1、什么是框架

框架在表现形式上，就是别人写好的一堆class字节码文件，这些字节码文件通常被打包成jar包（压缩包），使用框架只需要将框架的jar包引入到classpath环境变量中即可（这是针对不用集成开发工具的情况，在集成开发工具中在项目中导入框架的jar包即可）

使用框架开发是为了提高开发效率，就是很多开发中通用且繁琐的的程序被提前封装好了，我们直接拿来用。

JavaEE的框架很多：Struts、Struts2、Spring、SpringMVC、MyBatis、Hibernate等等

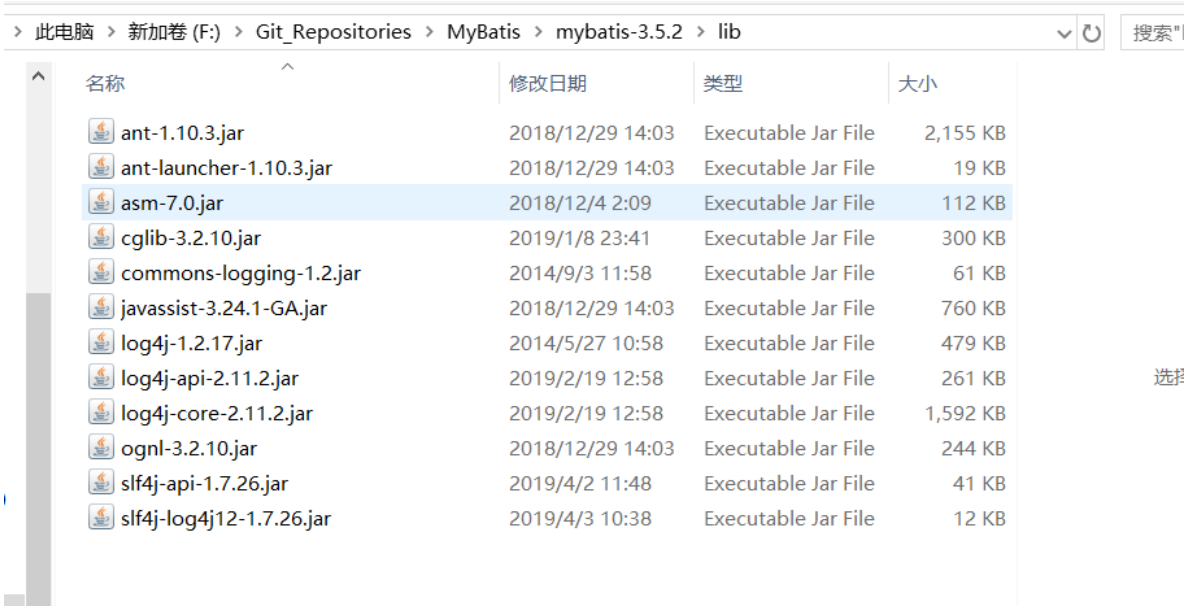
所有的框架都是基于反射机制和xml配置一起完成的

MyBatis和Hibernate都是持久层框架，属于DAO层。专门封装JDBC，以简化编程提高效率。

MyBatis之前称作：iBatis。这里用的MyBatis版本：MyBatis3.5.2

我们点进下载好解压缩的MyBatis3.5.2，如下图

可以看到其中的.jar结尾的就是我们开发中所需要的MyBatis框架jar包，是要导入或者添加到classpath中的。而.pdf结尾的是其官方帮助文档。lib目录中也是jar包，在MyBatis这个框架需要使用的依赖jar包，如下图



