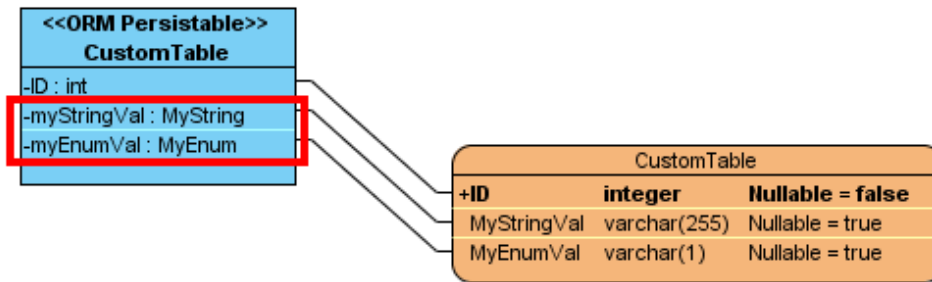


# About using User Type in ORM

This example simulate how customized type can be use in VP generated ORM layer source code. Assume you have the following model



The MyString type is a wrapper of String which having customized toString method. And the MyEnum is an customized enumerated type. We map MyString to varchar(255) and MyEnum to varchar(1) in database.

Both MyString and MyEnum having `<<ORM User Type>>` stereotype. This stereotype indicate type used in CustomTable are user type instead of primitive types.



When generate code, we will generate 2 extra classes, `MyStringUserType` and `MyEnumUserType` which implements `org.hibernate.usertype.UserType`. The `MyStringUserType` and `MyEnumUserType` are the linkage between persistent datatype and your custom datatype. Only the class structure for `MyStringUserType` and `MyEnumUserType` will be generated and no detail implementation included (since it is up to your decision on how it should behave).

```
package demo;

import java.io.Serializable;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Types;

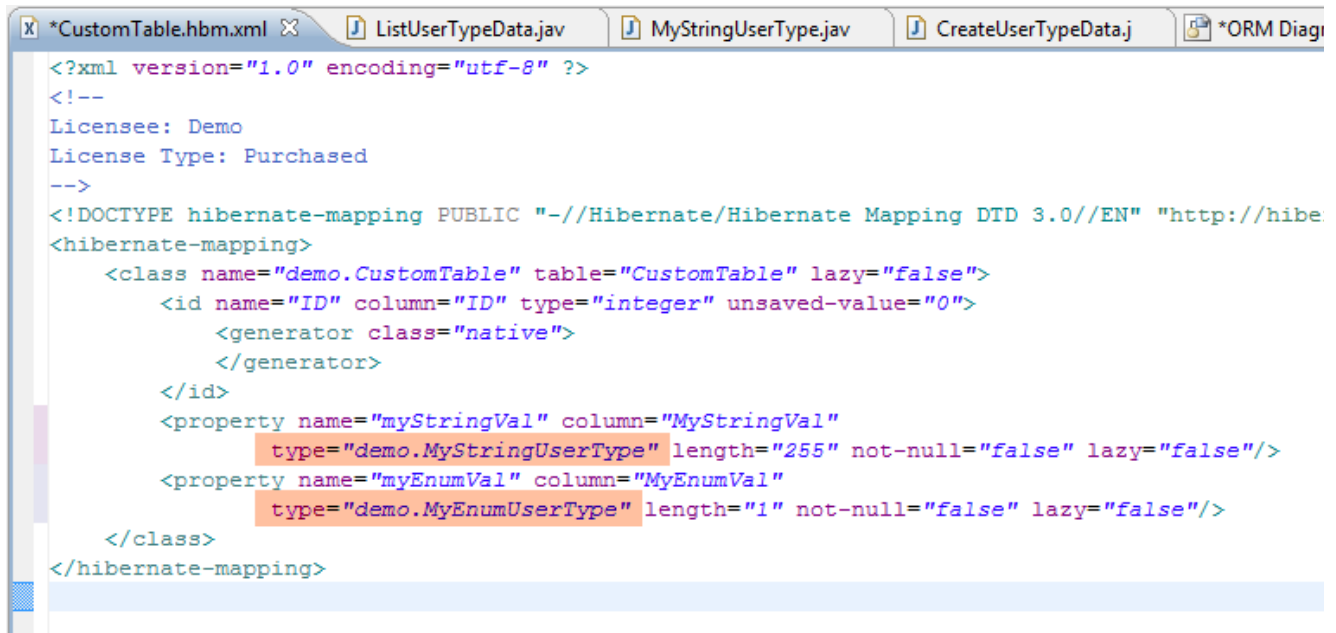
import org.hibernate.HibernateException;
import org.hibernate.usertype.UserType;

public class MyEnumUserType implements UserType {

    @Override
    public Object assemble(Serializable arg0, Object arg1)
        throws HibernateException {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public Object deepCopy(Object arg0) throws HibernateException {
        // TODO Auto-generated method stub
        return null;
    }
}
```

In the mapping file, it will showing the column is map to MyStringUserType and MyEnumUserType.



```
<?xml version="1.0" encoding="utf-8" ?>
<!--
Licensee: Demo
License Type: Purchased
-->
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibe
<hibernate-mapping>
  <class name="demo.CustomTable" table="CustomTable" lazy="false">
    <id name="ID" column="ID" type="integer" unsaved-value="0">
      <generator class="native">
        </generator>
      </id>
    <property name="myStringVal" column="MyStringVal"
      type="demo.MyStringUserType" length="255" not-null="false" lazy="false"/>
    <property name="myEnumVal" column="MyEnumVal"
      type="demo.MyEnumUserType" length="1" not-null="false" lazy="false"/>
  </class>
</hibernate-mapping>
```

The MyStringUserType and MyEnumUserType mainly handle how the data being load and save. You must implement the following methods in order to make it work

- public Object nullSafeGet(ResultSet resultSet, String[] names, Object owner) throws HibernateException, SQLException
- public void nullSafeSet(PreparedStatement statement, Object value, int index) throws HibernateException, SQLException
- public Class returnedClass()
- public int[] sqlTypes()

By implementing MyStringUserType and MyEnumUserType, you will able to save and load data through your custom types.

\* Please note that we will not create/generate MyString and MyEnum for you.