MyBatis

第一讲
MyBatis框架概述

目录

MyBatis介绍

- MyBatis原本是apache的一个开源项目iBatis
- 2010年该项目由apache software foundation迁移到了Google code，并且改名为MyBatis
- 2013年11月迁移到GitHub
- MyBatis是一款持久化框架，它支持自定义SQL查询、存储过程以及高级映射

MyBatis介绍

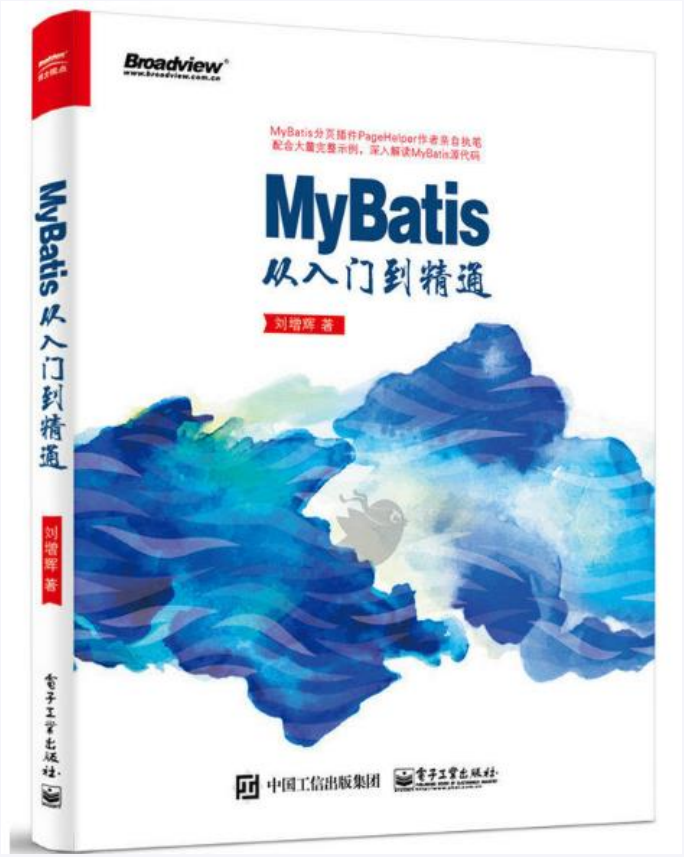
- 与传统的 JDBC 开发相比， MyBatis 消除了几乎所有的代码和参数的手工设置
- MyBatis是一个ORM框架
- MyBatis可以使用 XML 或注解方式进行配置和映射，它是把实体类和SQL语句之间建立了映射关系，而Hibernate是在实体类和数据库表之间建立了映射关系。

MyBatis介绍

- GitHub源码：
  - https://github.com/mybatis/mybatis-3
- MyBatis中文手册：
  - http://www.mybatis.org/mybatis-3/zh/index.html

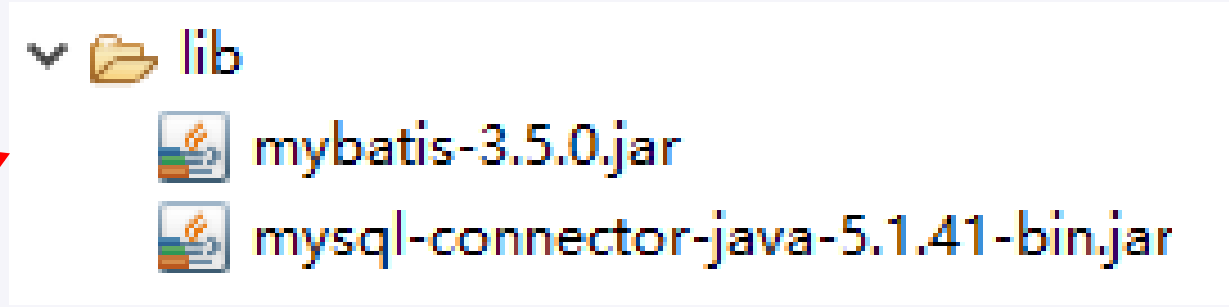
MyBatis介绍

- 参考书籍



MyBatis介绍—使用MyBatis流程

- 创建一个Java Project
- 导入MyBatis需要的jar包



- 创建MyBatis的主配置文件mybatis.xml
- 创建实体类和映射器接口
- 创建MyBatis的SQL映射XML文件
- 将SQL映射文件与主配置文件进行关联
- 编写代码进行测试

