<http://blog.csdn.net/cs408/article/details/45598795>

**SqlMapClient**对象

这个对象是iBatis操作数据库的接口（执行CRUD等操作），它也可以执行事务管理等操作。这个类是我们使用iBATIS的最主要的类。它是线程安全的。通常，将它定义为单例。（与hibernate中sessionFactory的定义类似）。如：

|  |
| --- |
| import java.io.Reader;  import com.ibatis.common.resources.Resources;  import com.ibatis.sqlmap.client.SqlMapClient;  import com.ibatis.sqlmap.client.SqlMapClientBuilder;  public class IbatisSQLMapConfig {      private static final SqlMapClient sqlMap;      //在静态区块中初试化返回      static {          try {              //声明配置文件的名称（映射文件被定义在其中）              String resource = "sql\_map\_config.xml";              //利用工具类Resources来读取到配置文件              Reader reader = Resources.getResourceAsReader(resource);              //创建SqlMapClient接口的变量实例              sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);          } catch (Exception e) {              e.printStackTrace();              throw new RuntimeException(                      "Error initializing MyAppSqlConfig class. Cause: " + e);          }      }      public static SqlMapClient getSqlMapInstance() {          //提供静态方法返回静态区块中得到的SqlMapClient          return sqlMap;      }  } |

主要用法：

如何获得刚插入记录的自增长**ID**值？

以下所有虚线上面代表User.xml中的内容，虚线下方是测试类中的内容：User类沿用上一篇中的User类

|  |
| --- |
| <insert id="insertUser" parameterClass="User">    insert into t\_user values (         null,#username#,#password#    )    <selectKey resultClass="int" keyProperty="id">         SELECT @@IDENTITY AS ID    </selectKey>   </insert> |
| User user = new User();         user.setUsername("张三");         user.setPassword("张三密码");           //如果主键是自动生成的，则其返回值可以通过<selectKey>标签来设置         //如果不通过<selectKey>标签来设置，则返回值为空！         //<selectKey >中的keyProperty，是指定User中的id属性，当调用结束之后，         //user对象的id值和insert方法的返回值都是这条记录的ID值！         Object obj = sqlMapper.insert("insertUser", user); |

**parameterClass**的使用

|  |
| --- |
| <insert id="insertUser" parameterClass="User">    insert into t\_user values (         null,#username#,#password#    )    <selectKey resultClass="int" keyProperty="id">         SELECT @@IDENTITY AS ID    </selectKey>   </insert>     <insert id="insertUser2">    insert into t\_user values (         null,#username#,#password#    )    <selectKey resultClass="int" keyProperty="id">         SELECT @@IDENTITY AS ID    </selectKey>   </insert> |

insertUser使用了parameterClass，所以必需传入User类型的对象

|  |
| --- |
| User user = new User();         user.setUsername("张三");         user.setPassword("张三密码");           //传递进去的对象，必须是User类型         Object obj = sqlMapper.insert("insertUser", user); |

insertUser2没有使用parameterClass，所以可以传入任意具有相应属性值的对象

|  |
| --- |
| JustAnObject anobj = new JustAnObject();         anobj.setUsername("用户名");         anobj.setPassword("用户密码");           //如果没有指定parameterClass属性，则任何一个具有相应属性值         //的对象都可以被传递进去         Object obj = sqlMapper.insert("insertUser2", anobj); |

**parameterMap**的使用

|  |
| --- |
| <parameterMap class="User" id="insertUser-param">    <parameter property="username"/>    <parameter property="password"/>   </parameterMap>   <insert id="insertUser" parameterMap="insertUser-param">    insert into t\_user values (         null,?,?    )    <selectKey resultClass="int" keyProperty="id">         SELECT @@IDENTITY AS ID    </selectKey>   </insert> |

parameterMap用于传入参数，以便匹配SQL语句中的?号

|  |
| --- |
| User user = new User();         user.setUsername("张三dd");         user.setPassword("张三密码dd");           Object obj = sqlMapper.insert("insertUser", user); |

