* Mybatis

1. JDBC
2. 注册驱动
3. 获取连接
4. 执行sql语句
5. 得到结果集，处理结果集CURD（增，删，改，查）
6. 关闭资源

*Connection c = null;*

*Statement stmt = null;*

*try {*

*Class.forName("org.sqlite.JDBC");*

*c = DriverManager.getConnection("jdbc:sqlite:test.db");*

*c.setAutoCommit(false);*

*System.out.println("Opened database successfully");*

*stmt = c.createStatement();*

*ResultSet rs = stmt.executeQuery( "SELECT \* FROM COMPANY;" );*

*while ( rs.next() ) {*

*int id = rs.getInt("id");*

*String name = rs.getString("name");*

*int age = rs.getInt("age");*

*String address = rs.getString("address");*

*float salary = rs.getFloat("salary");*

*}*

*rs.close();*

*stmt.close();*

*c.close();*

*} catch ( Exception e ) {*

*}*

1. Mybatis 简介
2. MyBatis的前身就是iBatis, iBatis本是apache的一个开源项目，2010年这个项目由apahce sofeware foundation 迁移到了google code，并且改名为MyBatis。
3. MyBatis主要完成两件事：
4. 根据JDBC规范建立与数据库的连接；
5. 通过Annotaion/XML + JAVA反射技术，实现Java对象与关系数据库之间相互转化
6. Mybatis & hibernate

共同点：

* + 从配置文件(通常是XML配置文件中)得到 sessionfactory.
  + 由sessionfactory  产生 session
  + 在session 中完成对数据的增删改查和事务提交等.
  + 在用完之后关闭session 。
  + 在java 对象和 数据库之间有做mapping 的配置文件，也通常是xml 文件。

1. object/ relational mapping，ORM代表一种将对象和关系数据库表相互转换的[技术](http://www.shenmeshi.com/Science/Index.html)
2. 对Hibernate"O/R"而言

iBATIS 是一种"Sql Mapping"的ORM实现。

Hibernate的O/R Mapping实现了POJO和数据库表之间的映射，以及SQL的自动生成和执行。

程序员往往只需定义好了POJO到数据库表的映射关系，即可通过 Hibernate提供的方法完成持久层操作。程序员甚至不需要对SQL的熟练掌握，Hibernate/OJB会根据制定的存储逻辑，自动生成对应的 SQL并调用JDBC接口加以执行。  
Mybatis的着力点在于 POJO与SQL之间的映射关系。也就是说，Mybatis并不会为程序员在运行期自动生成SQL执行。具体的SQL需要程序员编写，然后通过映射配置文 件，将SQL所需的参数，以及返回的结果字段映射到指定POJO。

二者的对比：  
1． Mybatis非常简单易学，Hibernate相对较复杂，门槛较高。  
2．Sql文件易于管理维护，优化不用改动java 代码。  
3．当系统属于二次开发，无法对数据库结构做到控制和修改, 那Mybatis的灵活性将比Hibernat更适合。  
4. 系统数据处理量巨大，性能要求极为苛刻，这往往意味着我们必须通过经过高度优化的SQL语句（或存储过程）才能达到系统性能设计指标。在这种情况下iBATIS会有更好的可控性和表现。  
5． Hibernate和MyBatis都有相应的代码生成工具。可以生成简单基本的DAO层方法。针对高级查询，Mybatis需要手动编写SQL语句，以及ResultMap。而Hibernate有良好的映射机制，开发者无需关心SQL的生成与结果映射，可以更专注于业务流程。类似的，如果涉及到数据库字段的修改，Hibernate修改的地方很少，而Mybatis要把那些sql mapping的地方一一修改。  
6． 以数据库字段一一对应映射得到的PO和Hibernte这种对象化映射得到的PO是截然不同的，本质区别在于这种PO是扁平化的，不像Hibernate映射的PO是可以表达立体的对象继承，聚合等等关系的，这将会直接影响到你的整个软件系统的设计思路。

7．hibernate数据库无关性比较好，如果数据库有改动，mybatis要修改较多，而hibernate不需要。

1. Mybatis使用教程
2. Mybatis的两个jar包，一个config.xml配置文件，DAO层的java接口和sql映射关系的xml文件。
3. CDATA标签的使用

在 XML 元素中，"<" 和 "&" 是非法的。

"<" 会产生错误，因为解析器会把该字符解释为新元素的开始。

"&" 也会产生错误，因为解析器会把该字符解释为字符实体的开始。

包含大量 "<" 或 "&" 字符。为了避免错误，可以将脚本代码定义为 CDATA。

CDATA 部分中的所有内容都会被解析器忽略，当作普通文本处理。

1. Like查询的写法

*select \* from person where name like "%"#{name}"%"*

*select \* from person where name like '%'||#{name}||'%'*

*select \* from person where name like '%${name}%'*

*select \* from person where name like concat('%',#{ name},'%')*

1. 插入操作获取自增长主键

**Oracle(sequence):**

<insert id="insert" parameterType="User">

<selectKey keyProperty="id" resultType="int" order="BEFORE">

select SEQ\_TEST\_USER\_ID.nextval from dual

</selectKey>

insert into TEST\_USER (ID,NAME,AGE)values (#{id},#{name},#{age})

