# JavaWeb

## 数据库(MySQL)

Double数据类型由两个参数：score double(总长度, 小数点后保留的位数（1 ~ 100）)

例如分数满分为100.00 就可以 score double(5, 2)

Char 和 varchar ， char的存储性能会高一点(因为varchar要计算字符串的长度之后再分配)，varchar节省空间



### 语句

操作表(DDL) ：

修改表名：alter table 表名 rename to 新表名

添加表项：alter table 表名 add 列名 数据类型

修改数据类型 ： alter table 表名 modify 列明 新数据类型

修改列名和数据类型 ： alter table 表名 change 列名 新列名 新数据类型

删除列 ： alter table 表名 drop 列名

添加成员： insert into 表名(列名1, 列名2, …) values(值1, 值2 …), (值1, 值2, …) -- **对所有列添加的时候可以不用写表名后面的东西，但是最好还是写上**。

修改数据： update 表名 set 列名1 = 值1, 列名2 = 值2 … [where 条件] –- **不加条件会把所有人的都修改**

删除数据： delete from 表名 [where 条件] –-- 不加条件会删除所有的数据

DQL：

查询语法：

Select

字段列表

From

表名列表

Where

条件列表 – between and, 布尔符号, in, is null – 不能用 != null, like \_单个字符占位符 %多个字符占位符, or -> ||, and -> &&, not -> !

Group By

分组字段

Having

分组后条件

Order by

排序字段

Limit

分页限定

基础查询： select 列名1, 列名2 … /\* from 表名;

去除重复的查询： select distinct列名1, 列名2 … /\* from 表名;

修改查询出的列名：select name as 名字 from 表名;

排序 Order by 列名 排序方式, 列名 排序方式, … 排序方式： asc升序 ， dese降序， 多个条件排序时，只有前面的相同后面的才会执行 default 为 asc

聚合函数：null 不参与聚合函数的运算

函数名：sum(列名)，count(列名)，min(列名), avg(列名), max(列名)

Count只能统计不为null的，建议使用\* 或者主键 。

分组查询：group by

分页查询：select 字段列表 from 表名 limit 起始索引 , 查询条目数 --起始索引 = （当前页码 - 1） \* 每页显示的条数 且每个数据库应用的语言不同 SQL Server 为top

### 约束



约束可以同时使用，中间不用加逗号 : name varchar(30) not null unique ,

自动增长(id等)： auto\_increment 约束从1开始的自动增长

添加外键约束：[constraint] [外键名称] foreign key(外键列名) reference 主表(主表列名) – constraint fk\_emp\_dept foreign key(dept\_id) reference dept(id) 两个表建立外键后有个主从关系

数据库关系：  
 1. 多对一：即建立一个外键即可，多的一方建立外键指向一的一方

2. 多对多：需要一个中间表，中间表与两个表之间都为多对一关系

3. 一对一：两个表任意一方建立外键，并且对外键设置unique的约束

多表查询：select 字段名 from 表一，表二 … -- 取出的数据是笛卡尔积条数 所以需要消除无效数据，

消除无效数据：后加where 将两张表中相同的字段进行比较

内连接：相当于两个表的交集

隐式内连接：select **表名.列名，表名.列名…**  from 表名1，表名2 … where [条件]

显式外连接：select **表名.列名，表名.列名…** from 表1 inner join 表2 on 条件

外连接：

左外连接：查询出A+A∩B的数据，与A表不同，查询出来的结果带有B表的数据

select **表名.列名，表名.列名…** from 表1 left outer join 表2 on 条件

右外连接：查询出B+A∩B的数据，与B表不同，查询出来的结果带有A表的数据

select **表名.列名，表名.列名…** from 表1 right outer join 表2 on 条件

子查询：

单行单列：select 字段 from 表 where 字段名 = (子查询(select … ))

多行单列：select 字段 from 表 where 字段名 in (子查询)