目录

MyBatis核心XML配置文件

- XML 配置文件（configuration XML）中包含了对 MyBatis 系统的核心设置，包含获取数据库连接实例的数据源（DataSource）和决定事务作用域和控制方式的事务管理器（TransactionManager）

MyBatis核心XML配置文件

```
<?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>
    <settings>
        <!-- 打印SQL语句 -->
        <setting name="logImpl" value="STDOUT_LOGGING" />
    </settings>
    <environments default="development">
        <environment id="development">
            <!-- 事务和数据源 -->
        </environment>
    </environments>
</configuration>
```

MyBatis核心XML配置文件

- settings是 MyBatis 中极为重要的调整设置，可以用来配置全局参数，它们会改变 MyBatis 的运行时行为
  - logImpl设置用来指定 MyBatis 所用日志的具体实现，未指定时将自动查找。如果设置为**STDOUT_LOGGING**则表示在控制台显示SQL语句

MyBatis核心XML配置文件

- 配置环境（environments）
  - MyBatis 可以配置成适应多种环境，也就是配置多个environment子元素，这种机制有助于将 SQL 映射应用于多种数据库之中
  - **注意：**environments的default属性取值要和其中一个environment的id属性取值一致

MyBatis核心XML配置文件

```
<environment id="development">
    <transactionManager type="JDBC"/>

    <dataSource type="POOLED">
        <property name="driver" value="com.mysql.jdbc.Driver"/>
        <property name="url" value=
    "jdbc:mysql://localhost:3306/mybatis?characterEncoding=utf-8"/>
        <property name="username" value="root"/>
        <property name="password" value=""/>
    </dataSource>
</environment>
```

MyBatis核心XML配置文件

- transactionManager 元素的type属性表示事务管理器类型，在MyBatis中有两种类型：
  - **JDBC** – 这种方式是直接使用了JDBC的事务提交和回滚设置
  - MANAGED(托管) –这种方式从来不提交或回滚一个连接。而是让容器来管理事务的整个生命周期(比如 Spring)

MyBatis核心XML配置文件

- dataSource元素中主要配置了 JDBC 连接对象的资源，它的type属性表示数据源类型，内建的数据源类型有三种：
  - UNPOOLED：每次被请求时简单打开和关闭连接
  - POOLED：这是JDBC连接对象的数据库连接池的实现，用来避免创建新的连接实例
  - JNDI：这个数据源的实现是为了使用如Spring或应用服务器这类的容器

MyBatis核心XML配置文件

- dataSource 元素的property子元素中配置了具体的数据库连接信息
  - driver – 是 JDBC 驱动的 Java 类的完全限定名
  - url – 是数据库的 JDBC URL 地址
  - username – 登录数据库的用户名
  - password – 登录数据库的密码