</insert>

**Mysql(auto\_increament):**

*<sql id='TABLE\_NAME'>TEST\_USER</sql>*

*<insert id=”insert” useGeneratedKeys=”true” keyProperty=”id” parameterType="User">*

*insert into TEST\_USER ( NAME,AGE) values (#{name},#{age})*

*</insert>*

*或者类似oracle的方法：*

*<insert id="insert" parameterType="User">*

*<selectKey keyProperty="id" resultType="int" order="BEFORE">*

*SELECT LAST\_INSERT\_ID()*

*</selectKey>*

*insert into TEST\_USER(ID,NAME,AGE)values (#{id},#{name},#{age})*

*</insert>*

1. Association和Connection

**Association:**

*<resultMap type="User" id="userResult">*

*<result property="id" column="id"/>*

*<result property="userCode" column="userCode" />*

*<result property="userName" column="userName" />*

*<result property="roleId" column="roleId" />  
　　　　<association property="role" javaType="Role" >*

*<result property="id" column="id"/>*

*<result property="roleCode" column="roleCode"/>*

*<result property="roleName" column="roleName"/>*

*</association>*

*</resultMap>*

*<select id="getUserListByRoleId" parameterType="Role" resultMap="userResult">*

*select u.\*,r.roleCode as roleCode,r.roleName as roleName from user u,role r where u.roleId = r.id and u.roleId = #{id}*

*</select>*

**Connection:**

获取指定用户的地址列表(user表-address表：1对多关系)

*<resultMap type="User" id="userMap">*

*<id property="id" column="userId"/>*

*//property是属性名 ofType是List《Address》的类对象*

*<collection property="addressList" ofType="Address">*

*<id property="id" column="a\_id"/>*

*<result property="postCode" column="postCode"/>*

*<result property="addressContent" column="addressContent"/>*

*</collection>*

*</resultMap>*

1. 缓存机制
2. MyBatis 提供了**一级缓存**和**二级缓存**的支持

|  |  |
| --- | --- |
|  | 1. 一级缓存: 存储作用域为Session，当 Session flush或close之后，该Session中的所有 Cache 就将清空，默认的以及缓存是开启的。 2. 二级缓存与一级缓存其机制相同，其存储作用域为 Mapper(Namespace)，并且可自定义存储源，如 Ehcache，二级缓存默认是关闭的 。 3. 对于缓存数据更新机制，当某一个作用域(一级缓存Session/二级缓存Namespaces)的进行了 C/U/D 操作后，默认该作用域下所有 select 中的缓存将被clear。 4. 当执行一条查询SQL时，流程为： 从二级缓存中进行查询 -> [如果缓存中没有，委托给 BaseExecutor] -> 进入一级缓存中查询 -> [如果也没有] -> 则执行 JDBC 查询。   一级缓存的使用：  二级缓存的使用：   1. 二级缓存默认为关闭，在mybatis.xml中配置:   元素的必需按照如下顺序配置:(properties>>settings>>typeAliases>>typeHandlers>>objectFactory>>objectWrapperFactory>>plugins>>environments>>databaseIdProvider>>mappers)  <settings>  <setting name="cacheEnabled" value="true"/>  </settings>   1. 当全局的二级缓存（setting中配置）设置为关闭时可以在Mapper XML中配置单个mapper的二级缓存为打开，配置如下：   <cache />   1. 当全局的二级缓存（setting中配置）设置为打开时，mapper中这个配置无效，即mapper中配置为关闭该mapper的二级缓存也是打开。 2. 实体类必需实现序列化接口 Serializable 3. 如果想对某条SQL单独对待，可以在SELECE语句中配置useCache,配置如下:   <select id="selectByPrimaryKey" parameterType="string" resultType="User" useCache="false">  深入研究缓存：http://www.iteye.com/topic/1112327 |
|  |  |

1. 利用工具生成java实体类，xml映射文件和接口

需要两个jar包（mybatis-generator-core-1.3.2.jar和mysql-connector-java-5.1.8-bin.jar），一个xml配置文件。

CMD命令行运行命令:

*java -jar C:\workarea.c\eclipse\workspace\CDT\_UI3.0\TicketCMS\src\test\mybatis-generator-core-1.3.2.jar –configfile C:\workarea.c\eclipse\workspace\CDT\_UI3.0\TicketCMS\src\test\mbgConfiguration.xm -overwrite*