# Mybatis概述

## Mybatis是什么

MyBatis 是一个优秀持久层框架，由apache的开源项目iBatis演化而来，主要用于解决数据持久化问题，底层实现了对JDBC操作的封装。

参考网址：http://www.mybatis.org

## MyBatis 应用场景

* 1. 简化JDBC编写步骤的复杂度
  2. 更好的实现ORM(对象关系映射)

提示:目前市场上还有一些类似的ORM框架,比如Hibernate.

## MyBatis 核心API

MyBatis 框架在实现数据的持久化操作时，需要关注了解的核心配置及对象为：

* + 1. Configs（xml） 配置文件&映射文件(mapper)
    2. SqlSessionFactoryBuilder（加载配置文件,创建SQLSessionFactory对象）
    3. SqlSessionFactory（负责创建session对象）
    4. SqlSession（负责执行SQL操作）

对以上API，先基本了解，然后再实践过程中再强化理解即可。

# Mybatis 编程实现

## Mybatis 应用配置

### MyBatis 依赖配置

要使用 MyBatis， 只需将 mybatis-x.x.x.jar 文件置于 classpath 中即可。

如果使用 Maven 来构建项目，则需将下面的 dependency 代码置于 pom.xml 文件中：

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>x.x.x</version>

</dependency>

### MyBatis 映射配置

MyBatis 中通过映射文件的配置实现SQL语句与具体操作的映射，例如:

现在mysql的test数据库中创建表：

CREATE TABLE `sys\_users` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`username` varchar(200) NOT NULL,

`password` varchar(200) NOT NULL,

`phone` varchar(20) NOT NULL,

`createdDate` date DEFAULT NULL,

`modifiedDate` date DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8

写入测试数据

insert into sys\_users

values

(null,'A','123456','139',now(),now()),

(null,'B','123456','137',now(),now());

定义映射文件（定义在资源目录的mapper包下）

<?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.project.sys.dao.SysUserDao"*>

<select id=*"findUsers"*

resultType=*"map"*>

select \* from sys\_users

</select>

</mapper>

### MyBatis 核心配置

此配置文件放在maven项目的resource根目录下,起名为mybatis-config.xml.

<?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=*"com.mysql.jdbc.Driver"*/>

<property name=*"url"* value=*"jdbc:mysql:///test"*/>

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

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

</dataSource>

</environment>

</environments>

<mappers>

<mapper resource=*"mapper/SysUserMapper.xml"*/>

</mappers>

</configuration>

## 创建SqlSessionFactory

每个基于 MyBatis 的应用都是以一个 SqlSessionFactory 的实例为中心的。这个SqlSessionFactory 的实例可以通过 SqlSessionFactoryBuilder 基于xml的配置创建。例如

SqlSessionFactory sessionFactory=

**new** SqlSessionFactoryBuilder()

.build(Resources.*getResourceAsStream*("mybatis-config.xml"));

## 创建SqlSession 执行操作

SqlSession 对象代表了一次SQL会话，包含了面向数据库执行 SQL 命令所需的所有方法。可以通过 SqlSession 实例来直接执行已映射的 SQL 语句。例如：

SqlSession sqlSession=sessionFactory.openSession();

List<Map<String,Object>> list=

sqlSession.selectList("com.project.sys.dao.SysUserDao.findUsers");

System.***out***.println(list);

sqlSession.close();

对于这样的操作，也可以先定义一个接口，然后通过接口类型对象去执行相关操作，例如

**package** com.project.sys.dao;

**import** java.util.List;

**import** java.util.Map;

**public** **interface** SysUserDao {

**public** List<Map<String,Object>> findUsers();

}

接口定义以后，设置mapper映射文件中的命名空间为这个接口的全类名，例如

<?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.project.sys.dao.SysUserDao"*>

<select id=*"findUsers"* resultType=*"map"*>

select \* from sys\_users

</select>

</mapper>

基于这个配置文件，实现SQL查询

SqlSession sqlSession=sessionFactory.openSession();

