Hibernate与Spring 配合生成表结构

前几天向大家介绍了一种用工具类生成数据表的方法，不过之前的方法需要使用一个跟项目关系不大的工具类。不免让人觉得有些多余，所以呢，今天再向大家介绍一种方法。即Hibernate与Spring配合生成表结构。

首先，要将Spring的信息配置的web.xml，配置Spring用于初始化容器对象的监听器。

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID"
version="2.5">
  <display-name>oa_01</display-name>

  <!-- 配置Spring用于初始化容器对象的监听器 -->
  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class>
  </listener>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext*.xml</param-value>
  </context-param>

  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

然后，将Hibernate的信息配置到Spring的配置文件中，将Hibernate交由Spring来管理。

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
    xmlns:tx="http://www.springframework.org/schema/tx"
```

```
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
                http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd
                http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">

    <!-- 自动扫描与装配bean -->
    <context:component-scan base-package="com.tgb.oa">
</context:component-scan>

    <!-- 导入外部的properties文件 -->
    <context:property-placeholder
location="classpath:jdbc.properties"/>

    <bean id="sessionFactory"
class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
        <!-- 指定hibernate配置文件的位置 -->
        <property name="configLocation"
value="classpath:hibernate.cfg.xml"></property>
        <!-- 配置c3p0数据库连接池 -->
        <property name="dataSource">
            <bean class="com.mchange.v2.c3p0.ComboPooledDataSource">
                <!-- 数据连接信息 -->
                <property name="jdbcUrl"
value="jdbc:mysql://127.0.0.1:3307/myoa"></property>
                <property name="driverClass"
value="com.mysql.jdbc.Driver"></property>
                <property name="user" value="root"></property>
                <property name="password" value="123456"></property>

                <!-- 初始化时获取三个连接（取值应在minPoolSize与
maxPoolSize之间。默认值: 3） -->
                <property name="initialPoolSize" value="3">
</property>
                <!-- 连接池中保留的最小连接数，默认值：3 -->
                <property name="minPoolSize" value="3"></property>
```

```
                <!-- 连接池中保留的最大连接数，默认值：15 -->
                <property name="maxPoolSize" value="5"></property>
                <!-- 当连接池中的连接数耗尽的时候，c3p0一次同时获取的连接数，默认值：3 -->
                <property name="acquireIncrement" value="3">
</property>
                <!-- 控制数据源内加载的PreparedStatements数量。如果maxStatements与maxStatementsPerConnection均为0，则缓存被关闭。Default: 0 -->
                <property name="maxStatements" value="8"></property>
                <!--maxStatementsPerConnection定义了连接池内单个连接所拥有的最大缓存statements数。Default: 0 -->
                <property name="maxStatementsPerConnection"
value="5"></property>
                <!--最大空闲时间,1800秒内未使用则连接被丢弃。若为0则永不丢弃。Default: 0 -->
                <property name="maxIdleTime" value="1800"></property>
            </bean>
        </property>
    </bean>

</beans>
```

这里我将数据库连接信息以及连接池都配置到了Spring中，当然你也可以将数据库连接信息写到Hibernate的配置文件里，Hibernate会有自己默认的连接池配置，但是它没有Spring的好用。具体写到哪看你需要吧。

接下来就是Hibernate的配置了，里面包括数据库方言、是否显示sql语句、更新方式以及实体映射文件。当然也可按上面说的将数据库连接信息写到里面。

hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>

        <property name="dialect">
```

```
            org.hibernate.dialect.MySQL5InnoDBDialect
        </property>

        <property name="show_sql">true</property>
        <property name="hbm2ddl.auto">update</property>

        <mapping resource="com/tgb/oa/domain/User.hbm.xml"/>

    </session-factory>
</hibernate-configuration>
```

实体映射文件，不做过多解释。

User.hbm.xml

```
<?xml version="1.0" <span style="font-family: Arial, Helvetica, sans-serif;">encoding="UTF-8"?></span>>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.tgb.oa.domain">

    <class name="User" table="T_User">
        <id name="id">
            <generator class="native"/>
        </id>
        <property name="name" />
    </class>

</hibernate-mapping>
```

实体类User.java

```
package com.tgb.oa.domain;

public class User {

    private String name;
    private  Long id;

    public String getName() {
```

```
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }

}
```

当tomcat启动的时候，会先找到web.xml，然后根据web.xml的配置，会找到spring中的applicationContext.xml的配置文件，在applicationContext.xml中有相应的SessionFactory的配置，里面有Hibernate的相关信息，接着就会找到Hibernate-cfg.xml，读取该文件，并找到实体映射文件User.hbm.xml，最后根据User.hbm.xml的配置映射成相应的表结构。

Tomcat启动以后，表结构也跟着生成了，生成表结构后的效果：

```
mysql> describe t_user;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| name  | varchar(255) | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
```

两种生成表结构的方式其实也没有哪种好，哪种不好之说。用工具类生成的方式不需要Spring的参与，但是需要一个工具类来支持；与Spring配合的方式不需要多余的东西，但是需要与Spring配合才能用。如果你只需要Hibernate那就用第一种，如果正好是配合Spring来使用那毫无疑问就用第二种。具体看情况吧。