Custom Tipler

Custom Tipler

- SELECT ve INSERT/UPDATE işlemleri sırasında veri üzerinde bir dönüşüm/çevrim ihtiyacı varsa **custom user type ihtiyacı** söz konusu demektir
- Custom tip implemantasyonu genel olarak **Nesne-JDBC karşılıklı dönüşümünü** gerçekleştirir
- JPA 2.1 ile birlikte gelen **AttributeConverter** arayüzü kullanılabilir
- Ya da Hibernate'in **UserType** arayüzü ile custom tip yazılabilir

JPA ve AttributeConverter

- JPA 2.1 ile gelen bir kabiliyettir
- Hibernate'deki **custom user type kabiliyeti**ne benzer, **ancak daha kısıtlı**dır
- Entity attribute değerini DB gösterimine ve geriye dönüştürme imkanı sağlar

```
public interface AttributeConverter<X,Y> {

    public Y convertToDatabaseColumn (X attribute);

    public X convertToEntityAttribute (Y dbData);
}
```

Java tipleridir

Attribute değerini DB değerine dönüştürür

DB değerini Entity attribute değerine dönüştürür

AttributeConverter Örneği

JAVA Eğitimleri

JPA tarafından tanınabilmesi için Converter anotasyonu ile işaretlenmelidir

```
@Converter
public class ColorConverter
        implements AttributeConverter<Color, String>{

    @Override
    public String convertToDatabaseColumn(Color attribute) {
        if(attribute == null) return null;
        int red = attribute.getRed();
        int green = attribute.getGreen();
        int blue = attribute.getBlue();
        return red + ":" + green + ":" + blue;
    }

    @Override
    public Color convertToEntityAttribute(String dbData) {
        if(dbData == null) return null;
        String[] rgb = dbData.split(":");
        int r = Integer.parseInt(rgb[0]);
        int g = Integer.parseInt(rgb[1]);
        int b = Integer.parseInt(rgb[2]);
        return new Color(r, g, b);
    }
}
```

AttributeConverter'ın Kullanımı

```
@Entity
public class Product {
    @Id
    private Long id;

    @Convert(converter=ColorConverter.class)
    @Column(name="PRODUCT_COLOR",columnDefinition="VARCHAR(11)")
    private java.awt.Color color;

    public Long getId() {
        return id;
    }

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

    public Color getColor() {
        return color;
    }

    public void setColor(Color color) {
        this.color = color;
    }
}
```

Eğer @Converter(autoApply=true) şeklinde tanımlanırsa burada @Convert anotasyonunu kullanmaya gerek kalmaz

Enum Tiplerin Custom Tip ile Eşleştirilmesi

- Eğer Enum tiplerin **ORDINAL veya STRING gösterimleri dışında bir değer**in veritabanında tutulması istenirse bu durumda **custom type mapping** yapmak gerekir

```
public enum Status {
    PENDING(0),SUCCESS(1),FAILED(-1);

    private int code;

    private Status(int code) {
        this.code = code;
    }

    public int getCode() {
    return code;
    }
}
```

Enum Tiplerin AttributeConverter ile Eşleştirilmesi

JPA API ile çalışılıyor ise AttributeConverter
Yazılarak enum tip için custom tip mapping yapılabilir

```
@Converter(autoApply=true)
public class StatusConverter
    implements AttributeConverter<Status, Integer> {

    @Override
    public Integer convertToDatabaseColumn(Status attribute) {
        if(attribute == null) return null;
        return attribute.getCode();
    }

    @Override
    public Status convertToEntityAttribute(Integer dbData) {
        if(dbData == null) return null;
        for(Status status:Status.values()) {
            if(status.getCode() == dbData) return status;
        }
        return null;
    }
}
```

Hibernate UserType Arayüzü

- Hibernate API ile çalışırken de custom tip mapping için **UserType implement** etmek gerekir
- UserType arayüzünün iki temel metodu vardır
  - **nullSafeGet**(): JDBC ResultSet'i process eder ve nesneyi elde eder
  - **nullSafeSet**(): Nesneyi JDBC PreparedStatement'a yazar
- Custom type kullanılmadan önce **global metadata** olarak tanımlanmalıdır
- Genellikle domain sınıflarının üstünde tanımlanır

Hibernate UserType Arayüzü

```
public interface UserType {


    Object nullSafeGet(ResultSet rs, String[] names,
    SharedSessionContractImplementor session, Object owner) throws HibernateException,
SQLException;

    void nullSafeSet(PreparedStatement st, Object value, int index,
SharedSessionContractImplementor session) throws HibernateException, SQLException;

    int[] sqlTypes();

    Class returnedClass();

    boolean equals(Object x, Object y) throws HibernateException;

    int hashCode(Object x) throws HibernateException;

    Object deepCopy(Object value) throws HibernateException;

    boolean isMutable();

    Serializable disassemble(Object value) throws HibernateException;

    Object assemble(Serializable cached, Object owner) throws HibernateException;

    Object replace(Object original, Object target,
                                        Object owner) throws HibernateException;
}
```

Enum Tiplerin UserType ile Eşleştirilmesi

```
public class StatusEnumType implements UserType {
    @Override
    public void nullSafeSet(PreparedStatement st, Object value, int index,
                                SharedSessionContractImplementor session)
                                throws HibernateException, SQLException {
        if (value == null) {
                st.setNull(index, Types.INTEGER);
        } else {
                st.setInt(index, ((Status)value).getCode());
        }
    }
    @Override
    public Object nullSafeGet(ResultSet rs, String[] names,
                              SharedSessionContractImplementor session,
                              Object owner) throws HibernateException, SQLException {
        int code = rs.getInt(names[0]);
        if(rs.wasNull()) {
                return null;
        }
        for(Object value : returnedClass().getEnumConstants()) {
            if(code == ((Status)value).getCode()) {
                return value;
            }
        }
        throw new IllegalStateException("Unknown code " + code + " for Status");
    }
...
}
```

Custom Enum UserType'ın Kullanımı

- Custom UserType attrbiute''larda kullanılmadan evvel sınıf düzeyinde **@TypeDef** anotasyonu ile **bir kereliğine tanımlanmalı**dır

```
@org.hibernate.annotations.TypeDef(
                name="statusEnumType",typeClass=StatusEnumType.class)
@Entity
public class Message {

    @Id
    private Long id;

    @org.hibernate.annotations.Type(type="statusEnumType")
    @Column(name="status_code", columnDefinition="integer")
    private Status status;

    ...

}
```

İletişim

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