**21.03.2023**

İlgili günde Hibernate ile mevcut bir veritabanı üzerinde mapping(haritalama) işleminin nasıl yapıldığı hakkında araştırma ve çalışma yapılmıştır. Konunun anlaşılması için küçük ölçekte örnek bir projeye başlanmıştır.

Mapping(haritalama) işlemi, mevcut bir veritabanında bulunan tablo ile Entity katmanındaki nesne yapısını eşleştirilmesi için yapılır. Bu eşleştirme nesnenin tipi ve ismi açısından gerçekleştirilir. Bu işlem sayesinde ne backend ne de veritabanı kısmında bir değişiklik gerçekleştirilmez.

Örnek proje için PHPMyAdmin paneli üzerinden MySQL veritabanında “citizens” isminde bir veritabanı oluşturulup bu veritabanı üzerinde “citizen” isminde 4 sütundan oluşan bir tablo oluşturulmuştur. (EK-4) Bu tablo vatandaşların isim,soyisim ve kimlik numarası bilgilerini saklamaktadır. Tablo dummy data (kukla veri) ile doldurulmuştur (EK-5). Haritalama işlemi oluşturulmuş olan bu tablo üzerinde gerçekleştirilmiştir. Hibernate yapılandırması için “application.properties” dosyası oluşturulmuştur. Bu dosya önceki günlerdekinin benzeri şekilde doldurulup gerekli düzenlemeler yapılmıştır. Yeni bir veritabanı oluşturulduğu için “application.properties” dosyasında veritabanına ulaşılmasını sağlayan URL adresinin sonuna veritabanının adı yazılmıştır. Hibernate başlatıldığı zaman yeni bir tablo oluşturmaya çalışmaması için DDL modu “create” yerine “none” olarak seçilmiştir.

spring.datasource.url=jdbc:mysql://localhost:3306/citizens

spring.jpa.hibernate.ddl-auto=none

Mapping işlemi “hibernate.cfg.xml” isimli XML dosyasında gerçekleştirilmektedir. Bu dosya “src/main/resources” dizinine manuel olarak oluşturulmuştur. Haritalama işlemi aşağıdaki gibi gerçekleştirilmiştir. “name” attribute’ları Entity katmanındaki ilgili sınıfın adı ve field’ları ile doldurulmuştur. “table” attribute’ına veritabanında bulunan tablonun adı yazılmıştır. “column” attribute’larına tabloda bulunan sütunların ismi field’lar ile eşleşecek şekilde yazılmıştır. “type” attribute’larına Hibernate’e özgü olan ancak değişkenlere gelebilecek değerleri saklayabilecek veri tipleri yazılmıştır.

<hibernate-mapping>

<class name = "Citizen" table = "citizen">

<id name = "id" type = "int" column = "id">

<generator class="native"/>

</id>

<property name = "name" column = "name" type = "string"/>

<property name = "surname" column = "surname" type = "string"/>

<property name = "citizen\_number" column = "citizen\_number" type = "string"/>

</class>

</hibernate-mapping>

Entity katmanında tabloya karşılık gelen Citizen isminde bir sınıf oluşturulmuştur. Sınıfa @Entity anotasyonu yazılarak Entity özelliği kazandırılmıştır. @Table(name=”citizen”) ifadesi ile bu entity’nin hangi tablo ile eşleştiği belirtilmiştir. Fieldların üstüne @Column(name=”sutun\_ismi”) anotasyonu ile ilgili fieldın tabloda hangi sütuna denk geldiği belirtilmiştir.

@Entity

@Table(name = "citizen")

public class Citizen {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

@Column(name = "surname")

private String surname;

@Column(name = "citizen\_number")

private String citizen\_number;

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getSurname() { return surname; }

public void setSurname(String surname) { this.surname = surname; }

public String getCitizen\_number() { return citizen\_number; }

public void setCitizen\_number(String citizen\_number) { this.citizen\_number = citizen\_number; }

}