MyBatis映射文件

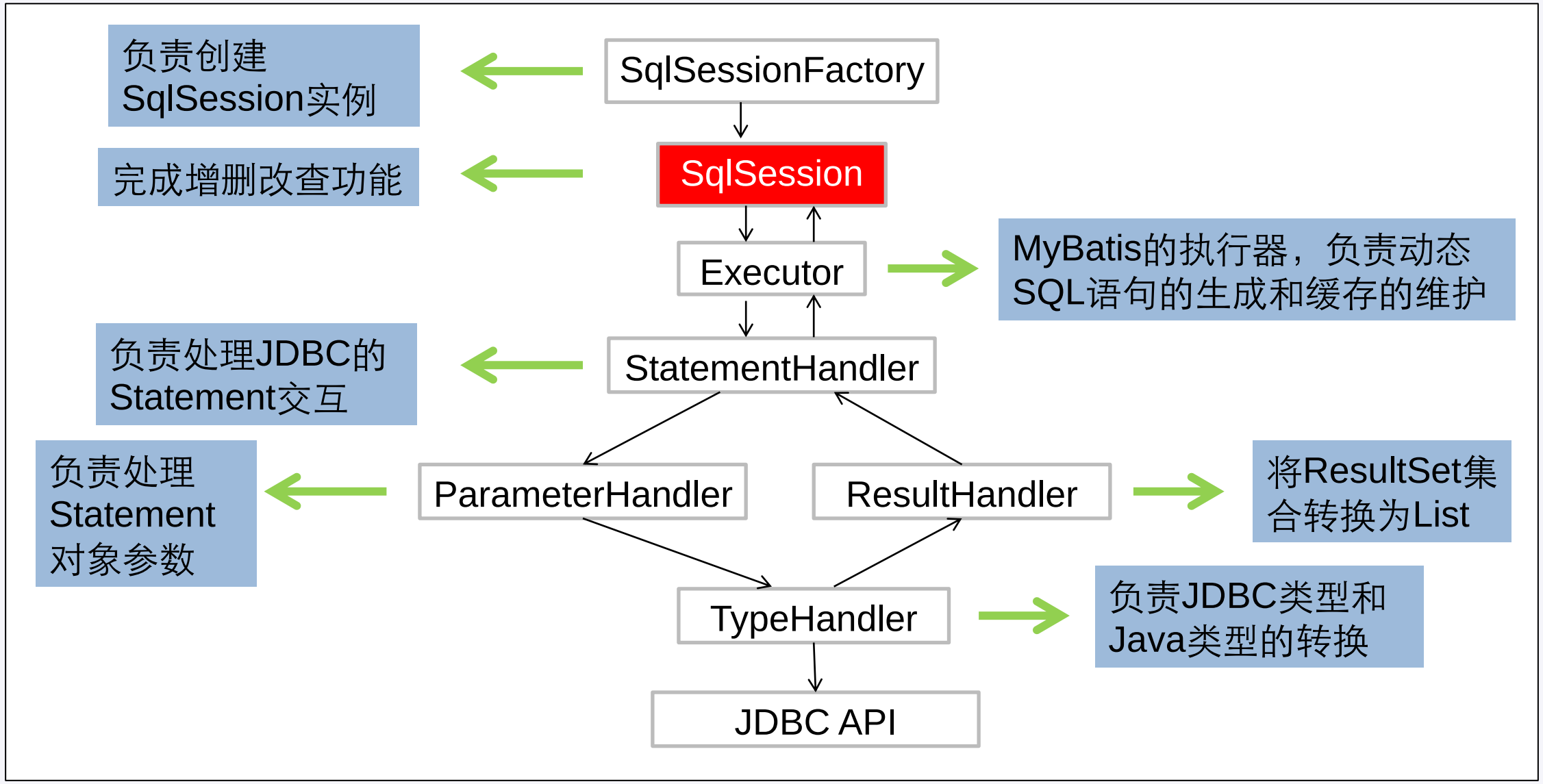
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
    <!-- namespace代表唯一标识符 -->
<mapper namespace="com.mybatis.mapper.UserMapper">
    <select id="selectAllUsers"
            resultType="com.mybatis.entity.User">
        select * from user
    </select>
</mapper>
```

MyBatis映射文件

- 映射文件的根元素为mapper，它的namespace属性为映射器接口的完全限定名，以下子元素的id属性值为接口中的某个方法名称
  - insert – 映射插入语句
  - update – 映射更新语句
  - delete – 映射删除语句
  - select – 映射查询语句

目录

MyBatis主要的类层次结构



MyBatis主要的类层次结构

- 每一个MyBatis的应用程序都以一个SqlSessionFactory对象的实例为核心 。
- SqlSessionFactory对象的实例可以通过SqlSessionFactoryBuilder对象来获得 。
- SqlSessionFactoryBuilder对象可以从 XML 配置文件中构建 SqlSessionFactory对象。

MyBatis主要的类层次结构

- 通过SqlSessionFactory对象，可以获得SqlSession的实例。
- SqlSession 对象完全包含以数据库为背景的所有执行 SQL 操作的方法。你可以用 SqlSession 实例来直接执行已映射的 SQL 语句

MyBatis主要的类层次结构

