query çok iyi ama ağır select sorgularına ihtiyaç varsa JPA maliyeti artırabiliyor maliyetin daha minimal seviyeye ulaşabilmesi için projection yöntemiyle birlikte veritabanına native queryle birlikte gitmek gerekiyor.

16) hibernate.dll-auto (application.properties'de) ne işe yarar?
Entity tablolarımızın veritabanına gidişi için gerekli olan akisyon.

17) Auditing Listenerlar:
@PrePersist —> Select işleminden önce çalışır.
@PreUpdate —> Update işleminden önce çalışır.
@PreRemove —> Delete işleminden önce çalışır.

@PostPersist —> Select işleminden sonra çalışır.
@PostUpdate —> Update işleminden sonra çalışır.
@PostRemove —> Delete işleminden sonra çalışır.

@EntityListeners(AuditingEntityListener.class)
@CreatedBy —> Oluşturan.
@CreatedDate —> Oluşturulma tarihi.
@LastModifiedBy —> Son modifiye eden.
@LastModifiedDate —> Son modifiye edilme tarihi.

17) Log4j nedir?
Arka planda hataların yönetimi için verilen kütüphane. Logging kütüphanesini kullanıyor, log4j aynı yapıya hizmet edebilme yeteceğine sahip olan bir özellik. Log4j'nin manipülasyonu yapılarak burda çok daha hızlı ve çok daha iyi bir şekilde, yani Logging'den daha yetenekli olacak şekilde bir hizmetle arka planda otomatik olarak hata olduğunda dosyanın içine yazan, yeni dosyalar üreten çok gelişmiş bir araç.

18) Authentication Manager nedir?
Springde oturumları kontrol eden oturumları denetleyen ve bizim daha önceki kurallarımıza bakarak o oturumların o hizmeti almaya uygun olup olmadığını denetleyen bir mekanizma.

19) Secuirty Context nedir?
Giriş yapan kullanıcının rolünü, kullanıcı adını, ne zaman giriş yaptığını ve buna yönelik bilgileri kendisinde barındıran bir yapı. Session gibi ama session'a datayı kendimiz yazıyorduk session'i biz oluşturmuyoruz session!in butun yönetimini spring yapacak.

20) AccessDecision Manager nedir?
Giriş yapan kullanıcının o sayfaya girmeye yetkisi var mı? Rolü uygunsa sayfaya girsin yoksa girmesin. Spring karar veriyor.

21) Using generated security password nedir?
Domain ilk açıldığında —> Console'da : Using generated security password: 4n4km3n5-t4rc-e-43-r3 gibi bir şey bize default halinde bir şifre verir ve user verilen bu şifreyle giriş giriş yapabilir. Config yapısı —> açılacak olan formun bizim form olmasını istiyoruz. WebSecurityConfig extends WebSecurityConfigurerAdapter : bu sınıfın içindekiler geçerli ve login bu şekilde çalışıyor fakat biz bu sınıfın içindekilerini override ediyoruz ve bizim dediklerimiz artık geçerli oluyor. Bu bir abstract class.İçindekiler: WebSecurityConfigurerAdapter, AuthenticationManagerBuilder, HttpSecurity.

22) AuthenticationManagerBuilder ne işe yarar?
SQL içinde sorgulama yaparak kullanıcının varlığını ve rolünü denetler.

23) HttpSecurity ne işe yarar?
Rollere göre kullanıcı hangi sayfaya giriş yapacak ise ilgili denetimi yapar.

24) @Configuration nedir?
Bu notasyon olmadan program arka planda bu sınıfın içindeki elemanları dikkate almıyor.

25) permitAll() ne işe yarar?
Herkese izin verir.

26) hasAnyRole("…") ne işe yarar?
İzin verilen roller girebilsin.

27) hasRole(".") ne işe yarar?
İzin verilen rol girebilsin.

28) http.csrf.disable(); hangi sorunu çözer?
Security arka planda sürekli aktif bu nedenle, dışarıdan veri alırken kontrol sağlanıyor, ajaxtan giderken direkt program hata veriyor. Verileri url'den alıyor ama data olarak almıyor.

29) .headers().frameOptions().sameOrigin().and() ne işe yarıyor?
Gömülü olan iFrame'in açılmasını sağlıyor.