多行多列： select 字段 from 表，子查询(由子查询产生的虚拟表) where

### 事务

开始事务start transaction / begin

提交：commit

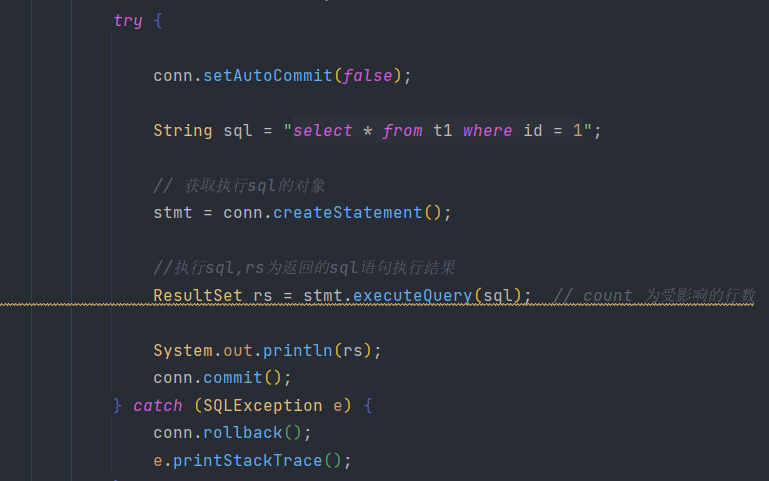
取消事务/回滚：rollbackz

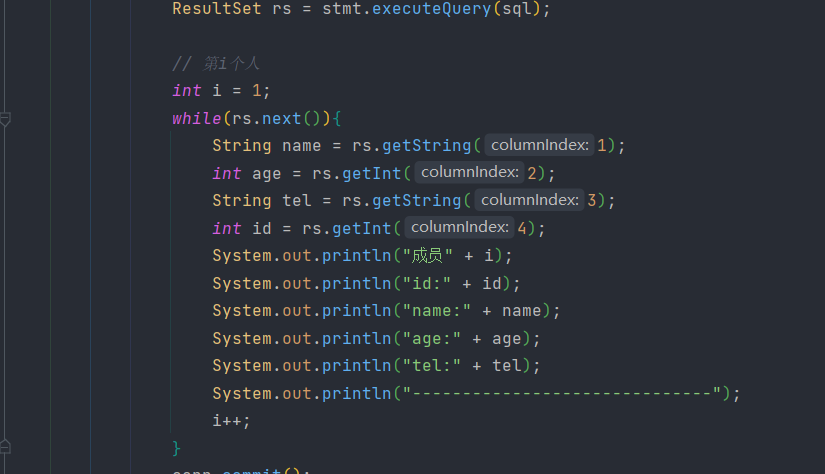
## JDBC(Java DataBase Connectivity)

步骤：  
 0, 导入jar包

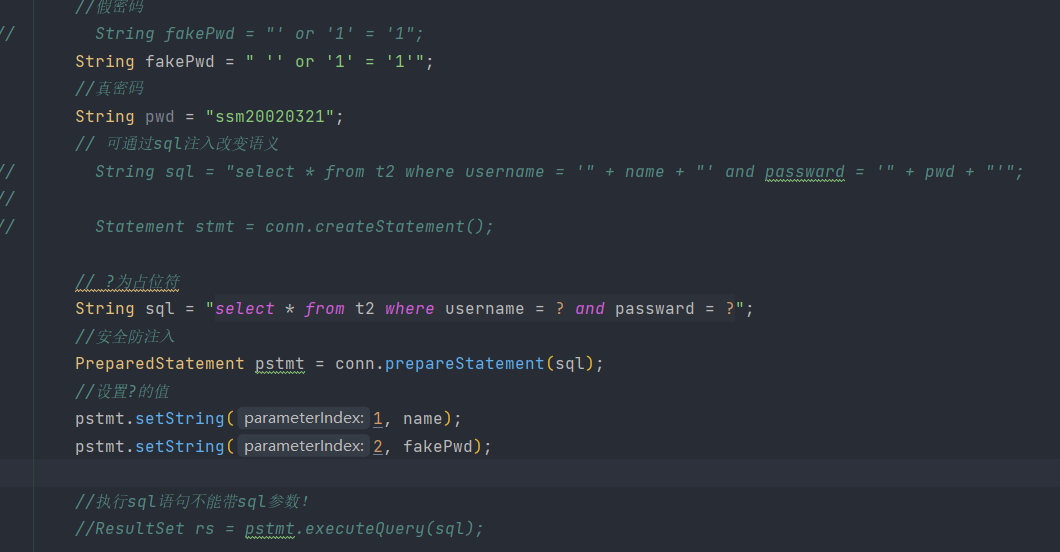
1. 注册驱动：
   1. Class.forName(“com.mysql.jdbc.Driver”); // 5之后可以不用写，库中自带
2. 获取链接：
   1. Connection conn = new Connection(url, username, passward);
   2. String url : “jdbc:mysql://127.0.1:3306/[对应数据库名称]” // 若是在本机链接且端口号为3306则可以简写为“jdbc:mysql:///[数据库名称]?参数键值对”即可，配置userSSL = false 参数，禁用安全连接方式，解决警告提示
3. 定义sql语句
   1. String sql = “update … ”
4. 获取sql对象
   1. Statement stmt = conn.createStatement();
5. 执行sql
   1. stmt.executeUpdate(); // 仅适用于 update 和 insert 返回值为int 型 为受影响的行数
   2. stmt.executeQuery(); // 仅适用于 select 的查询语句 返回值为ResultSet型，不是查询结果
   3. stmt.execute(); // 适用于所有的语句
6. 处理返回结果
7. 释放资源 🡪 ….close();.

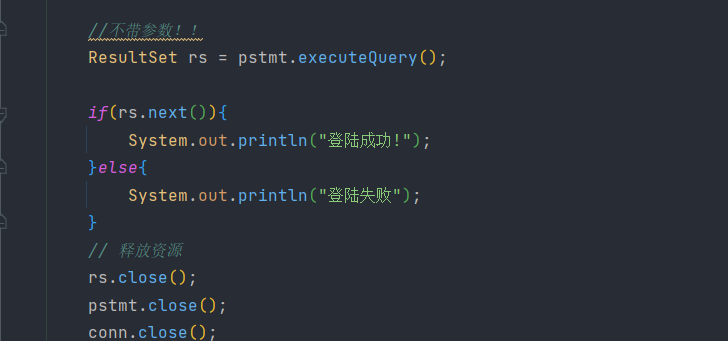
Connection也可以进行回滚，需要设置conn.set setAutoCommit(false); false:手动提交，true为自动提交，并把执行sql语句的语句放入try catch 中



