SqlMapConfig.xml

```
mybatis全局配置文件
内容
* properties(属性)
* setting(全局配置参数)
* typeAliases(类型别名)
* typeHandlers(类型处理器)
* objectFactory(对象工厂)
* plugins(插件)
* environments(环境集合属性对象)
 + environment(环境子属性对象)
  - transactionManager(事务管理)
        - dataSource(数据源)
* mappers(映射器)
```

1. properties属性

1. 需求

- 将数据库连接参数单独配置在db.properties中，只需要在SqlMapConfig.xml中来加载该配置文件的属性值。
- 在SqlMapConfig.xml中就不需要对数据库连接参数硬编码。
- 将数据库的连接参数只配置到db.properties中，方便对参数进行统一管理，其他xml可以引用该配置文件。

2. db.properties

```
jdbc.driver = com.mysql.jdbc.Driver
jdbc.url = jdbc:mysql://localhost:3306/mybatis?useUnicode=true&characterEncoding=utf-8
jdbc.username = root
jdbc.password = root
```

3. SqlMapConfig.xml中加载配置属性

```
<properties resource="db.properties" />
```

4. 修改数据库连接池参数

```
<dataSource type="POOLED">
    <property name="driver" value="${jdbc.driver}" />
    <property name="url" value="${jdbc.url}" />
    <property name="username" value="${jdbc.username}" />
    <property name="password" value="${jdbc.password}" />
</dataSource>
```

5. 注意：properties中还可以配置一些属性名和属性值

```
<properties resource="db.properties">
 <!-- 在properties中还可以配置一些属性名和属性值 -->
 <!-- <property name="" value="" />-->
</properties>
```

6. mybatis将按照以下顺序加载属性：

- 在properties元素体内定义的属性首先被读取。
- 然后会读取properties元素中的resource或url加载的属性，它会覆盖已读取的同名属性。
- 最后读取parameterType传递的属性，它会覆盖已读取的同名属性。
- 建议：不要在properties元素体内添加任何属性，只将属性值定义在properties属性文件中。在properties属性文件中，属性名要有一定的特殊性，

比如xxx.xxx.xxx

2. settings 全局参数配置

mybatis框架在运行时可以调整一些运行参数。
比如：开启二级缓存、开启延迟加载。。。

- 模板

```
<!-- settings是 MyBatis 中极为重要的调整设置，它们会改变 MyBatis 的运行时行为。 -->
<settings>
 <!-- 该配置影响的所有映射器中配置的缓存的全局开关。默认值true -->
 <setting name="cacheEnabled" value="true"/>
 <!--延迟加载的全局开关。当开启时，所有关联对象都会延迟加载。 特定关联关系中可通过设置fetchType属性来覆盖该项的开关状态。默认值false  -->
 <setting name="lazyLoadingEnabled" value="true"/>
 <!-- 是否允许单一语句返回多结果集（需要兼容驱动）。 默认值true -->
 <setting name="multipleResultSetsEnabled" value="true"/>
 <!-- 使用列标签代替列名。不同的驱动在这方面会有不同的表现， 具体可参考相关驱动文档或通过测试这两种不同的模式来观察所用驱动的结果。默认值tru
 <setting name="useColumnLabel" value="true"/>
 <!-- 允许 JDBC 支持自动生成主键，需要驱动兼容。 如果设置为 true 则这个设置强制使用自动生成主键，尽管一些驱动不能兼容但仍可正常工作（比如 D
 <setting name="useGeneratedKeys" value="false"/>
 <!--  指定 MyBatis 应如何自动映射列到字段或属性。 NONE 表示取消自动映射；PARTIAL 只会自动映射没有定义嵌套结果集映射的结果集。 FULL 会自动
 <!-- 默认值PARTIAL -->
 <setting name="autoMappingBehavior" value="PARTIAL"/>

 <setting name="autoMappingUnknownColumnBehavior" value="WARNING"/>
 <!--  配置默认的执行器。SIMPLE 就是普通的执行器；REUSE 执行器会重用预处理语句（prepared statements）； BATCH 执行器将重用语句并执行批量更
 <setting name="defaultExecutorType" value="SIMPLE"/>
 <!-- 设置超时时间，它决定驱动等待数据库响应的秒数。 -->
 <setting name="defaultStatementTimeout" value="25"/>

 <setting name="defaultFetchSize" value="100"/>
 <!-- 允许在嵌套语句中使用分页（RowBounds）默认值False -->
 <setting name="safeRowBoundsEnabled" value="false"/>
 <!-- 是否开启自动驼峰命名规则（camel case）映射，即从经典数据库列名 A_COLUMN 到经典 Java 属性名 aColumn 的类似映射。  默认false -->
 <setting name="mapUnderscoreToCamelCase" value="false"/>
 <!-- MyBatis 利用本地缓存机制（Local Cache）防止循环引用（circular references）和加速重复嵌套查询。
        默认值为 SESSION，这种情况下会缓存一个会话中执行的所有查询。
        若设置值为 STATEMENT，本地会话仅用在语句执行上，对相同 SqlSession 的不同调用将不会共享数据。  -->
 <setting name="localCacheScope" value="SESSION"/>
 <!-- 当没有为参数提供特定的 JDBC 类型时，为空值指定 JDBC 类型。 某些驱动需要指定列的 JDBC 类型，多数情况直接用一般类型即可，比如 NULL、VA
 <setting name="jdbcTypeForNull" value="OTHER"/>
 <!--  指定哪个对象的方法触发一次延迟加载。 -->
 <setting name="lazyLoadTriggerMethods" value="equals,clone,hashCode,toString"/>
</settings>
```