30) @Valid nedir?

Validation eklememizi sağlayan anotasyon. API'de validasyon işlemi için parametrede @Valid yazmak zorunlu değildir. Rest API'nin kendi hata mesajı döner, method içinde bizim belirttiğimiz mesajlar dönmez.Kendi hata mesajlarımızın dönemesini istediğimizden API'de parametre içine @Valid yazarız.

31) @ModelAttribute("…") nedir?
Web sayfamız ile backend arasında veriler arasında iletişimi sağlamaktadır. Controller'dan HTML'e veri göndermek istediğimiz model.AddAttribute("name", object) şeklinde; HTML'den Controller'a veri göndermek istediğimizde de @ModelAttribute("name") Object object şeklinde yazıyoruz.

32) BindingResult nedir?
BindingResult, bir doğrulama ve bağlamanın sonucunu tutar ve gerçekleşmiş olabilecek hataları içerir. BindingResult, doğrulanan model nesnesinden hemen sonra gelmelidir, aksi takdirde Spring nesneyi doğrulayamaz ve bir exception atar.

33) @ResponseBody nedir?
Bu anotasyon kullanıldığında uygun formata dönüştürme işlemi yapılır. Ajaxtan JSON ile veri yolluyoruz.

34) @RequestBody nedir?
Bu anotasyon kullanıldığında uygun formata dönüştürme işlemi yapılır. Ajaxtan JSON ile veri alıyoruz.

CACHE: *Birkaç anotasyon yardımıyla rahat yönetilebilir önbellekleme yapmamızı sağlıyor.*

1) @EnableCaching nedir?
Spring Cache'i aktifleştirebilmek için bir @Configuration sınıfına bu anotasyonu eklememiz gerekiyor.

2) @Cachable nedir?
Üzerine yazdığımız methoddan dönen şey cachelenecek demek.

3) Cache nasıl sıfırlanır?
Refresh methoduyla.

REST API: *REST API, istemci ve sunucu arasında XML, JSON, HTML, TEXT başta olmak üzere pek çok formatta veri taşıyarak uygulamaların haberleşmesini sağlar. REST standartlarına uygun yazılan web servislerine RESTful servisler denir.*

(REpresentational State Transfer), web uygulamalarında temsili state transferi için kullandığımız bir mimaridir. Örneğin bir alışveriş sitesinde bir ürünün ilgili sayfasına gittiğimizde bir REST isteği ile resource'un state'ini istiyoruz ve bize cevap olarak bir JSON dosyası geliyor. (JSON olması zorunlu değil, XML veya

başka bir şey de olabilir.)

1) @RestController nedir?
Controller sınıflarının oluşturulmasında kullanılır. @Controller geriye bir web sayfası döndürürken, @RestController geriye data döndürür. Herhangi bir template engine ihtiyacı olmaz, @Controller ise mutlaka bir template engine sahip olmalı veya ResponseBody ile String döndürmesi sağlanmalıdır.

2) @Service nedir?
Bu sınıfın altında methodlarımızı yazıp, RestController sınıfında methodlarımızı çağırıyoruz.

3) RestController sınıfında neden ResponseBody yazmıyoruz?
RestController'ın içerisinde zaten ResponseBody olduğundan.

SWAGGER: *RestAPI'nin dokümantasyonunu sağlar. Swagger bir çok dili desteklemektedir. Spring Boot ile kullanmamız için ise SpringFox çıkarılmıştır. Swagger'ın önemli bir amacı RestAPI için bir arayüz sağlamaktır. Swagger ile gelen bir başka kullanım kolaylığı ise artık biz projelerimizde API bilgilerimize Postman veya SoapUI üzerinden bakma mecburiyetimiz ortadan kalkmıştır. Bu işlemleri swagger üzerinden de gerçekleştirebilmekteyiz.*

1) @ApiModel nedir?
Entity classına yazılır. ApiModel ile oluşturduğumuz API modelleri hakkında bilgi veririz. Böylece geliştiriciler hangi modelin ne için yazıldığını dokümantasyon üzerinden direkt anlayabilirler.

2) @ApiModelProperty nedir?
ApiModelProperty ise model sınıfında eklediğimiz değerlerimizin üzerinde açıklama yapmak için kullanırız.

