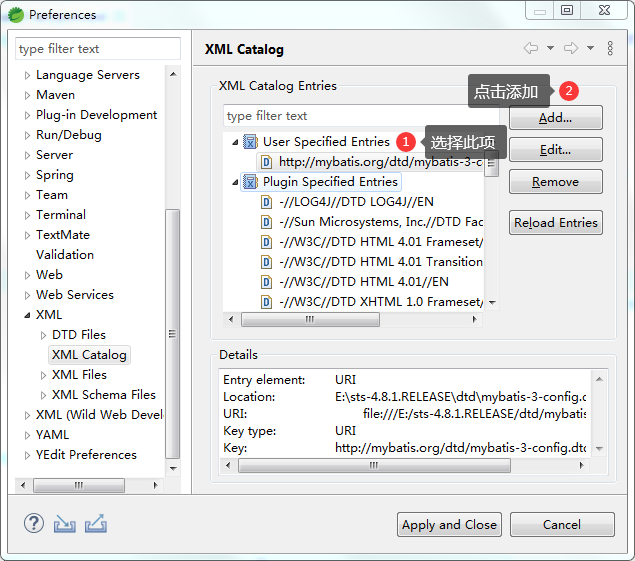
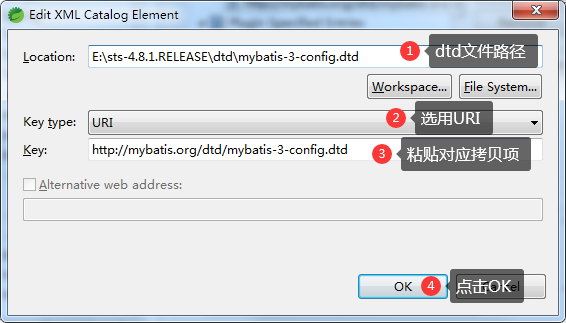
本文档仅记录eclipse使用mybatis的接口编程方式的使用方法

## eclipse配置xml自动提示

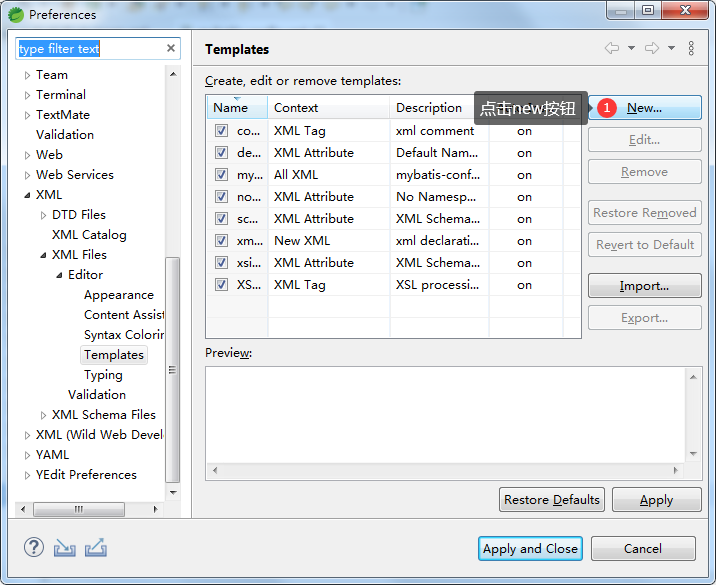
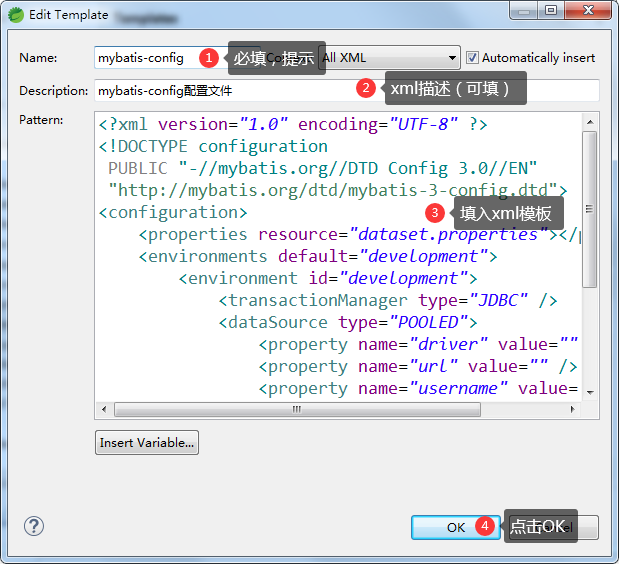
1. 打开mybatis.jar找org\apache\ibatis\builder\xml路径下的mybatis-3-config.dtd和mybatis-3-mapper.dtd。拷贝这两文件到eclipse安装目录中。
2. 拷贝：http://mybatis.org/dtd/mybatis-3-config.dtd和http://mybatis.org/dtd/mybatis-3-mapper.dtd（根据mybatis版本修改红色字）。