DQL显示查询结果：getXxx(参数)参数为一行中的第i个数据，从1开始，Xxx根据数据类型自行选择

可以创建一个包、类去专门存放查询出来的数据，并放入ArrayList中

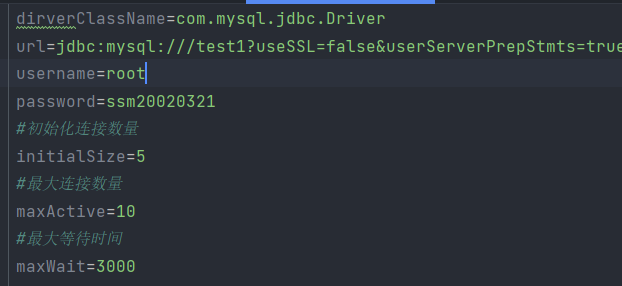
预防sql注入：原理：将敏感字符转义，“ **’**“ 🡪 “ **/’** ”   


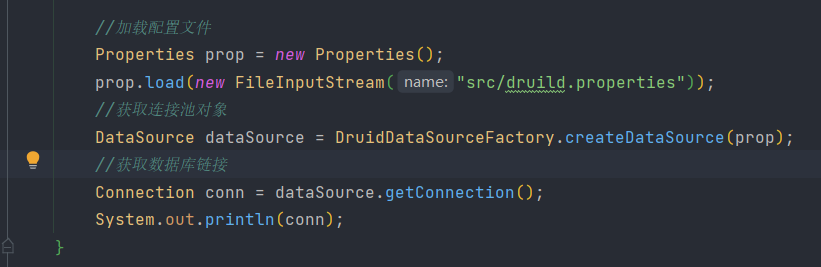


在url后写上 useServerPrepStmts=true 开启预编译功能 预编译只能编译一个sql

以上内容没用，接下来的数据库连接池Druid比较重要，经典白学

Properties中内容





实体类中，基本数据类型最好使用其对应的包装类型 例如: int 🡪 Integer

## Idea快捷指令

Alt + insert getter&setter, toString等操作

Ctrl + alt + t 快速构造环绕方式 try catch等

Alt + 鼠标左键拖动 可以选中多个行 并且不必从头开始

Ctrl + alt + L 格式化文件

Alt + / 快捷提示

## Maven

指令：

1. Mvn compile
2. Mvn clean
3. Mvn test
4. mvn package
5. Mvn instsll

坐标：都是双标签

1. groupId : 定义当前隶属组织的名称 通常是域名的反写 例如 com.itheima
2. artifactId : 定义maven项目的名称 通常是模块的名称 mysql-connector-java
3. version : 定义当前的版本号