比如说要在MyBatis中记录日志，就可以引入log4j-1.2.17.jar

从官方文档学习MyBatis，在百度搜索Mybatis3中文文档，有在线版的。根据文档学习以及实现其中的例子，是一个程序员学习新技术所必须的。

1）这里我们创建一个普通的java工程b-mybatis-001

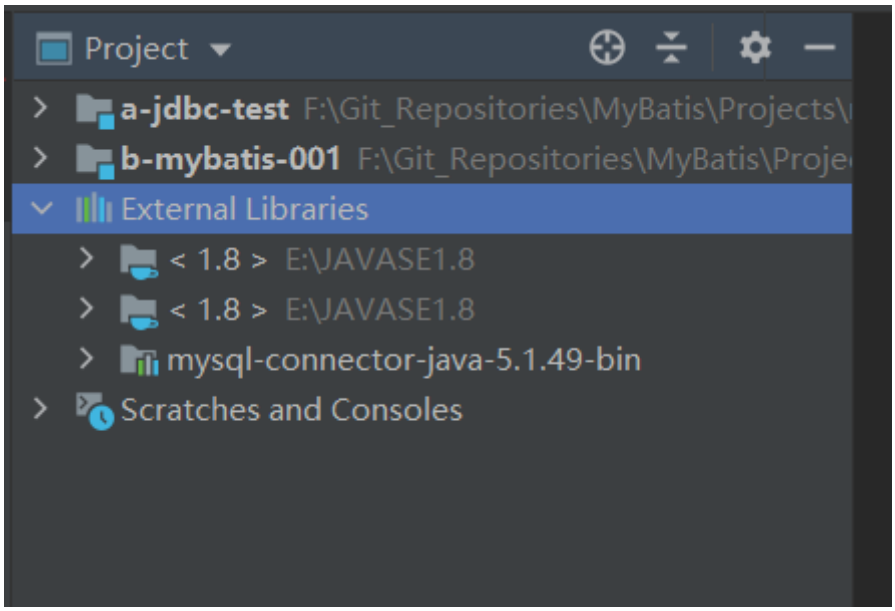
--不需要创建web工程，在web工程中MyBatis是在DAO层起作用的，但是现在还不需要。MyBatis封装了JDBC，写一个main方法即可测试。

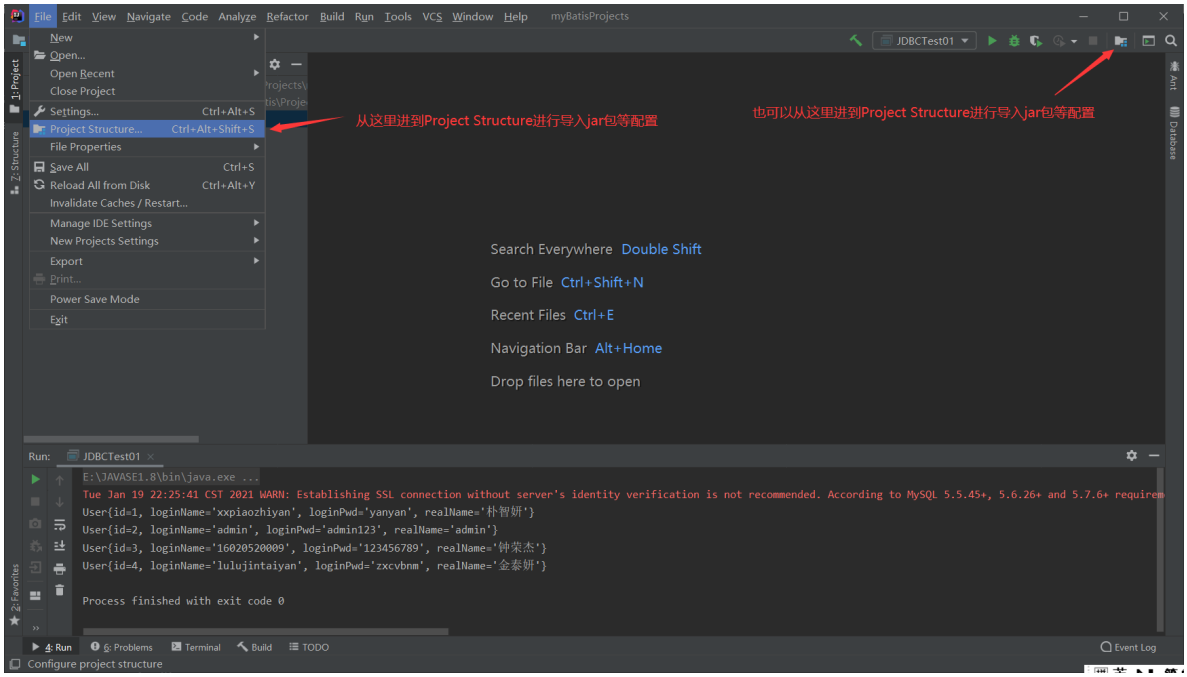
这里使用的是idea集成开发工具，引入的jar包保存在External Libraries目录中，