SysUserDao dao=sqlSession.getMapper(SysUserDao.**class**);

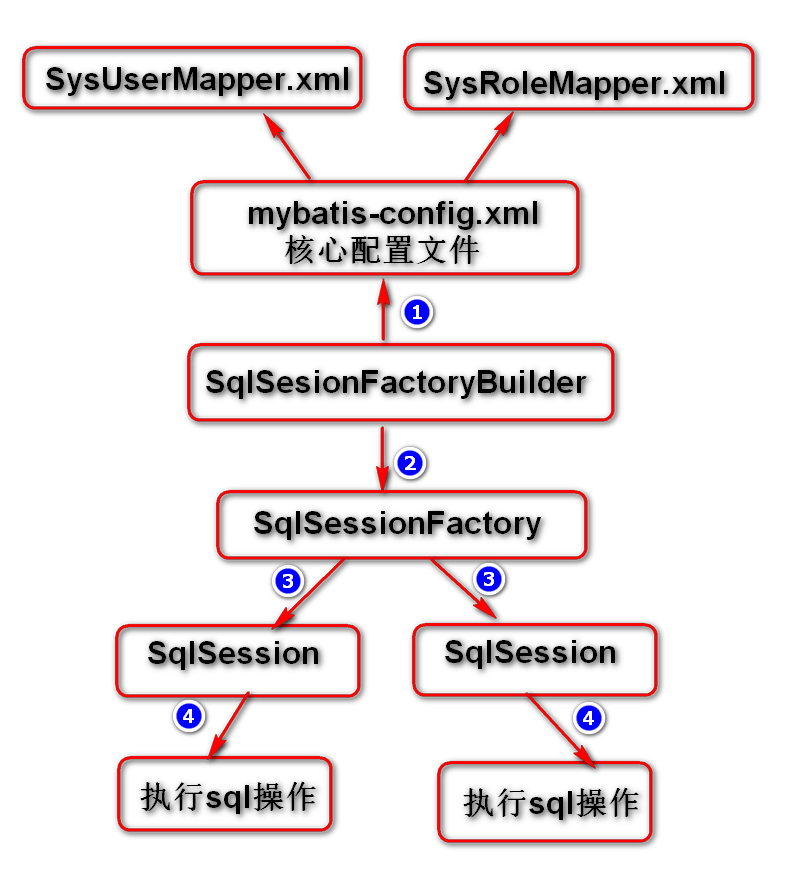
List<Map<String,Object>> list=dao.findUsers();

System.***out***.println(list);

sqlSession.close();

## MyBatis 编程流程总结

Mybatis 的执行流程可以简单概括为加载配置，SQL解析，SQL执行，结果映射。



# Mybatis 编程增强

## 核心配置配置文件

MyBatis 的配置文件包含了会深深影响 MyBatis 行为的设置（settings）和属性（properties）信息。文档的顶层结构如下：

1. configuration 配置
2. properties 属性
3. settings 设置
4. typeAliases 类型别名
5. typeHandlers 类型处理器
6. objectFactory 对象工厂
7. plugins 插件
8. environments 环境
9. environment 环境变量
10. transactionManager 事务管理器
11. dataSource 数据源
12. databaseIdProvider 数据库厂商标识
13. mappers 映射器

