

JPA/Hibernate ile Veri Erişimi



JPA/Hibernate ile Çalışmak

- Spring Boot JPA desteği **Spring Data** projesi üzerine kuruludur
- JPA ile çalışabilmek için **data-jpa starter**'ına ihtiyaç vardır
- JPA vendor olarak **Hibernate** kullanılır

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
</dependencies>
```

JPA/Hibernate ile Çalışmak

- **EntityManagerFactory** bean tanımı otomatik olarak yapılır
- Spring Boot **doğrudan Hibernate** üzerinden çalışmayı **desteklemediği** için **SessionFactory** bean tanımı **mevcut değildir**

- JPA/Hibernate property'leri **application.properties** içerisinde uygulamaya göre özelleştirilebilir

```
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect  
spring.jpa.properties.hibernate.show_sql=true  
spring.jpa.properties.hibernate.format_sql=true  
spring.jpa.properties.hibernate.use_sql_comments=true
```


JPA Auto Scan Kabiliyeti

- **Entity sınıflarını** tespit etmek için **@EnableAutoConfiguration** veya **@SpringBootApplication** anotasyonuna sahip sınıfın bulunduğu **paket ve alt paketler** taranır
- Benzer biçimde Spring Data **Repository** veya **CrudRepository** arayüzlerinden türeyen uygulamaya özel arayüzler de otomatik olarak tespit edilmektedir

@EntityScan

- **@EntityScan** anotasyonu ile JPA entity sınıflarının aranacağı base paketler değiştirilebilir

```
@EntityScan(basePackages= {"com.module1", "com.module2"})
@Configuration
public class AppConfig {
    ...
}
```



Eğer anotasyon basePackage verilmeden kullanılırsa, kullanıldığı sınıfının bulunduğu paket basePackage olarak kabul edilir

JPA ile Otomatik Şema Yönetimi

- Sadece **gömülü veritabanı** kullanımı durumunda JPA ile veritabanı otomatik olarak yaratılacaktır
- **Connection tipi** H2, HSQLDB, DERBY ise gömülü veritabanı olarak kabul edilir
- Gömülü olmayan veritabanı kullanımında da bu özelliği aktive etmek için

```
spring.jpa.hibernate.ddl-auto=create-drop
```

- Bu tanım aşağıdaki tanımı her zaman ezer

```
spring.jpa.generate-ddl=true
```

Open EntityManager in View

- Spring Boot JPA/Hibernate ile çalışırken lazy hatalarının önüne geçmek için default olarak **OpenEntityManagerInViewInterceptor** tanımlar
- İstenirse **devre dışı** bırakılabilir

```
spring.jpa.open-in-view=false
```

- Hibernate 5.x için **tercih edilen yöntem enable lazy load no trans** özelliğidir

```
spring.jpa.properties.hibernate.enable_lazy_load_no_trans=true
```


Hibernate Second Level Cache

- Hibernate'in ikincil önbellek kabiliyetini devreye almak için **fiziksel önbellek sağlayıcısı pom.xml**'de bağımlılık olarak eklenmelidir

```
<dependency>  
    <groupId>org.hibernate</groupId>  
    <artifactId>hibernate-ehcache</artifactId>  
</dependency>
```

Hibernate Second Level Cache

- Ayrıca **application.properties** içerisinde ikincil önbelleği devreye alan **property tanımları** da yapılmalıdır

application.properties



```
spring.jpa.properties.hibernate.cache.use_second_level_cache=true  
spring.jpa.properties.hibernate.cache.use_query_cache=true  
spring.jpa.properties.hibernate.cache.region.factory_class=  
org.hibernate.cache.ehcache.SingletonEhCacheRegionFactory
```

İletişim

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