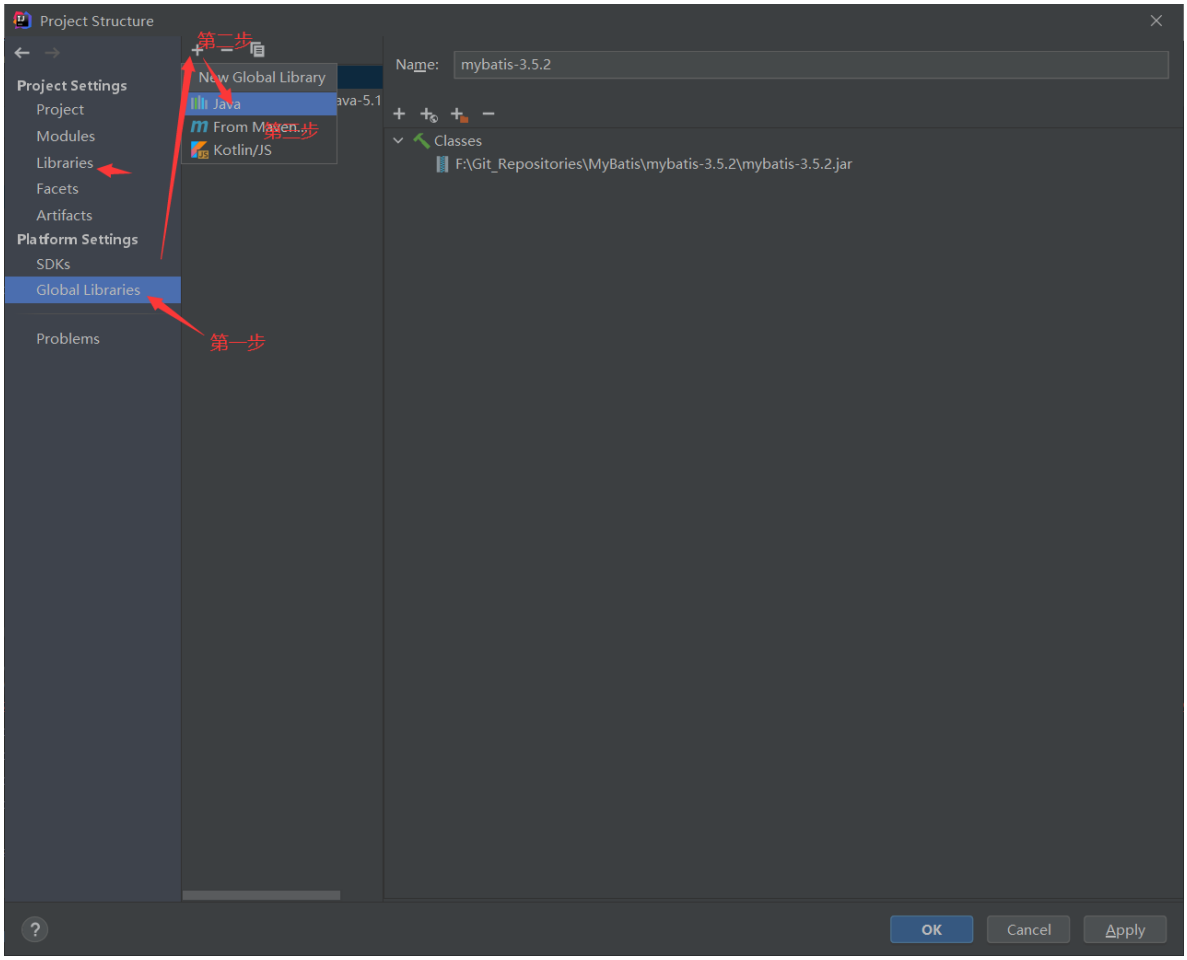
可以进入到Project Structure中进行引入，里面有Project Settings（对当前项目配置）和Plateform Settings（对整个idea配置），我一般引入jar包是使用对整个idea的，