<dependencies>

<dependency>

<groupId></groupId>

<artifactId></artifactId>

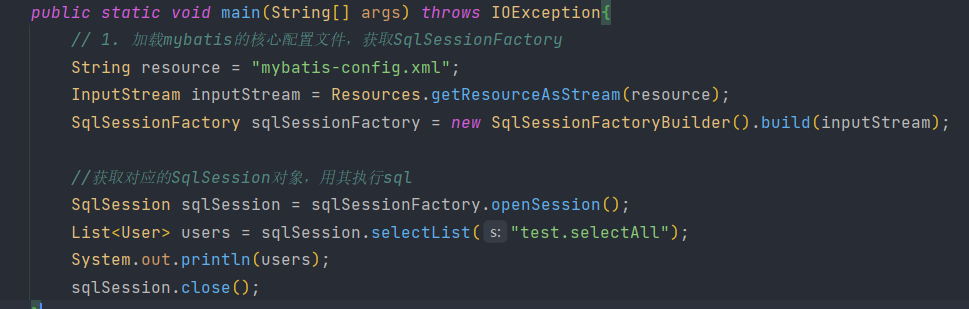
<version></version>

</dependency>

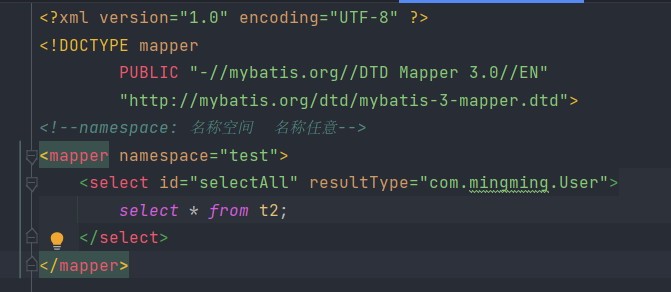
</dependencies>

## Mybatis

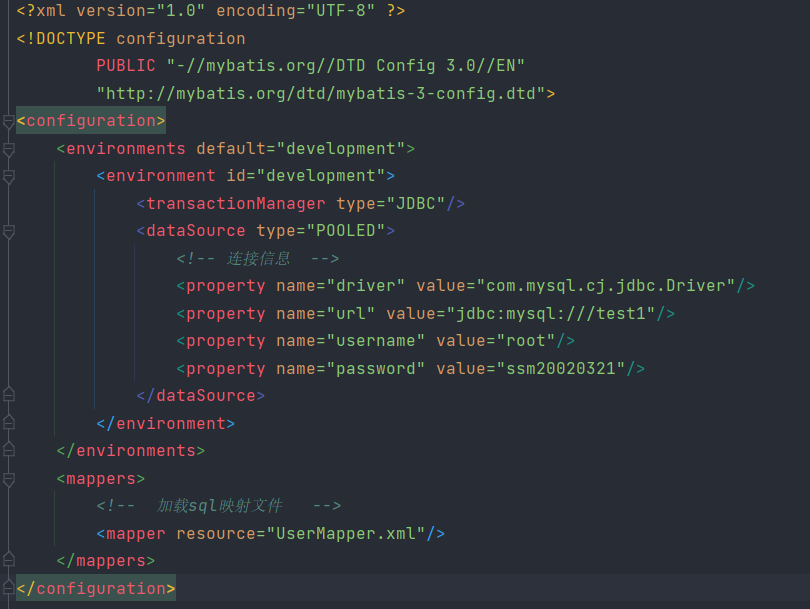
测试代码：前三行复制即可，导入对应的包并抛出IOException即可



UserMapper.xml ： 对User表进行查询，不同的表应有不同的名称



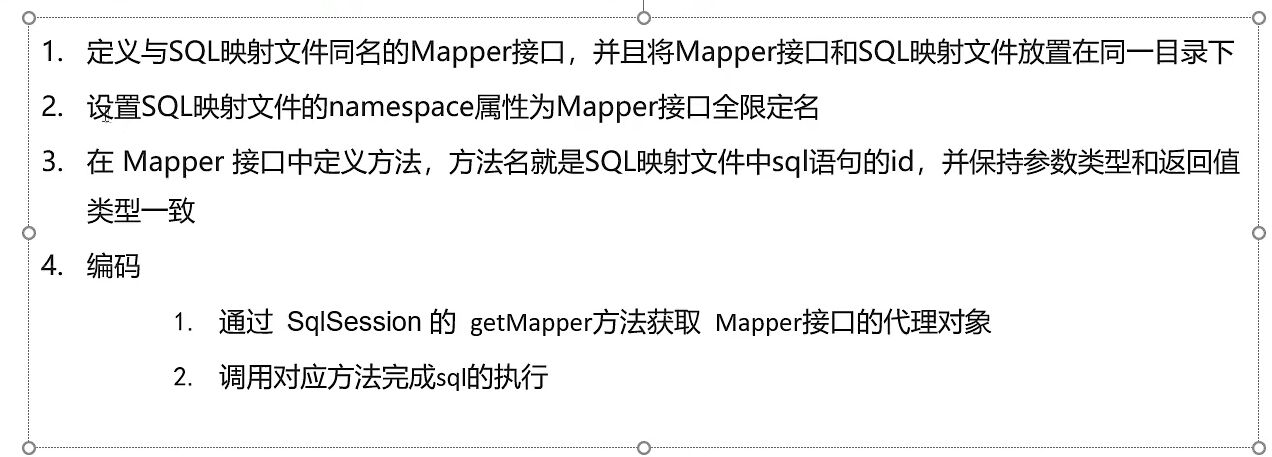
Mybatis-config.xml : 里面是mybatis的配置信息，也是复制即可



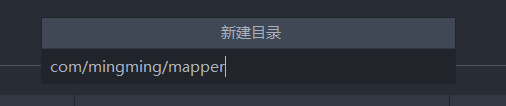
User类：一个表一个类



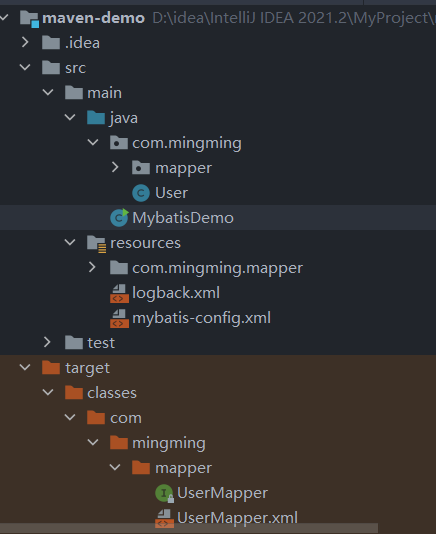
Mapper代理：  
resources



1. 放在同一目录下的方法（保证UserMapper.xml与接口UserMapper.class在同一目录）：在resourses下建一个与UserMapper.java路径相同的包，建包时注意平常的“.”要替换成”/”：e.g.：