利用parameterMap，可以定义参数对象的属性如何映射到SQL查询语句的动态参数上，注意parameterMap中<parameter/>标签的先后顺序不能颠倒！

如何将查询结果映射到不同的对象？**(resultClass**的使用**)**

package com.ibatis.model;

publicclassOtherObject {

    privateintid;

    private String prop1;

    private String prop2;

    publicint getId() {

       returnid;

    }

    publicvoid setId(int id) {

       this.id = id;

    }

    public String getProp1() {

       return Prop1;

    }

    publicvoid set Prop1 (String Prop1) {

       this. Prop1 = Prop1;

    }

    public String getProp2() {

       returnusername;

    }

    publicvoid setProp2 (String Prop2) {

       this.Prop2 = Prop2;

    }

}

|  |
| --- |
| <select id="selectUserForOtherObject" resultClass="com. ibatis.OtherObject" parameterClass="int">    select    username as prop1,    password as prop2    from t\_user where id=#value#   </select> |

|  |
| --- |
| //查找t\_user表，将其结果映射到一个属性名不同的对象中！         OtherObject obj = (OtherObject)sqlMapper.queryForObject("selectUserForOtherObject", 1);         System.*out*.println(obj.getProp1()+","+obj.getProp2()); |

如何将查询结果集映射到不同的对象？**(resultMap**的基本使用**)**

|  |
| --- |
| <resultMap class="com.ibatis.model.OtherObject" id="ooResult">    <result property="prop1" column="username"/>    <result property="prop2" column="password"/>   </resultMap>   <!--   如果使用resultMap来定义如何映射，则如下语句不可写成：   select username as prop1,password as prop2 ....   -->   <select id="selectUserForOtherObject2" parameterClass="int"resultMap="ooResult">        select        username,        password        from t\_user where id=#value#   </select> |

|  |
| --- |
| //查找t\_user表，将其结果映射到一个属性名不同的对象中！         OtherObject obj = (OtherObject)sqlMapper.queryForObject("selectUserForOtherObject2", 17);         System.*out*.println(obj.getProp1()+","+obj.getProp2()); |

如何将查询结果集映射为**xml**格式的数据？

|  |
| --- |
| <select id="selectXmlData" resultClass="xml" xmlResultName="User"parameterClass="int">    select \* from t\_user where id=#value#   </select>   <select id="selectXmlDatas" resultClass="xml" xmlResultName="User">    select \* from t\_user   </select> |

|  |
| --- |
| //查找t\_user表，将其结果映射到xml！         //返回值是xml形式的字符串         Object obj = (Object)sqlMapper.queryForObject("selectXmlData", 1);         System.*out*.println(obj);         //查找t\_user表，将其结果映射到xml！         List list  = (List)sqlMapper.queryForList("selectXmlDatas");         System.*out*.println(list); |

如何用**Map**类型的对象作为传入参数？

|  |
| --- |
| <!--   这里，可以使用全路径类名，如：   java.util.Map   java.util.HashMap   java.util.TreeMap   或   map   -->   <insert id="insertUser" parameterClass="map">    insert into t\_user values (         null,#username#,#password#    )   </insert> |

|  |
| --- |
| Map user = new TreeMap();         user.put("username", "Map用户");         user.put("password", "Map用户密码");         sqlMapper.insert("insertUser",user); |

如何将查询结果集的元素转换为**Map**类型的对象？

|  |
| --- |
| <!--   resultClass可以定义为java.util.HashMap类型，   将能自动转换   -->   <select id="selectMapUsers" resultClass="java.util.HashMap">    select \* from t\_user   </select> |

|  |
| --- |
| List list = (List)sqlMapper.queryForList("selectMapUsers");         System.*out*.println(list);         for (Iterator iter = list.iterator(); iter.hasNext();) {             Map map = (Map) iter.next();             //可在此输出map的数据         } |

事务处理

可以使用sqlMapClient的startTransaction/commitTransaction/endTransaction等方法来控制事务的边界。

如果与spring整合（这是iBatis推荐的方式），则我们需要在spring配置文件中指定其事务特性。