3. typeAliases（别名） 重点

1. 需求

- 在mapper.xml中，定义很多的statement，statement需要parameterType指定输入参数的类型、需要resultType指定输出结果的映射类型。
- 如果在指定类型时输入类型全路径，不方便进行开发，可以针对parameterType或resultType定义一些别名，在mapper.xml中通过别名定义，方便开发。

2. mybatis默认支持别名

别名 | 映射类型 --- | --- _byte | byte _long | long _short | short _int | int _integer | int _double | double _float | float _boolean | boolean string | String byte | Byte long | Long short | Short int | Integer integer | Integer double | Double float | Float boolean | Boolean date | Date decimal | BigDecimal bigdecimal | BigDecimal

3. 自定义别名

- 单个别名定义

```
<typeAliases>
    <!-- 针对单个别名定义
     type：类型路径
     alias：别名-->
    <typeAlias type="com.swxc.mybatis.po.User" alias="user" />
</typeAliases>
```

```
<select id="findUserById" parameterType="int" resultType="user">
```

- 批量别名定义（常用）

```
<typeAliases>
    <!-- 批量定义别名
     指定包名，mybatis自动扫描包中的po类，自动定义别名，别名就是类名（首字母大写或小写都可以）-->
    <package name="com.swxc.mybatis.po" />
</typeAliases>
```

4. typeHandlers（类型处理器）

```
mybatis中通过typeHandlers完成jdbc和java类型的转换。
通常情况下，mybatis提供的类型处理器满足日常需要，不需要自定义。
```

1. mybatis支持的类型处理器

类型处理器 | Java类型 | JDBC类型 --- | --- | --- BooleanTypeHandler | Boolean,boolean | 任何兼容的布尔值 ByteTypeHandler | Byte,byte | 任何兼容的数字或字节类型 ShortTypeHandler | Short,short | 任何兼容的数字或短整型 IntegerTypeHandler | Integer,int | 任何兼容的数字或整型 LongTypeHandler | Long,long | 任何兼容的数字或长整型 FloatTypeHandler | Float,float | 任何兼容的数字或单精度浮点型 DoubleTypeHandler | Double,double | 任何兼容的数字或双精度浮点型 BigDecimalTypeHandler | BigDecimal | 任何兼容的数字或十进制小数类型 StringTypeHandler | String | CHAR或VARCHAR类型 ClobTypeHandler | String | CLOB或LONGVARCHAR类型 NStringTypeHandler | String | NVARCHAR或NCHAR类型 NClobTypeHandler | String | NCLOB类型 ByteArrayTypeHandler | byte[] | 任何兼容的字节流类型 BlobTypeHandler | byte[] | BLOB或LONGVARBINARY类型 DateTypeHandler | Date(java.util) | TIMESTAMP类型 DateOnlyTypeHandler | Date(java.util) | DATE类型 TimeOnlyTypeHandler | Date(java.util) | TIME类型 SqlTimestampTypeHandler | Timestamp(java.sql) | TIMESTAMP类型 SqlDateTypeHandler | Date(java.sql) | DATE类型 SqlTimeTypeHandler | Time(java.sql) | TIME类型 ObjectTypeHandler | 任意 | 其他或未指定类型 EnumTypeHandler | Enumeration类型 | VARCHAR-任何兼容的字符串类型，作为代码存储（而不是索引）

5. mappers（映射配置）

1. 通过resource加载单个映射文件

```
<!-- 通过resource方法一次加载一个映射文件 -->
<mapper resource="mapper/UserMapper.xml" />
```

2. 使用url加载单个映射文件

```
<mapper url="D:\workspace\mybatis\config\sqlmap\User.xml" />
```

3. 通过mapper接口加载单个mapper

```
<!-- 通过mapper接口加载映射文件
 遵循一些规范：需要将mapper接口类名和mapper.xml映射文件名称保持一致，且在一个目录中。
 规范前提是：使用的是mapper代理的方法。
 -->
<mapper class="com.swxc.mybatis.mapper.UserMapper" />
```

4. 批量加载多个mapper

```
<mappers>
    <!-- 批量加载mapper
     指定mapper接口的包名mybatis自动扫描包下的所有的mapper接口进行加载。
     遵循一些规范：需要将mapper接口类名和mapper.xml映射文件名称保持一致，且在一个目录中。
     规范前提是：使用的是mapper代理的方法。
     -->
    <package name="com.swxc.mybatis.mapper" />
</mappers>
```