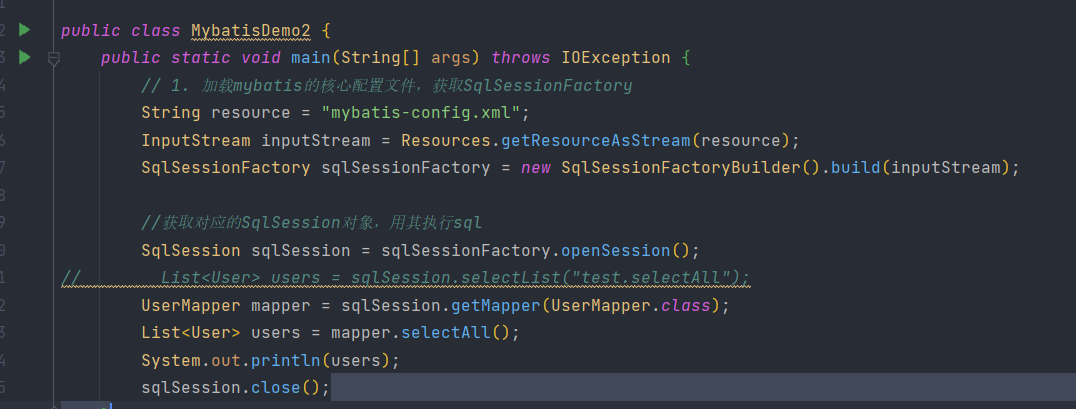


显示出来是正常的包的命名规则



1. 将namesapce指向对应的接口

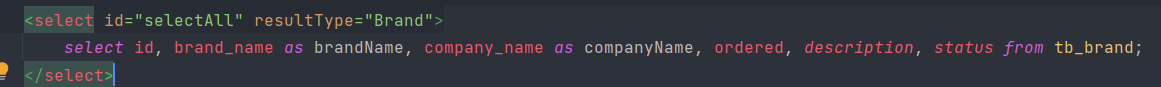


1. 
2. 
3. 如果以后映射文件过多，使用<package>

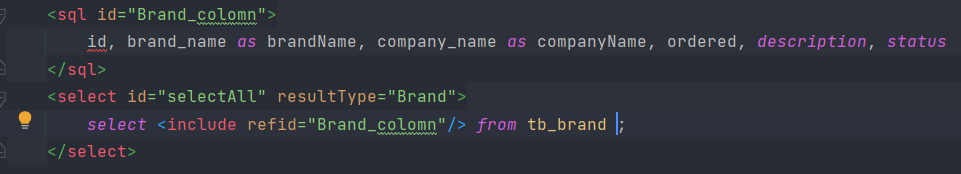


输出的数据为null 把字段名称 as 为你函数的名字，或者通过sql标签统一书写后导入，或者resultMap

一：



二：



三：resultMap中主要两个标签，<id>&<result>

<id>是主键的映射（唯一标识）

<result>是普通列的映射

！！！***使用resultMap之后 <select>中的resultType要改为resultMap!!!***

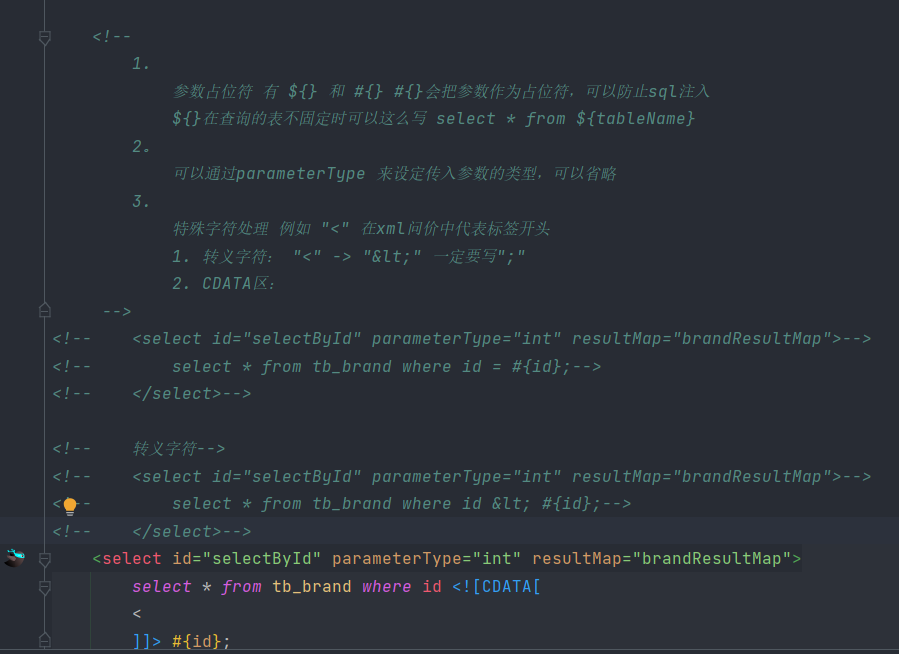


### 增删改查的细节

#### 查询

${ } 在查询的表不固定时可以这么写 select \* from ${tableName}

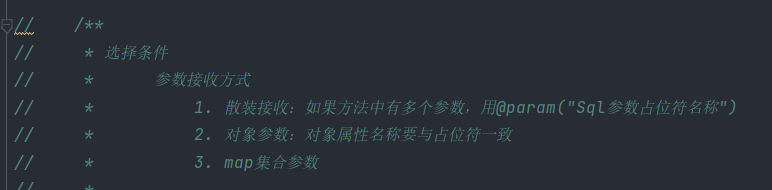
传参的细节

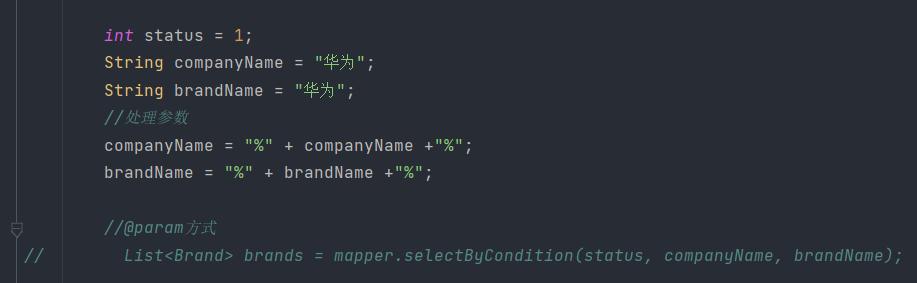
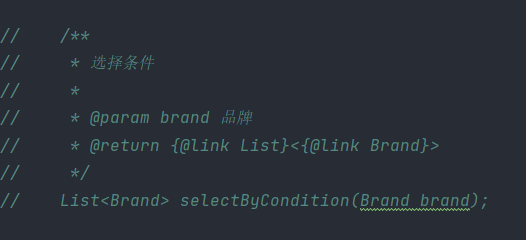
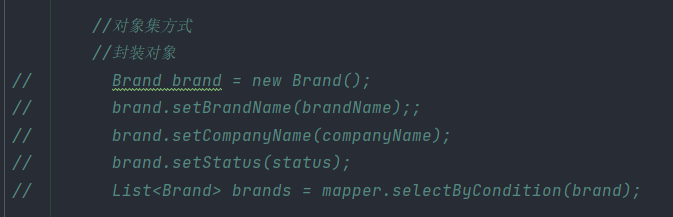