eclipse：Window → Perferences → XML → XML Catalog

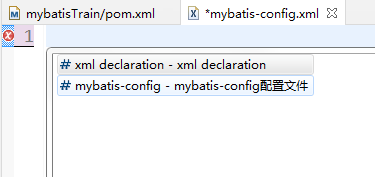
 

## eclipse配置xml自动生成模板

eclipse: Window → Preference → XML→ XML Files → Editor → Templates

使用模板：创建xml后，快捷键：alt + /调出自定义模板选择框（如下图）



## mybatis-config.xml（重要）

mybatis-config.xml是mybatis的重要且基础配置文件，基础格式如下。

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

<environments default=*"development"*>

<environment id=*"development"*>

<transactionManager type=*"JDBC"* />

<dataSource type=*"POOLED"*>

<property name=*"driver"* value=*""* />

<property name=*"url"* value=*""* />

<property name=*"username"* value=*""* />

<property name=*"password"* value=*""* />

</dataSource>

</environment>

</environments>

<mappers>

<mapper resource=*""* />

<mapper class=*""* />

<mapper url=*""* />

</mappers>

</configuration>

### environments

environments下可配置多个环境，用default指定项目运行时使用哪个环境，default所带值和众多environment的id必须吻合一个。



<transactionManager type="JDBC" />：事务管理器，type可选值JDBC,MANAGED。

JDBC:

MANAGED:

<dataSource type="POOLED">：数据源，type可选值UNPOOLED，POOLED，JNDI。

UNPOOLED:为每一个SQL请求开一个新连接，使用完毕后关闭。

POOLED:mybatis创建一个连接池，SQL请求用完归还入池。

JNDI:

<mapper resource/class/url="" />：mapper中属性值可选

resource：

class：

url：

### 引入properties配置文件

添加配置<properties resource="dataset.properties"></properties>数据库配置就可以写在properties文件中了，应用格式如下（重点看红字）

dataset.properties配置文件

jdbc.driver=..

jdbc.url=...

jdbc.username=...

jdbc.password=...

mybatis-config.xml配置文件

**<properties resource=*"dataset.properties"*></properties>**

<environments default=*"development"*>

<environment id=*"development"*>

<transactionManager type=*"JDBC"* />

<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>

</environment>

</environments>

### settings配置mybatis

### typeAliases配置java类别名

#### 配置单个别名

alias就是别名名称。

<typeAliases>

<typeAlias type=*"com.pojo.Person"* alias=*"Per"*/>

</typeAliases>

除了xml中配置方式，在pojo类中使用@Alias("别名称")也可以达到同样目的。

#### 配置包下所有类的别名

Mybatis扫描该包下所有类纳入别名命名，规则是大写首字母

<typeAliases>

<package name=*"com.pojo"* />

</typeAliases>

### 利用plugins配置mybatis插件

## mapper.xml

运算符：xml中大于小于符号这样写

|  |  |  |
| --- | --- | --- |
| < | &lt; | 小于 |
| > | &gt; | 大于 |
| & | &amp; | 与 |

mapper.xml是SQL映射文件，注意事项如下：

1. mapper.java中接口方法必须和mapper.xml中的操作名一样且相同
2. Mapper.java中接口方法的参数必须和mapper.xml中的参数类型一样