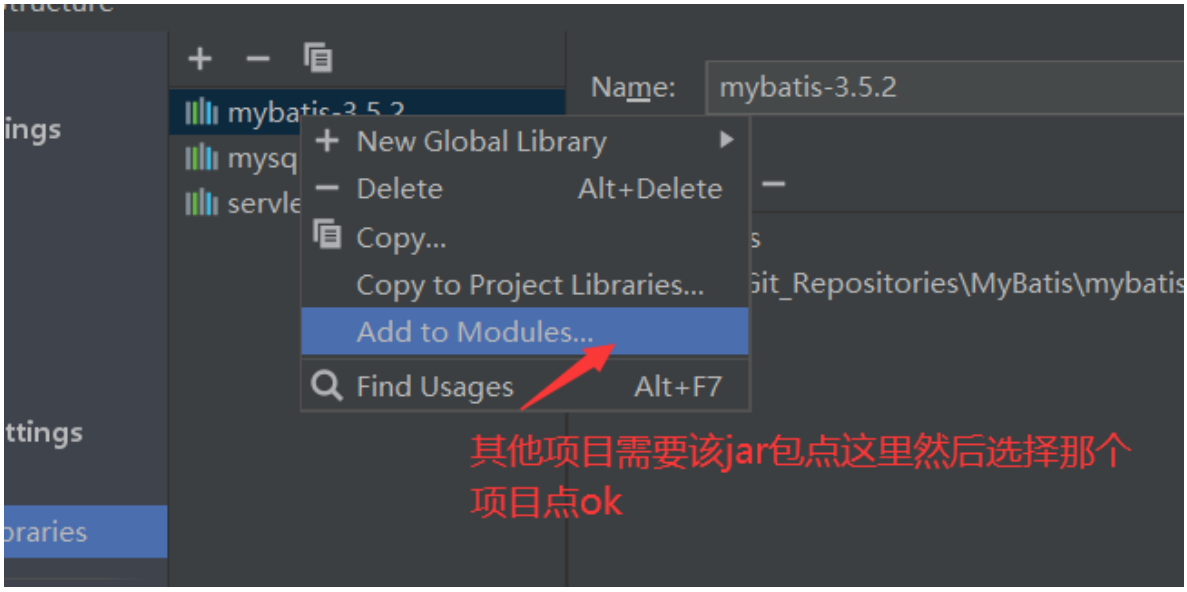
所以选中Global Libraries->点击+号->点击java->然后找到jar包路径（这里是MyBatis的jar包），然后选中右键（一般第一次引入jar包会让你选择生效的工程）点击Add To Modules，选择需要的项目即可。

在idea的全局依赖中添加的jar包的好处是以后创建项目可以直接右键添加到Moduls中，快速。截图如下




File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help myBatisProjects
New
Open...
Open Recent
Close Project
Settings... Ctrl+Alt+S
Project Structure... Ctrl+Alt+Shift+S
File Properties
Save All Ctrl+S
Reload All from Disk Ctrl+Alt+Y
Invalidate Caches / Restart...
Manage IDE Settings
New Projects Settings
Export
Print...
Power Save Mode
Exit
从这里进到Project Structure进行导入jar包等配置
也可以从这里进到Project Structure进行导入jar包等配置
Search Everywhere Double Shift
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here to open
Run: JDBCTest01
E:\JAVASE1.8\bin\java.exe ...
Tue Jan 19 22:25:41 CST 2021 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirem
User{id=1, loginName='xxpiaozhiyan', loginPwd='yanyan', realName='朴智妍'}
User{id=2, loginName='admin', loginPwd='admin123', realName='admin'}
User{id=3, loginName='16020520009', loginPwd='123456789', realName='钟荣杰'}
User{id=4, loginName='lulujintaiyan', loginPwd='zxcvbnm', realName='金泰妍'}
Process finished with exit code 0
Run Problems Terminal Build TODO Event Log
Configure project structure


Project Structure
第二步
第二步
Project Settings
Project
Modules
Libraries
Facets
Artifacts
Platform Settings
SDKs
Global Libraries
Problems
第一步
New Global Library
Java
From Maven...
Kotlin/JS
Name: mybatis-3.5.2
Classes
F:\Git_Repositories\MyBatis\mybatis-3.5.2\mybatis-3.5.2.jar
OK
Cancel
Apply