条件查询：、

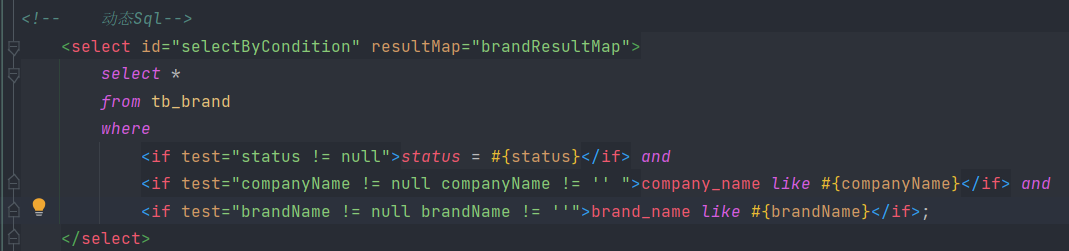
***静态***



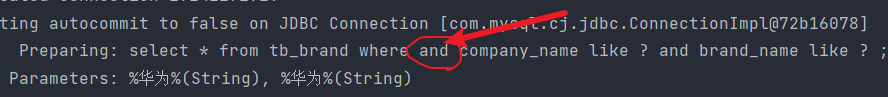


1.   
   
2.   
   
3.   
   

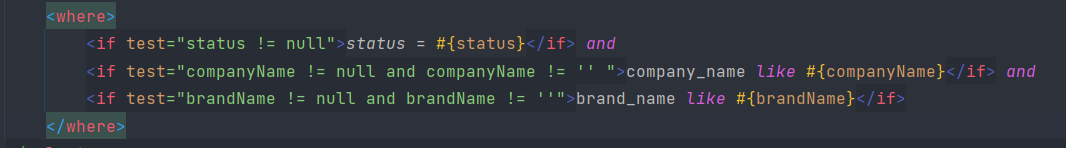
***动态：使用动态 Sql***



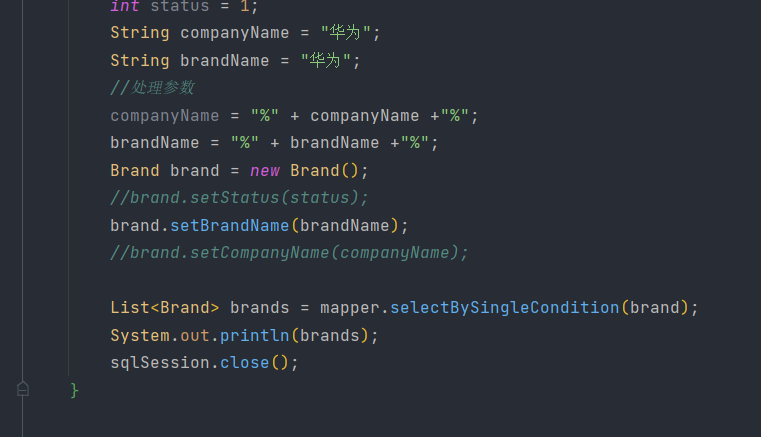
根据用户输入的不同可以查询不同的东西，但是上面的有bug



解决方法：where标签会自动删掉开头的and，但是说实话把and放if的末尾就很好的解决了

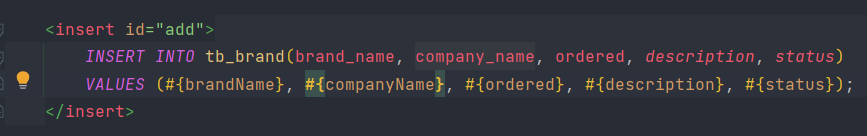


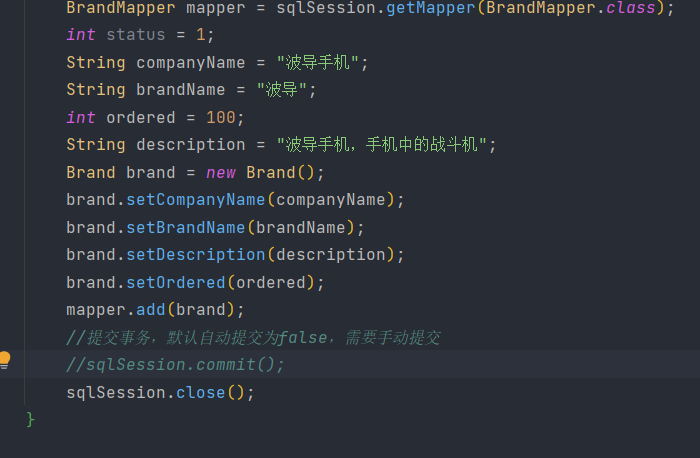
单条件动态查询：  

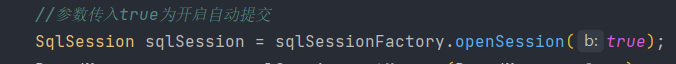



怕错误可以使用<where>标签

添加：需注意如果不设置自动提交会回滚事务，提交事务有两种方式1. 手动设置：sqlSession.commit(); 2. 设置自动提交：在获取sqlSession时传入参数，true为开启自动提交





