1 概述

特点

- MyBatis是一款持久层框架，支持自定义SQL、存储过程、高级映射。
- MyBatis消除了大部分JDBC代码，消除了大部分设置参数和处理结果集的代码。
- MyBatis可以通过xml或注解来映射原始类型、接口、POJO。

为什么使用MyBatis

- jdbc：过程: 预编译、设置参数、执行、封装执行结果
- Dbutils：工具，只执行(查询)，不封装执行结果
- Hibernate：全自动的ORM（对象关系映射）框架，无需手写sql，所以无法优化sql
- MyBatis：半自动ORM框架，手写sq1， 自动封装执行结果

2 使用

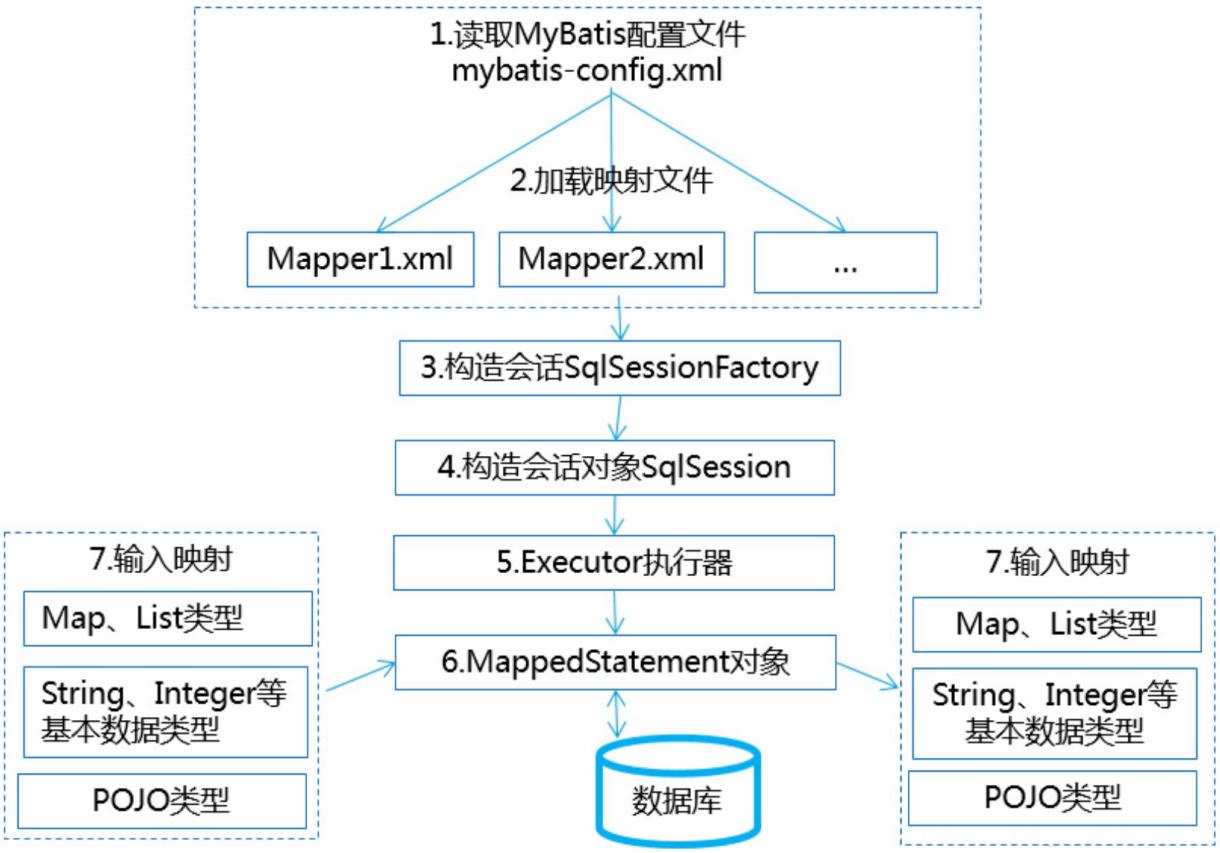
2.1 配置文件

- 导入包 `MyBatis` 、 `MysqlDriver`
- 创建数据库、表
- 创建持久化类 POJO
- 创建映射文件 `Mapper.xml` ，在其中编写 SQL 语句
- 创建 `MyBatis` 核心配置文件 `MyBatisConfig.xml` 配置数据库连接

2.2 操作过程

- 读取配置文件
- 根据配置文件构建 `SqlSessionFactory`
- 通过 `SqlSessionFactory` 创建 SqlSession
- 使用 SqlSession 对象实现增删改查
- 关闭 SqlSession

2.3 工作原理



3 核心对象

3.1 SqlSessionFactory

- 单个数据库映射关系经过编译后的内存镜像
- 用来创建 `SqlSession`
- 可以由 `SqlSessionFactoryBuilder` 通过 `xml` 配置构建出 `SqlSessionFactory` 实例
- `SqlSessionFactory` 对象是数据安全的，整个应用的执行期间都存在

3.2 SqlSession

- 是应用程序与持久层执行交互操作的单线程对象
- 用来执行持久化操作
- `SqlSession` 是非线程安全的，每个线程有自己的实例
- 使用完成后及时关闭

4 CURD

4.1 update

- `#{}` 单个参数时候可以随便取值
- 多个 `#{}` 参数需要使参数名对应 `POJO` 中 get 方法的小写名

```
<update id="update1">
  update Spring.Users set uname = #{uname}, usex = #{usex}, money = #{money}
  where uid = #{uid}
</update>
```

4.2 delete

```
<delete id="delete1">
delete from Spring.Users where uid = #{uid}
</delete>
```

4.3 insert

```
<insert id="insert1">
  insert into Spring.Users (uname, usex, money)
  values (#{uname}, #{usex}, #{money})
</insert>
```

4.4 select

```
<select id = "queryByName" resultType="com.bean.User">
select * from Spring.Users where uname = #{name}
</select>
```