2）这里引入相关jar包mybatis和数据库驱动包。

按照文档中的来，但是我们这里创建的不是Maven工程，所以忽略其中的Maven工程步骤。

3）（官方原话）从xml文件中构建SqlSessionFactory

```
--上面的描述信息所能提取到的信息
    - mybatis中一定要有一个xml文件
    - mybatis中一定有一个类叫做：SqlSessionFactory
    -SqlSessionFactory的对象可以通过依赖xml文件来创建
官方给出的代码如下：
```

从 XML 中构建 SqlSessionFactory

每个基于 MyBatis 的应用都是以一个 SqlSessionFactory 的实例为核心的。SqlSessionFactory 的实例可以通过 SqlSessionFactoryBuilder 获得。而 SqlSessionFactoryBuilder 则可以从 XML 配置文件或一个预先配置的 Configuration 实例来构建出 SqlSessionFactory 实例。

从 XML 文件中构建 SqlSessionFactory 的实例非常简单，建议使用类路径下的资源文件进行配置。 但也可以使用任意的输入流（InputStream）实例，比如用文件路径字符串或 file:// URL 构造的输入流。MyBatis 包含一个名叫 Resources 的工具类，它包含一些实用方法，使得从类路径或其它位置加载资源文件更加容易。

```
String resource = "org/mybatis/example/mybatis-config.xml";
InputStream inputStream = Resources.getResourceAsStream(resource);
SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);
```

```
//里面的三行代码我们放到main方法中，这里新建一个com.mybatis.test.MyBatisTest01类，代码如下：
//需要注意的是这里的包是idea自己导入的，都是ibatis包，是正确的。有时类名重复而包不同需要我们一个一个去试
package com.mybatis.test;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import java.io.IOException;
import java.io.InputStream;

public class MyBatisTest01 {

    public static void main(String[] args) {

        try {
```

```
            String resource = "org/mybatis/example/mybatis-config.xml";
            InputStream inputStream = Resources.getResourceAsStream(resource);
            SqlSessionFactory sqlSessionFactory = new
SqlSessionFactoryBuilder().build(inputStream);

        } catch (IOException e) {
            e.printStackTrace();
        }

    }

}
```

```
/*解读提供的三行代码
* 第一行代码中给了这样一个路径:"org/mybatis/example/mybatis-config.xml"
    1、可以知道mybatis中有一个配置文件叫：mybatis-config.xml（是核心配置文件）
    2、这个路径开头没有/也不是绝对路径，可知mybatis-config.xml这个配置文件存放到的是该项目的
类的根路径下的，
    在IDE中开发类的根路径是在src目录下，实质上从硬盘文件中打开项目存到的是bin目录，因为项目运
行需要的是class
    文件，而不是java文件，src存放的是java文件。但是开发中这两个目录可以等同看成同一个路径。
mybatis默认从类的
    根路径下为起点查找资源，所以我们在src目录下创建一个mybatis-config.xml文件
    String resource = "org/mybatis/example/mybatis-config.xml";
    改为
    String resource = "mybatis-config.xml";
    修改后的main方法代码如下：
*/
  public static void main(String[] args) {

        try {

            String resource = "mybatis-config.xml";
            InputStream inputStream = Resources.getResourceAsStream(resource);
            SqlSessionFactory sqlSessionFactory = new
SqlSessionFactoryBuilder().build(inputStream);

        } catch (IOException e) {
            e.printStackTrace();
        }

    }
```

```
编辑mybatis-config.xml配置文件，文档中相关代码直接复制到xml文件中，如下以及截图
<?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="${driver}"/>
        <property name="url" value="${url}"/>
```

```
        <property name="username" value="${username}"/>
        <property name="password" value="${password}"/>
      </dataSource>
    </environment>
  </environments>
    <!--通过这里的配置可以看出mybatis还有另外一个配置文件，专门编写SQL语句-->
    <!--而且该配置文件的查找路径也是从类的根路径下开始查找-->
  <mappers>
    <mapper resource="org/mybatis/example/BlogMapper.xml"/>
  </mappers>
</configuration>
```

