

Hibernate Basic




Presented by Hosein Zare
Twitter: @zare88r






عدم تطابق Object-Relational

- اشیائی که در برنامه ی ما هستند به فرمت سطر و ستون نیستند ، آنها حالت را در قالب فیلدها نگه داری میکنند، پس نمیتوان آنها را مستقیماً در پایگاه داده نگهداری کرد
- در سطوح خیلی بالا ، عدم تطابق بین این دو را "مقاومت ظاهری تطبیق پذیری" یا object-relational impedance mismatch میگویند



عدم تطابق – ارث بری

- بزرگترین شاخصه ای که دنیای اشیا آن را پشتیبانی میکند و الگوهای رابطه ای آن را نمیشناسند ارث بری است.
- راهکار مستقیمی برای پیاده سازی این نوع رابطه نیست ، منتها با یک سری روش ها میتوان آن را شبیه سازی کرد



عدم تطابق – هویت (identity)

- اشیا در برنامه های جاوا دو قابلیت هویت و تساوی را دارند.
- در الگوهای رابطه ای سطرها و ستون ها با مقادیرشان شناسایی میشوند ، پایگاه داده از استراتژی "کلیدهای اصلی" برای یکتایی رکورد ها استفاده میکنند.

```
Trade trade1 = new Trade();  
Trade trade2 = trade1;
```

```
// Memory location is identical-identity check!  
if(trade1==trade2){..}
```

```
// Values are identical-equality check!  
if(trade1.equals(trade2)){..}
```



عدم تطابق – روابط

- در زبانهای شی گرا مثل جاوا ، روابط (association) ها از ویژگی های کلیدی به شمار می آیند.
- این روابط در پایگاه های داده ی رابطه ای توسط “کلیدهای خارجی” پیاده سازی میشوند
- به هر حال پیاده سازی روابطی چون یک به یک و چند به چند در پایگاه داده ی کمی با چالش روبرو است ولی غیر ممکن نیست



قسمتهای اصلی Hibernate

- کلاس های Persistent یا Entity
- فایل های پیکر بندی و نگاشت ها
- کد های مربوط به دسترسی ها و تغییرات برنامه



استفاده از Annotation ها


- نگاهت جداول و اشیا معمول توسط xml فایل ها صورت میگیرد اما میتوان برای سهولت بیشتر از Annotation ها استفاده کرد

```
@Entity
@Table(name="TRADES")
public class Trade implements Serializable {
    @Id
    private long tradeId = -1;
    private double quantity = 0;
    private String security = null;
    // getters and setters
    ...
}
```



استفاده از Annotation ها


- Hibernate از annotation های JPA استفاده میکند
- JPA یک کتابخانه ی استاندارد می باشد که برای ذخیره سازی Java Object ها استفاده میشود
- این Annotation ها در پکیج javax.persistence موجود میباشند
 - هر کلاس Persistent با یک @Entity تگ میشود
 - @Table جدول پایگاه داده مان را تعریف میکند
 - @Id برای مشخص کردن ستون کلید اصلی مان استفاده میشود



پیکربندی

hibernate.cfg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<hibernate-configuration>
  <session-factory>
    <property name="connection.url">jdbc:derby:memory:JH;create=true</property>
    <property name="connection.driver_class">
      org.apache.derby.jdbc.EmbeddedDriver</property>
    <property name="connection.username">jhuser</property>
    <property name="dialect">org.hibernate.dialect.DerbyDialect</property>
  </session-factory>
</hibernate-configuration>
```



پیکربندی

hibernate.cfg.xml

Property	Values	Notes
hibernate.show_sql	true/false	If true, all the SQL statements are printed out
hibernate.jdbc.fetch_size	>=0	Set the JDBC fetch size
hibernate.jdbc.batch_size	>=	Used to batch the statements
hibernate.hbm2ddl.auto	validate/update/ create/create-drop	Schema options
hibernate.connection.pool_size	>=1	Connection pool size



پیکربندی وابسته به برنامه

Hibernate همچنین از config شدن توسط کد نیز پشتیبانی میکند

```
Configuration cfg = new Configuration()
    .setProperty("hibernate.dialect",
        "org.hibernate.dialect.DerbyDialect")
    .setProperty("hibernate.connection.username", user);
    .setProperty("hibernate.connection.password", password);
    .setProperty("hibernate.connection.url",
        "jdbc:derby:memory:JH;create=true");
```



استراتژی ساخت اعداد یکتا

- هر شی باید با یک عدد یکتا در پایگاه داده ثبت گردد
- Hibernate چندین استراتژی برای ساختن اعداد یکتا استفاده میکند

- assigned
- sequence
- identity



Session API

- SessionFactory یک Factory کلاس میباشد که instance هایی از کلاس Session تولید میکند. این کلاس Thread-Safe میباشد و میتواند بین چندین کلاس و Thread به اشتراک گذاشته شود.
- SessionFactory همچنین Second Level Cache را پیاده سازی میکند که به طور عمومی بین تمامی کلاس ها یکتاست



Session API

- مادامی که SessionFactory کلید درگاه اتصال به پایگاه داده است ، Session کلید تعامل و دسترسی به اطلاعات آن است
- Session Single Thread میباشد
- First Level Cache در Session نگهداری میشود



تراکنش Transaction

- یکی از بخش های مهم کار با داده های سازمانی کار با تراکنش هاست. به زبان ساده تراکنش ها کار ما از بقیه جدا میکنند و آن را در یک محل ماندگار نگهداری میکنند بدون دغدغه ای که اطلاعاتمان به اشتباه نوشته یا واکشی شوند
- تراکنش ها خاصیت ACID دارند یعنی

Atomicity , Consistency , Isolation , Durability



تراکنش Transaction

- دو روش برای گرفتن Transaction ها وجود دارد ، یکی آنکه خودمان در کد آنها را مدیریت کنیم و دیگری اینکه آن را به دست Container بسپاریم



تراکنش Transaction

```
private void persist() {  
    Transaction tx = null;  
    try {  
        tx = session.beginTransaction();  
        Course course = createCourse();  
        session.save(course);  
        tx.commit();  
    } catch (HibernateException he) {  
        if(tx!=null)  
            tx.rollback();  
        throw he;  
    } finally{  
        session.close();  
    }  
}
```