```
public class MyBatisUtil {
    private static SqlSessionFactory factory;
    static {
        try {
            InputStream is =
                Resources.getResourceAsStream("mybatis.xml");
            factory = new SqlSessionFactoryBuilder().build(is);
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public static SqlSession openSqlSession() {
        return factory.openSession();
    }
}
```

目录

第一个MyBatis程序

- 创建一个Java Project
- 导入MyBatis需要的jar包
- 创建实体类和映射器接口
- 创建MyBatis的主配置文件mybatis.xml
- 创建MyBatis的SQL映射XML文件
- 将SQL映射文件与主配置文件进行关联
- 编写代码进行测试

第一个MyBatis程序

- 创建实体类User和映射器接口

```
//实体类
public class User {

    private Integer id;
    private String userName;
    private String password;
    //省略构造方法
    //省略getter、setter方法
}
```

```
//映射器接口
public interface UserMapper {

    List<User> selectAllUsers();

}
```

第一个MyBatis程序

- MyBatis中使用Mapper接口的具体要求
  - Mapper接口的全限定名为映射文件的namespace的值
  - Mapper接口的方法名称和映射文件中定义的每个sql的id相同
  - Mapper接口的方法参数和映射文件中定义每个sql的parameterType类型相同
  - Mapper接口的方法返回的单个对象类型和映射文件中定义的每个sql的resultType类型相同

第一个MyBatis程序

- 创建MyBatis的主配置文件mybatis.xml
  - 在工程中新建Source Folder命名为resources来专门存放MyBatis的主配置文件，在resources中新建XML File命名为mybatis.xml

第一个MyBatis程序

- 创建MyBatis的SQL映射XML文件，并且映射文件要同刚才创建的映射器接口在同一包中，与映射器接口命名也相同，所以映射文件名为UserMapper.xml

- 注意：要将映射文件关联到主配置文件

第一个MyBatis程序

- 将SQL映射文件与主配置文件进行关联

```
<?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">
        <!-- 事务和数据源 -->
    </environments>
    <mappers>
        <mapper resource="org/mybatis/example/BlogMapper.xml"/>
    </mappers>
</configuration>
```

第一个MyBatis程序

- 映射器（mappers）用来配置多个映射文件的位置， 有多种配置方式。

```
<!-- 使用相对于类路径的资源引用 -->
<mapper resource="org/mybatis/builder/AuthorMapper.xml"/>