基础格式如下:

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace=*"com.mybaties.EmployeeMapper"*>

<insert id=*""*>...</insert>

<delete id=*""*>...</delete>

<select id=*""* resultType=*""*>...</select>

<update id=*""*>...</update>

</mapper>

### insert

#### 简单添加

Mapper.java

public void insertAPerson1(@Param("id") Integer id,

@Param("name") String name,

@Param("age") Integer age,

@Param("email") String email);

Mapper.xml

<insert id="insertAPerson" parameterType="Person">

insert into person(name,age,email,id) values ( #{name}, #{age},#{email},#{id} )

</insert>

#### 单对象添加

单对象添加，映射接口中方法的参数可以写对象，但是在SQL语句中还需要手动对齐字段。

Mapper.java

public void insertAPerson(@Param("person") Person person);

Mapper.xml

<insert id="insertAPerson" parameterType="com.pojo.Person">

insert into person(name,age,email,id) values

(#{person.name}, #{person.age}, #{person.email}, #{person.id} )

</insert>

#### 对象集添加

Mapper.java

public void insertPersons(@Param("persons") List<Person> persons);

Mapper.xml

<insert id="insertPersons" parameterType="java.util.List" >

insert into person(name,age,email,id) values

<foreach collection="persons" item="item" index="index" separator="," >

( #{item.name},#{item.age},#{item.email},#{item.id} )

</foreach>

</insert>

#### 不对应添加

在数据表字段与实体对象属性不对应的前提下，手动对应即可。

Mapper.java

略

Mapper.xml

略

### delete

Mapper.java

public void deletePerson(@Param("id") Integer id);

Mapper.xml

<delete id="deletePerson">

delete from person where id = ${id}

</delete>

### select

#### 普通查询

Mapper.java

public Person selectAPerson(Integer id)

Mapper.xml

<select id="selectAPerson" resultType="com.pojo.Person">

select \* from person where id = #{id}

</select>

注意:sql中查询的字段若不包含所有pojo中所有属性也不会出错。

#### 查询对象集

条件是所查字段必须与对象属性对应（包括）

Mapper.java

public List<Person> selectPersons(@Param("id") Integer id);

Mapper.xml

<select id="selectPersons" resultType="com.pojo.Person">

select \* from person where id > #{id}

</select>

#### resultMap查询不对应的对象集

数据表字段和实体属性并不一一对应，使用resultMapper进行手动对齐。Column是数据表字段名，property是java类属性。

Mapper.java

public List<Person> selectPersons(@Param("id") Integer id);

Mapper.xml

<!--property是java类属性。column是数据库字段-->

<resultMap type="com.pojo.Person" id="resultMapPerson">

<result column="id" property="id" javaType="java.lang.String" jdbcType="VARCHAR"/>

<result column="name" property="name" javaType="java.lang.String" jdbcType="VARCHAR"/>

<result column="age" property="age" javaType="java.lang.Integer" jdbcType="INTEGER"/>

<result column="email" property="email" javaType="java.lang.String" jdbcType="VARCHAR"/>