XML 配置文件中包含了对 MyBatis 系统的核心设置，包括获取数据库连接实例的数据源（DataSource）以及决定事务作用域和控制方式的事务管理器（TransactionManager）。后面会再探讨 XML 配置文件的详细内容，这里先给出一个简单的示例：

```
<?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="${driver}"/>
        <property name="url" value="${url}"/>
        <property name="username" value="${username}"/>
        <property name="password" value="${password}"/>
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="org/mybatis/example/BlogMapper.xml"/>
  </mappers>
</configuration>
```

当然，还有很多可以在 XML 文件中配置的选项，上面的示例仅罗列了最关键的部分。注意 XML 头部的声明，它用来验证 XML 文档的正确性。environment 元素体中包含了事务管理和连接池的配置。mappers 元素则包含了一组映射器（mapper），这些映射器的 XML 映射文件包含了 SQL 代码和映射定义信息。

<!--从中我们可以知道value中的值就是连接数据库所需要的，如数据库驱动类名，连接具体数据的URL，数据库用户名和密码
可以发现mapper标签中还有一个xml文件的路径，想象以下可以知晓该mapper.xml文件应该是配置SQL语句的文件，存放位置与mybatis核心配置文件一样。那么我们创建一个SqlMapper.xml文件

编写好的核心配置文件如下
    （核心配置文件主要配置连接数据库的信息，以及“SQL语句配置文件”的路径。-->

```
<?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://localhost:3306/mysqlstudy"/>
        <property name="username" value="root"/>
        <property name="password" value="rong195302"/>
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="SqlMapper.xml"/>
  </mappers>
</configuration>
```

官方SqlMapper.xml文件提供如下：

探究已映射的 SQL 语句

现在你可能很想知道 SqlSession 和 Mapper 到底具体执行了些什么操作，但 SQL 语句映射是个相当广泛的话题，可能会占去文档的大部分篇幅。 但为了让你能够了解个大概，这里会给出几个例子。

在上面提到的例子中，一个语句既可以通过 XML 定义，也可以通过注解定义。我们先看看 XML 定义语句的方式，事实上 MyBatis 提供的所有特性都可以利用基于 XML 的映射语言来实现，这使得 MyBatis 在过去的数年间得以流行。如果你用过旧版本的 MyBatis，你应该对这个概念比较熟悉。但相比于之前的版本，新版本改进了许多 XML 的配置，后面我们会提到这些改进。这里给出一个基于 XML 映射语句的示例，它应该可以满足上个示例中 SqlSession 的调用。

```
<?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="org.mybatis.example.BlogMapper">
  <select id="selectBlog" resultType="Blog">
    select * from Blog where id = #{id}
  </select>
</mapper>
```

```
<!--这里根据需求对 "sql映射文件" (SqlMapper.xml)进行修改和解释-->


<?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">
<!--这里先不要管namespace的值，随便写-->
<mapper namespace="asssdd">
    <!--编写查询语句，查询所有用户信息-->
    <!--id的值代表的是下面是一条sql语句，具有唯一性，id的值要拷贝到java程序中，尽量不要自己写，避免写错-->
    <!--我们得告诉mybatis框架需要自动创建什么java对象封装查询出的数据以及自动将查询的结果集放到java对象的对应属性上-->
    <!--resultType的值就是让mybatis自动创建的对象的完整类名-->
    <!--让mybatis框架自动将查询的结果集放到java对象的对应属性上，要保证查询出的结果的展示字段与Java对象中属性字段一致，如果Java对象的字段名和数据库表中中的字段名不一致，查询sql语句需要起别名为Java对象中的对应字段名，若没有起别名，也就没法将查询结果集放到Java对象中，该对象的属性字段就为null-->
  <select id="getAll" resultType="com.mybatis.entity.User">
    select id as userid,loginName as username,loginPwd as password from t_user
  </select>
</mapper>
```

根据官方文档，我们也可以不使用xml配置文件来创建SqlSessionFactory，自己有时间了解。

4）（官方原话）从 SqlSessionFactory 中获取 SqlSession。

这是关于mybatis的核心API：SqlSession对象

从 SqlSessionFactory 中获取 SqlSession

既然有了 SqlSessionFactory，顾名思义，我们可以从中获得 SqlSession 的实例。SqlSession 提供了在数据库执行 SQL 命令所需的所有方法。你可以通过 SqlSession 实例来直接执行已映射的 SQL 语句。例如：

```
try (SqlSession session = sqlSessionFactory.openSession()) {
  Blog blog = (Blog) session.selectOne("org.mybatis.example.BlogMapper.selectBlog", 101);
}
```

```
/*
```