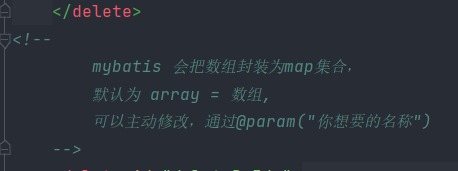


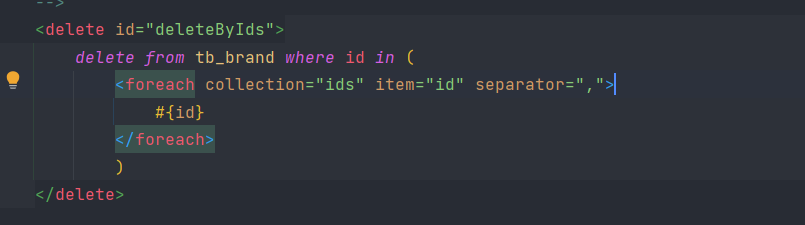
修改：也同时应该注意事务的提交



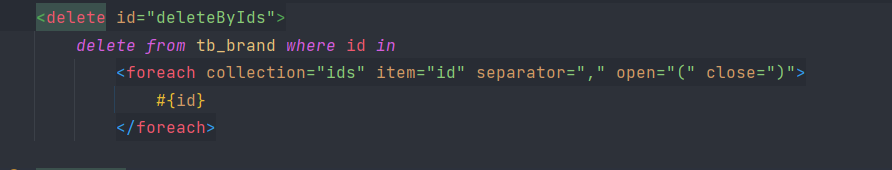


批量删除：





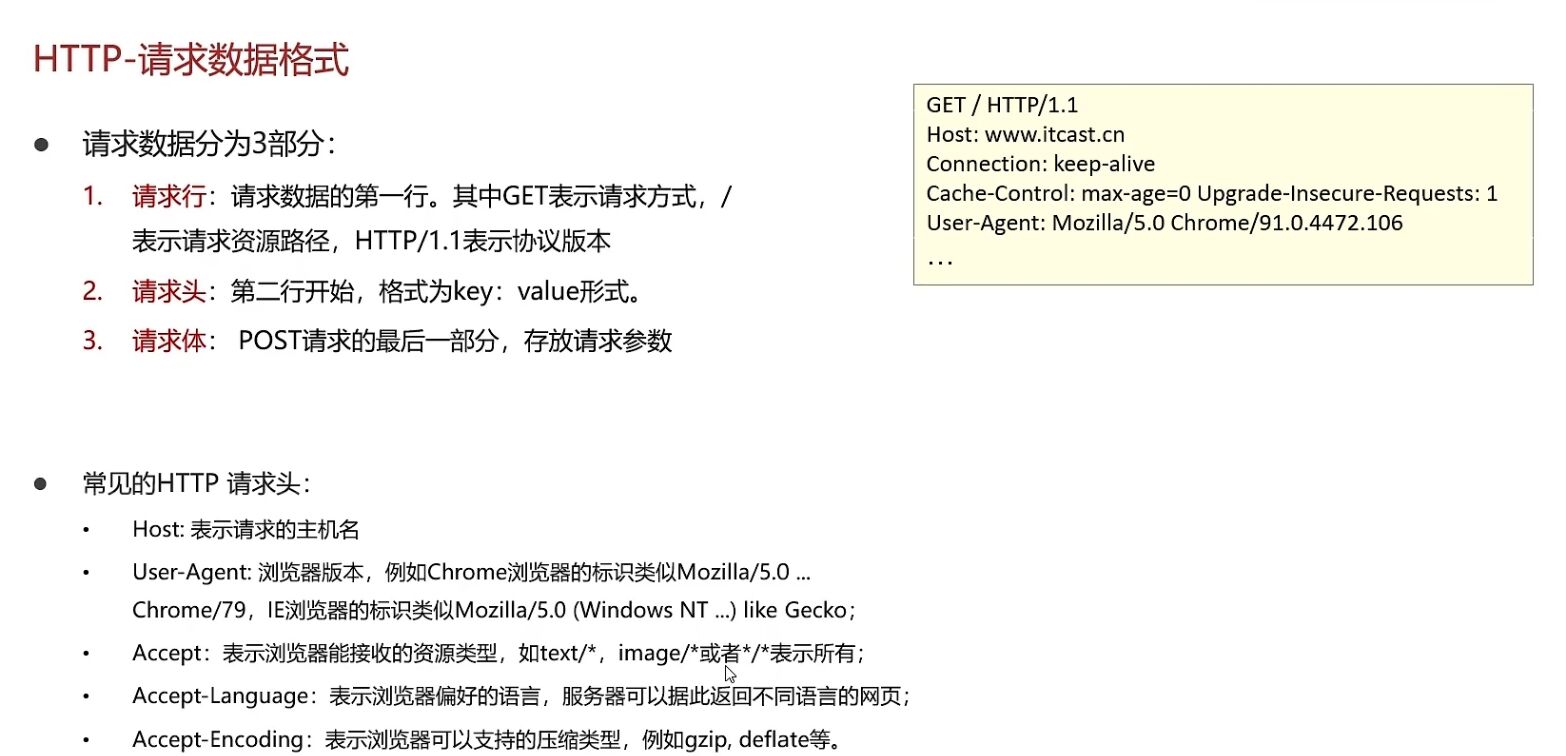
通过open和close可以少些俩括号



### 注解开发：

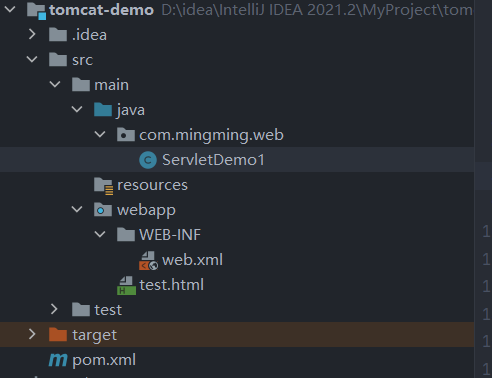
仅用来进行一些简单的操作，复杂操作还是要用配置文件(xml)的方式，例如动态sql

## HTTP

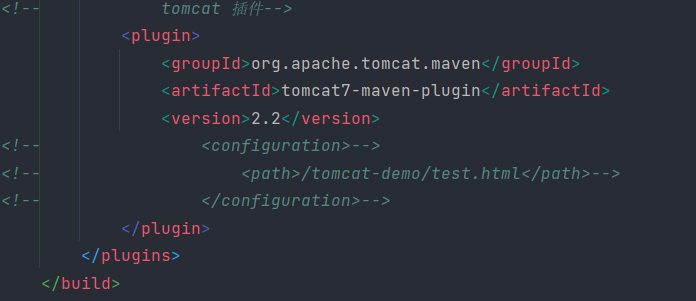


## Tomcat

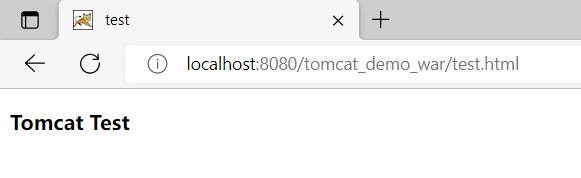