3) @Api nedir?
@Api ile sınıfımızın bir API dokümantasyonu olduğunu belirtiyoruz. Controller sınıfına yazıyoruz.

4) @ApiOperation nedir?
@ApiOperation ile metotlarımızın hangi işlemi yaptığını belirtiyoruz. Mapping methodlarımızın üzerine yazıyoruz.
@ApiOperation(value = "New User adding method") —> Swagger'da mappinglerin açıklaması olarak value geliyor.

5) SwaggerConfig sınıfı ne işe yarar?
Swagger konfigürasyonlarını yazdığımız sınıf. @Configuration ve @EnableSwagger2 anotasyonlarını kullanıyoruz, bu şekide doküman türünü swagger olarak belirliyoruz.

6) ApiInfo methodu ne işe yarar?

ApiInfo metodumuz ile projemize ait doküman bilgilerimizi veriyoruz. Projemizin başlığını, açıklamalarını, versiyon bilgileri vs. belirttiğimiz bir alandır.

```
private ApiInfo apiEndPointsInfo() {
return new ApiInfoBuilder().title(“Spring Boot Swagger Örnekler”)
.description(“Api Dokümantasyonu”)
.contact(new Contact(“Selen Kösoğlu”, “” ,””))
.licence(“Apache 2.0”)
.licenceUrl(“http://www.apache.org/licenses/LICENSE-2.0.html”)
.version(“1.12.3”)
.build();
}
```

REDIS: *Bir Cache yönetim programıdır. Normal bir veri tabanı değil, NoSQL özelliğine sahip bir veritabanıdır. Veri iletişimini sağlar, çok büyük verileri atmamamız gerekiyor sadece sonradan cachelemek istediğimiz durumlarda Redis kullanıyoruz. Redis'in içindeki veritabanı yapısı ilişkisel veritabanı yapısına uygun değildir. Redis'i sunucuda kullanmak istediğimizde dockerize ediyoruz. Senkron çalışır. (NoSQL'deki asıl amaç veriyi hızlı bir şekilde almaktır.)*

1) @RedisHash nedir?

Değerlendirmek istediğimiz Property Class'ının üzerine yazıyoruz.
@RedisHash(“Session”) —> Session tablosu.

2) @EnableRedisRepositories ne işe yarar?

JPA'nın Redis'i yönetebilmesi için izin veriyor. Repository sınıfının üzerine yazılır.

3) @MappedSuperclass nedir?

Entitylerde extend alacaksak koymak zorundayız.

ELASTICSEARCH: *Büyük veri yığınları içerisinde tam metin (full-text) aramada ve bu büyük veri yığınlarının analiz edilmesinde kullanılan Apache Lucene tabanlı, Java ile yazılmış açık kaynak kodlu bir arama ve analitik motorudur. Indexler üzerinden arama yaparak çok hızlı bir şekilde sonuç üretebilmektedir. Dolayısıyla big data kavramının geçtiği yerlerde adından sıkça söz ettirmektedir.*

1) ElasticsearchConfig sınıfı ne işe yarar?

Elasticsearch instance'ına nasıl bağlanılacağını tanımlamak için Elasticsearch API tarafından sağlanan RestHighLevelClient'ı kullanıyoruz.

2) @EnableElasticsearchRepositories ne işe yarar?

Oluşturulacak olan repository'lerin istenilen işlevleri sağlayabilmesi için uygulamanın ayağa kalktığı sınıf @EnableElasticsearchRepositories

anotasyonu ile işaretlenip parametre olarak repository'nin yer aldığı paketin yolu verilir. Böylelikle Spring tarafından gerekli konfigürasyonlar yapılacaktır.
@EnableElasticsearchRepositories(basePackages = "com.kodgemisi.elasticsearchdemo.dao")

3) @Document ne işe yarar?

Elasticsearch'teki her bir kayıt doküman olarak geçmektedir. Bunu belirtmek için model, @Document anotasyonu ile işaretlenip kaydedileceği index'i belirtmek için ise indexName parametresi veriliyor.
@Document(indexName = "erp")

4) @Field(type = FieldType.) ne işe yarar?

FieldType.Text, FieldType.Integer olabilir, alanın türünü belirler.