Hibernate with XML mapping files

Student Number: 1403724

Won Seob Seo

Before changing from Annotations to XML mapping, my classes look like

```
@XmlRootElement
@Entity
@NamedQuery(name = "FindAll", query = "from Car")
public class Car {

    @Id
    @GeneratedValue
    private Integer id;

    @ManyToOne
    @JoinColumn (name = "owner_column",
            nullable=false)
    private Person owner;

    @Column (name = "price_column")
    private double price;

    public Person getOwner() {
        return owner;
    }

    @XmlElement
```

```
@XmlRootElement
@Entity
public class Person {

    @Id
    @GeneratedValue
    @Column (name = "id_column")
    private Integer id;

    @Column (name = "name_column")
    private String name;

    @OneToMany(mappedBy="owner",
            targetEntity=Car.class,
fetch=FetchType.EAGER,
cascade=CascadeType.ALL)
    private Collection<Car> cars = new ArrayList<Car>();

    @XmlTransient
    public Collection<Car> getCars() {
        return cars;
```

And I deleted the hibernate or JPA related annotations and added hibernate mapping files. And added the mapping files as resources as shown below screenshots.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibe
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/mynewdatabase?zeroDateTimeBehavior=convertToNull</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.show_sql">true</property>
    <property name="hibernate.query.factory_class">org.hibernate.hql.internal.classic.ClassicQueryTranslatorFactory</property>
    <property name="hbm2ddl.auto">create</property>
    <mapping resource="com/wontheone/hiber01/Person.hbm.xml"/>
    <mapping resource="com/wontheone/hiber01/Car.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hiber
<hibernate-mapping>
    <class name="com.wontheone.hiber01.Car" table="car">
        <id name="id" type="integer">
            <column name="idColumn" />
            <generator class="increment"></generator>
        </id>
        <property name="price" type="double">
            <column name="priceColumn" />
        </property>
        <many-to-one name="owner"
        column="ownerId"
        not-null="true" />
    </class>
</hibernate-mapping>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Map
<hibernate-mapping>
    <class name="com.wontheone.hiber01.Person" table="person">
        <id name="id" type="integer">
            <column name="idColumn" />
            <generator class="increment"></generator>
        </id>
        <property name="name" type="string">
            <column length="37" name="nimiColumn" />
        </property>
    </class>
</hibernate-mapping>
```

In both ways, there are advantages and short-comings. The advantages of annotations would be its simplicity to use, less verbose way. But the short-coming is, it mixes Hibernate configurations with Plain-Old-Java-Object(POJO). Also configuration is scattered to the fields that need annotation. The advantages and short-comings of XML mapping method is the exact opposite of annotations method. That is, XML method tends to be more verbose and complex. But it can separately keep the all the configurations in the configuration files, so that it is easy to maintain and the java code looks cleaner.

In my opinion, although XML has advantages in separations and more modular approaches, I prefer annotations for now. That is because I don't have so many classes in the project and the codes of the classes are not too long, so it does not become so difficult to maintain even though Hibernate

configuration is inside the classes. Also a big plus for annotations was that, because the annotations were included in Hibernate library, I got code completion suggestions from IDE, which I didn't have in editing XML so for example I had to google if I wanted to generate IDs automatically but with annotations if I knew the first few characters, IDE helped me with finishing the annotations. But I will keep my eyes on learning more of the XML mapping method too.