在main方法中，我们先获取mybatis核心配置文件，用字节码输入流打开通道将核心配置文件中获取数据库驱动以及数据库连接对象等信息流进SqlSessionFactoryBuilder对象的build方法中，然后返回一个SqlSessionFactory对象，通过SqlSessionFactory对象中的openSession方法再获取到SqlSession对象，每一步的作用如下

```
*/
package com.mybatis.test;

import com.mybatis.entity.User;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import java.io.IOException;
import java.io.InputStream;
import java.util.List;

public class MyBatisTest01 {

    public static void main(String[] args) {

        SqlSession sqlSession = null;
        List<User> userList = null;

        try {

            String resource = "mybatis-config.xml";
            InputStream inputStream = Resources.getResourceAsStream(resource);
            SqlSessionFactory sqlSessionFactory = new
SqlSessionFactoryBuilder().build(inputStream);

            //执行下面语句，事务自动提交就关闭了，等同于conn.setAutoCommit(false);
            //然后开启事务
            //SqlSession对象等同看作Connection，是专门用来执行SQL语句的对象
            sqlSession = sqlSessionFactory.openSession();

            //do work(执行核心业务逻辑)
            //获取所有用户信息，返回List集合，集合中存储User
            //sqlSession.selectList()中填的是sql映射文件某条sql语句的唯一标识id的字符串
            userList = sqlSession.selectList("getAll");

            //若没有出现异常，事务结束，提交
            sqlSession.commit();

        } catch (IOException e) {
            //遇到异常，回滚事务,所以sqlSession对象要在try catch外面定义
            if (sqlSession != null) {
                sqlSession.rollback();
            }
            e.printStackTrace();
        }finally {
            //关闭资源
            if (sqlSession != null) {
                sqlSession.close();
            }
        }

        for (User user:
```

```
            userList) {
            System.out.println(user);
        }

    }

}
```

```
//要实现查询所有用户，需要提前创建用户的javabean，即用户的实体类，如下：



package com.mybatis.entity;

public class User {

    //注意属性字段名与要查询的表t_user中的字段名不一致
    private int userid;
    private String username;
    private String password;

    public User() {
    }

    public int getUserid() {
        return userid;
    }

    public void setUserid(int userid) {
        this.userid = userid;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    @Override
    public String toString() {
        return "User{" +
                "userid=" + userid +
                ", username='" + username + '\'' +
                ", password='" + password + '\'' +
                '}';
    }
}
```

至此一个用mybatis框架进行连接数据库进行查询的简单用例就完成了。

出现的问题就是数据库驱动忘记导包了，还有就是sql映射文件最上面空了一格报异常。

思考：是否需要在项目目录下新建一个lib目录?

我们用idea创建的项目，创建java项目有两种方式，

第一种是直接创建一个java工程，该Java工程的名字就是该项目的名字（即一个Java工程就是一个Java项目），从硬盘上点击项目名打开，里面包含.idea目录、src目录（保存java文件以及其他依赖文件和配置文件等），还有一个out目录（就是所熟知的bin目录），其中保存的是class字节码文件以及其他依赖文件和配置文件等，我们运行项目用的是out目录中的文件。

第二种是先创建一个空工程，然后添加模块，可以选择Java项目，这样创建的工程包含多个模块，模块就不一定是Java项目了，但是几乎不可能在一个工程中的模块是分不同类型的项目。这样我们在硬盘中找到该工程，点击打开，里面是.idea目录、各模块（项目）名的目录、out目录。打开某个模块的目录里面是src，打开out目录里面是production目录（说明是项目产品），打开，里面各模块名字目录，里面保存的是各模块的class文件。

上面两种方式中并没有将开发时导入的jar报存到硬盘的工程目录中，我们打包项目给客户就没有项目运行所依赖的jar包，所以需要我们手动在工程或项目目录下新建lib目录将jar包复制粘贴进去

整个项目目录结构截图如下