### properties 属性

在java代码对应的资源目录resources下创建config.properties文件，内容如下：

driver=com.mysql.jdbc.Driver

url=jdbc:mysql:///test

username=root

password=root

在核心配置文件中引入此配置，并获取对应值

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

<dataSource type="POOLED">

<property name="driver" value="${driver}"/>

<property name="url" value="${url}"/>

<property name="username" value="${username}"/>

<property name="password" value="${password}"/>

</dataSource>

这个例子中的 username 和 password 将会由 properties 元素中设置的相应值来替换，目的是让程序变得更加灵活。

### Setting元素

这是 MyBatis 中极为重要的调整设置，它们会改变 MyBatis 的运行时行为，例如MyBatis 缓存设置，延迟加载设置等。

<settings>

<setting name=*"cacheEnabled"* value=*"true"*/>

<setting name=*"lazyLoadingEnabled"* value=*"true"*/>

<setting name=*"multipleResultSetsEnabled"* value=*"true"*/>

<setting name=*"useColumnLabel"* value=*"true"*/>

<setting name=*"useGeneratedKeys"* value=*"false"*/>

<setting name=*"autoMappingBehavior"* value=*"PARTIAL"*/>

<setting name=*"autoMappingUnknownColumnBehavior"* value=*"WARNING"*/>

<setting name=*"defaultExecutorType"* value=*"SIMPLE"*/>

<setting name=*"defaultStatementTimeout"* value=*"25"*/>

<setting name=*"defaultFetchSize"* value=*"100"*/>

<setting name=*"safeRowBoundsEnabled"* value=*"false"*/>

<setting name=*"mapUnderscoreToCamelCase"* value=*"false"*/>

<setting name=*"localCacheScope"* value=*"SESSION"*/>

<setting name=*"jdbcTypeForNull"* value=*"OTHER"*/>

<setting name=*"lazyLoadTriggerMethods"*

value=*"equals,clone,hashCode,toString"*/>

</settings>

提示：暂时不需要掌握所有配置，先了解

### TypeAliases 设置

类型别名是为 Java 类型设置一个短的名字。它只和 XML 配置有关，存在的意义仅在于用来减少类完全限定名的冗余。例如:

<typeAliases>

<typeAlias alias="Author" type="domain.blog.Author"/>

<typeAlias alias="Blog" type="domain.blog.Blog"/>

<typeAlias alias="Comment" type="domain.blog.Comment"/>

<typeAlias alias="Post" type="domain.blog.Post"/>

<typeAlias alias="Section" type="domain.blog.Section"/>

<typeAlias alias="Tag" type="domain.blog.Tag"/>

</typeAliases>

例如：

当这样配置时，Blog可以用在任何使用domain.blog.Blog的地方。

也可以指定一个包名，MyBatis 会在包名下面搜索需要的 Java Bean。

MyBatis 为许多常见的Java类型内置类相应的类别名，例如

|  |  |
| --- | --- |
| 别名 | 映射类型 |
| iterator | Iterator |
| collection | Collection |
| arraylist | ArrayList |
| list | List |
| hashmap | HashMap |
| map | Map |
| bigdecimal | BigDecimal |
| decimal | BigDecimal |
| date | Date |
| boolean | Boolean |
| float | Float |
| double | Double |
| integer | Integer |
| int | Integer |
| short | Short |
| long | Long |
| byte | Byte |
| string | String |
| boolean | boolean |
| \_float | float |
| \_double | double |
| \_int | int |
| \_short | short |
| \_long | long |
| \_byte | byte |

## 核心操作（Operator）

### 定义实体对象

**public** **class** SysUser **implements** Serializable{

**private** **static** **final** **long** ***serialVersionUID*** = -9202121822896212030L;

**private** Integer id;

**private** String username;

**private** String password;

**private** String phone;

**private** Date createdDate;

**private** Date modifiedDate;

//set,get

}

### 定义mapper文件

定义查询操作

<select id=*"findUsers"* resultType=*"message"*>

select \* from sys\_users

</select>

定义写入字符串，存在于tomat容器中

<insert id=*"insertObject"*

parameterType=*"message"*>

insert into (username,password,phone,

createdTime,modifiedType)

values(null,'AA','123','139',now(),now())

</insert>

更新记录信息

<update id=*"updateObject"*

parameterType=*"sysUser"*>

update

sys\_users

set

username = #{username},

password = #{password},

phone = #{phone},

modifiedTime = NOW(),

modifiedUser =now ()

where

id = #{id}

</update>

删除记录信息

<delete id=*"deleteObject"*

parameterType=*"integer"*>

delete from sys\_users where id=#{uid}

</delete>

## 编写单元测试（junit）