</resultMap>

<select id="selectPersons1" resultMap="resultMapPerson">

select \* from person where id > #{id}

</select>

注意：javaType和jdbcType分别是java类型和数据表类型，相同的情况下可以忽略不写，但为了严谨，建议写上。此外，若java类中多余属性并没有对应的查询结果，也不会影响，mybatis会自动忽略掉Java类中没有对应的属性。如下是javaType和jdbcType对应关系参考:

|  |  |  |
| --- | --- | --- |
| Java | MySQL | Oracle |
| java.lang.String | char |  |
|  | varchar |  |
|  | longtext |  |
| java.lang.Integer | int |  |
|  | integer |  |
| java.lang.Double |  |  |
|  |  |  |
| Java.math.BigDecimal | decimal（） |  |
|  | numeric（） |  |
| java.lang.Boolean | bit(1:true,0:false) |  |
| java.sql.Date | date |  |
|  | datetime |  |

#### 分页

##### 用SQL分页

简单利用SQL提供的语句进行分页，但是在web应用中，这样的分页无法做到查询后一同返回对应页数值。

Mapper.java

/\*current：当前页，pagesize：每页数量\*/

public List<Person> getByPage(@Param("current") Integer current,@Param("pagesize") Integer pagesize);

Mapper.xml

<select id="selectPersonsPage" resultType="Person" >

select <include refid="personfield"></include> from person

order by id limit **#{current},#{pagesize}**

</select>

##### ipage分页（mybatis-plus提供）

略

### update

#### 普通更新

Mapper.java

public void modName(@Param("id") Integer id,@Param("name") String name)

Mapper.xml

<update id="modName">

update person set name = #{name} where id = #{id}

</update>

#### 更新单对象

Mapper.java

public void modPerson(@Param("person") Person person);

Mapper.xml

<update id="modPerson" parameterType="Person">

update person set name=#{person.name},email=#{person.email},age=#{person.age}

where id = #{person.id}

</update>

#### 更新对象集

更新对象集即可使用对象list也可使用Map<name,List>。

mybatis-config.xml

<dataSource type="POOLED" >

<property name="driver" value="com.mysql.cj.jdbc.Driver" />

<property name="url" value="jdbc:mysql://localhost:3306/train?characterEncoding=utf-8

&amp;useSSL=false&amp;serverTimezone=UTC&amp;**allowMultiQueries=true**" />

<property name="username" value="root" />

<property name="password" value="root" />

</dataSource>

mapper.java

public void modPersons(@Param("persons") List<Person> persons);

mapper.xml

<update id="modPersons" parameterType="java.util.List">

<foreach collection="persons" index="index" item="item" separator=";" close=";">

update person set name=#{item.name},email=#{item.email},age=#{item.age}

where id = #{item.id}

</foreach>

</update>

### 事务

### 存储过程

### 注释

Mapper.xml文件使用<!-- -->打注释，不能用SQL注释形式“--”

### foreach

配置在sql语句中用于循环组织sql语句。

例如：

<foreach collection="persons" index="index" item="item" separator=";" close=";">

update person set name=#{item.name} where id = #{item.id}

</foreach>

collection:遍历项，需要接口传递进来，通常是list或list的实现类

index:下标，可以在sql语句中使用

item:遍历项,在sql中的使用格式为：item.属性

separator:分隔符号，常常用于分割sql语句

close:结束符号（上述例子其实也可以不写）

### if

配置在sql语句中mybatis根据条件动态选择sql执行。

<select id="selectGrade" resultType="Person">

select sum(score) from grade where id = #{id}

<if test="timer == day">group by substr(time,1,11)</if>

<if test="timer != day">group by name</if>

</select>

### sql

可被其他语句引用的可重用语句块;查询，插入都涉及到指定字段，重复写浪费时间。使用<include refid="">调用即可。

Mapper.xml

<sql id="personfield">

name,age,email,id

</sql>

<!--调用-->

<select id="selectAPerson" resultType="Person">

select

<include refid="personfield"></include>

from person where id = #{id}

</select>

<insert id="insertAPerson1">

insert into person(<include refid="personfield"></include> )

values ( #{name}, #{age},#{email},#{id} )

</insert>