## 1、主配置文件

hibernate.cfg.xml文件主要配置3方面信息：

1. 连接数据库的基本信息：驱动、URL、用户名、密码
2. Hibernate框架特性
3. 注册映射文件，即指定映射文件的位置。

### 1）数据库连接设置

DB连接四要素：

<**property name="hibernate.connection.driver\_class"**>

com.mysql.jdbc.Driver  
</**property**>  
<**property name="hibernate.connection.url"**>  
 jdbc:mysql:///hibernate?characterEncoding=UTF-8  
</**property**>  
<**property name="hibernate.connection.username"**>root</**property**>  
<**property name="hibernate.connection.password"**>mysql</**property**>

### 2）方言设置

可以从hibernate-core-5.0.1.Final.jar中的org.hibernate.dialect包中找到相应的类。复制全类名即为方言的值。

<**property name="hibernate.dialect"**>

org.hibernate.dialect.MySQL57Dialect

</**property**>

### 3）自动建表设置

hibernate.hbm2ddl.auto的取值有3种：

* create：每次加载主配置文件时都会删除上一次生成的表，然后再生成新表，哪怕两次表结构没有任何变化。
* create-drop：每次加载主配置文件时会生成表，但是sessionFactory一旦关闭，表就自动删除。
* update：当字段增加时，会添加字段；当字段减少时，不会减少字段。若表结构没变化，但数据变化时，会修改数据。

### 4）C3P0数据库连接池配置

c3p0-0.9.2版本之后分离,还需要导入mchange-commons-java的包。

注意各个jar包的版本。

<**dependency**>

<**groupId**>com.mchange</**groupId**>  
 <**artifactId**>c3p0</**artifactId**>  
 <**version**>0.9.5.2</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>com.mchange</**groupId**>  
 <**artifactId**>mchange-commons-java</**artifactId**>  
 <**version**>0.2.11</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>org.hibernate</**groupId**>  
 <**artifactId**>hibernate-c3p0</**artifactId**>  
 <**version**>5.4.1.Final</**version**>  
</**dependency**>

Hibernate5默认使用的是其自己开发的内置（built-in）连接池。连接池只是让代调试代码时使用，在真正产品中不能使用。可以从控制台的Hibernate启动信息中查看到。



注意，若不指定第三方数据源，将使用Hibernate内置的数据源。该数据源无法自动将数据库连接Connection释放，而数据源作为SessionFactory的属性，一直被当前程序占用，导致SessionFactory在程序运行结束后无法自动关闭。而这将导致整个程序无法结束。其表现为Eclipse的Terminate按钮一直显示为红色运行状态。

### 5）注册映射文件

映射文件若存在多个，则可写多个<mapping/>

<**mapping resource="mapping/Student.hbm.xml"**/>

## 2、映射文件

1. 实体类与数据库中表的映射关系；
2. 属性与表中字段的映射关系。

<**hibernate-mapping package="com.chance.pojo"**>

<**class name="Student" table="t\_student"**>  
 <**id name="id" column="sid"**>  
 <**generator class="native"**/>  
 </**id**>  
 <**property name="name" column="sname"**/>  
 <**property name="age" column="sage"**/>  
 <**property name="score" column="score"**/>  
 </**class**>  
</**hibernate-mapping**>

### 1）class标签

用于设置POJO类与数据表间的映射关系。

catalog属性：指定数据库。默认为主配置文件中指定的DB。

### 2）<id/>与<property/>标签

length属性：指定属性所映射字段的长度，单位字节。

not-null属性：为指定字段添加非空约束。

unique属性：为指定字段添加唯一性约束。

type属性：指定属性所映射的字段的类型。若省略Hibernate会自动从持久化类中检测到类型。这里的类型取值支持两类：Java类型与Hibernate类型。

### 3）Hibernate常用的内置主键生成策略

increment生成策略：

Hibernate自己维护主键的值。当准备在数据库表中插入一条新纪录时，首先从数据库表中获取当前主键字段的最大值，然后在最大值基础上加1，作为插入记录的主键值。

用其生成的主键字段所对应的属性类型可以是long、short、int及其封装类的类型。这种生成策略只有在没有其他进程向同一张表中插入数据时才能使用。在高并发下或集群下不能使用。

identity生成策略：

使用数据库自身的自增长来维护主键值。如mysql使用auto\_increment来维护。用其生成的主键字段所对应的属性类型可以是long、short、int及其封装类的类型。

对于插入操作，即使最后的执行是回滚，DB中记录主键值的变量也会增一。因为该生成策略在发生回滚之前已经调用过DB的主键自增，所以无论是否提交，对于DB来说已经执行。

sequence生成策略：  
在Oracle、DB2和PostgreSQL等数据库中创建一个序列（sequence），然后Hibernate通过该序列为当前记录获取主键值，从而为实体对象赋予主键字段映射属性值。用其生成的主键字段所对应的属性类型可以是long、short、int及其封装类的类型。

native生成策略：

由Hibernate根据所使用的的数据库支持能力从identity、sequence生成策略中选择一种。

uuid生成策略：

uuid生成策略采用UUID（Universally Unique Identifier，通用唯一识别码）算法来生成一个34字符串类型的主键值，该值使用IP地址、JVM的启动时间（精确到1/4秒）、当前系统时间和一个计数器值（在当前的JVM中唯一）经过计算产生，可以用于分布式的Hibernate应用中。产生的标识符属性是一个32位长度的字符串。使用这种生成策略，要求属性的类型必须为String类型。

这种标识符属性生成策略生成的数值可以保证多个数据库之间的唯一性，并且由于其生成与具体的数据库没有关系，所以其移植性较强。但由于该值是32位长的字符串，所以占用的数据库空间较大，并且检索速度较慢。不过，实际开发中使用这种生成策略较多。

assigned生成策略：

该生成策略的主键值来自于程序员的手工设置，即通过 setId()方法设置。属性类型可以是整型，也可以是 String，但一般为 String。此生成策略，主要应用于业务相关主键。例如学号、身份证号做主键。