<!-- 使用映射器接口的完全限定类名 -->
<mapper class="org.mybatis.builder.AuthorMapper"/>

<!-- 将包内的映射器接口全部注册为映射器 -->
<package name="org.mybatis.builder"/>
```

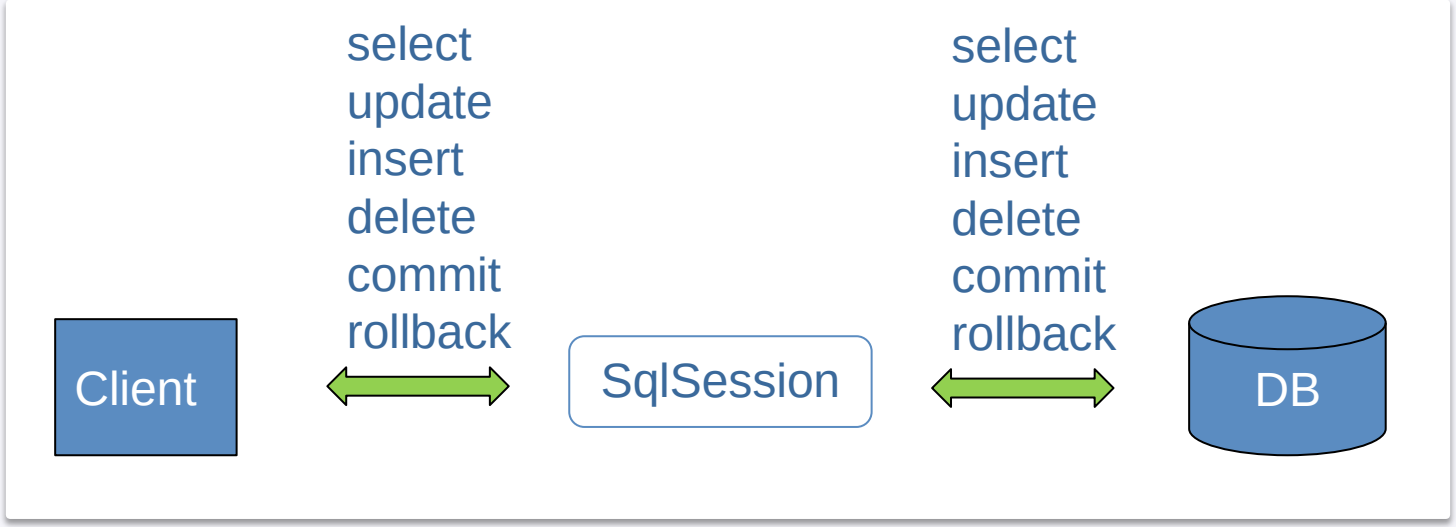
第一个MyBatis程序

- 编写代码进行测试（第一种方式）

```
SqlSession session = MyBatisUtil.openSqlSession();
List<User> users =
session.selectList("com.mybatis.mapper.UserMapper.selectAllUsers");
for(User u : users) {
    System.out.println(u);
}
session.close();
```

SqlSession的工作流程

- **1. 开启一个数据库访问会话---创建SqlSession对象：**
- MyBatis封装了对数据库的访问，把对数据库的会话和事务控制放到了SqlSession对象中。

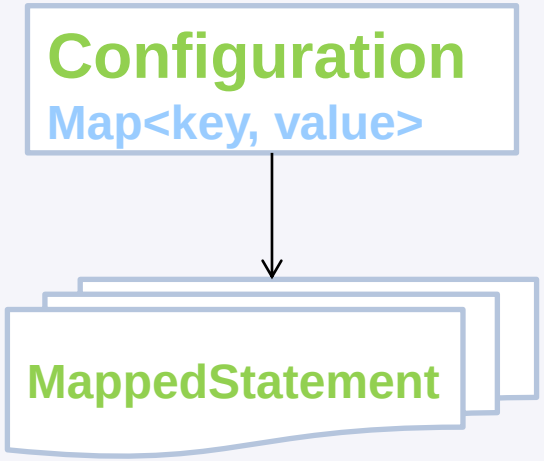


SqlSession的工作流程

- **2.为SqlSession传递一个映射的SQL语句的Statement Id和参数，然后返回结果：**
  - SqlSession根据Statement ID, 在MyBatis配置对象Configuration中获取到对应的MappedStatement对象
  - 调用MyBatis执行器来执行具体的操作。

SqlSession的工作流程

- MyBatis在初始化的时候，会将MyBatis的配置信息全部加载到内存中，使用Configuration实例来维护。



- 映射文件加载到内存中会生成n个对应的MappedStatement对象
  - key="net.onest.mapper.UserMapper.selectAllUsers" ，value为MappedStatement对象的形式维护到Configuration的一个Map类型的属性中。

第一个MyBatis程序

- 编写代码进行测试（第二种方式）

```
SqlSession session = MyBatisUtil.openSqlSession();
UserMapper userMapper = session.getMapper(UserMapper.class);
List<User> users = userMapper.selectAllUsers();
for(User u : users) {
    System.out.println(u);
}
session.close();
```

本章小结

- MyBatis介绍
- MyBatis核心XML配置文件
  - 主配置文件
  - 映射配置文件
- MyBatis的API的使用

THANK YOU