Maven配置Tomcat结构



Maven下的tomcat插件



访问



## Servlet

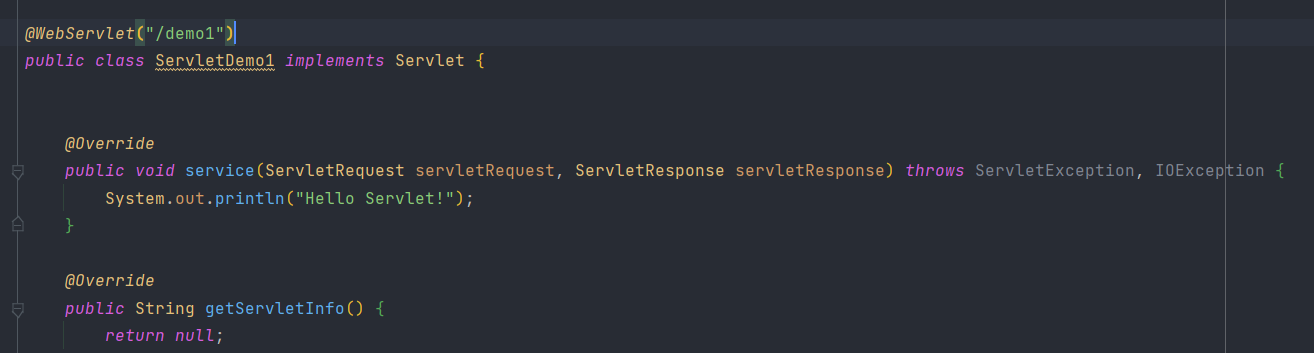
Maven导入依赖：(注意加上scope provided)



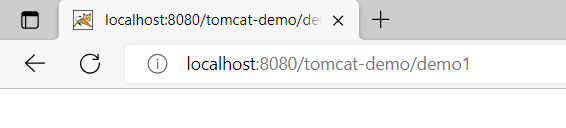
### Servlet接口

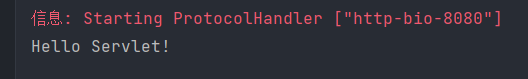


最重要的 service方法的实现：



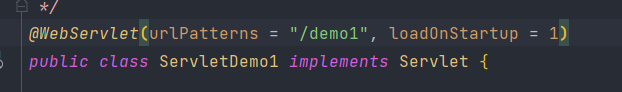
启动项目后并访问WebServlet中的地址后

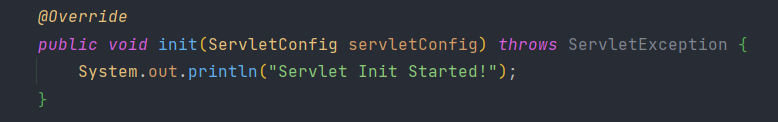




Init方法：默认loadOnStartup = -1 即访问时才初始化，修改为非负整数的时候按照数字的大小为优先级进行加载





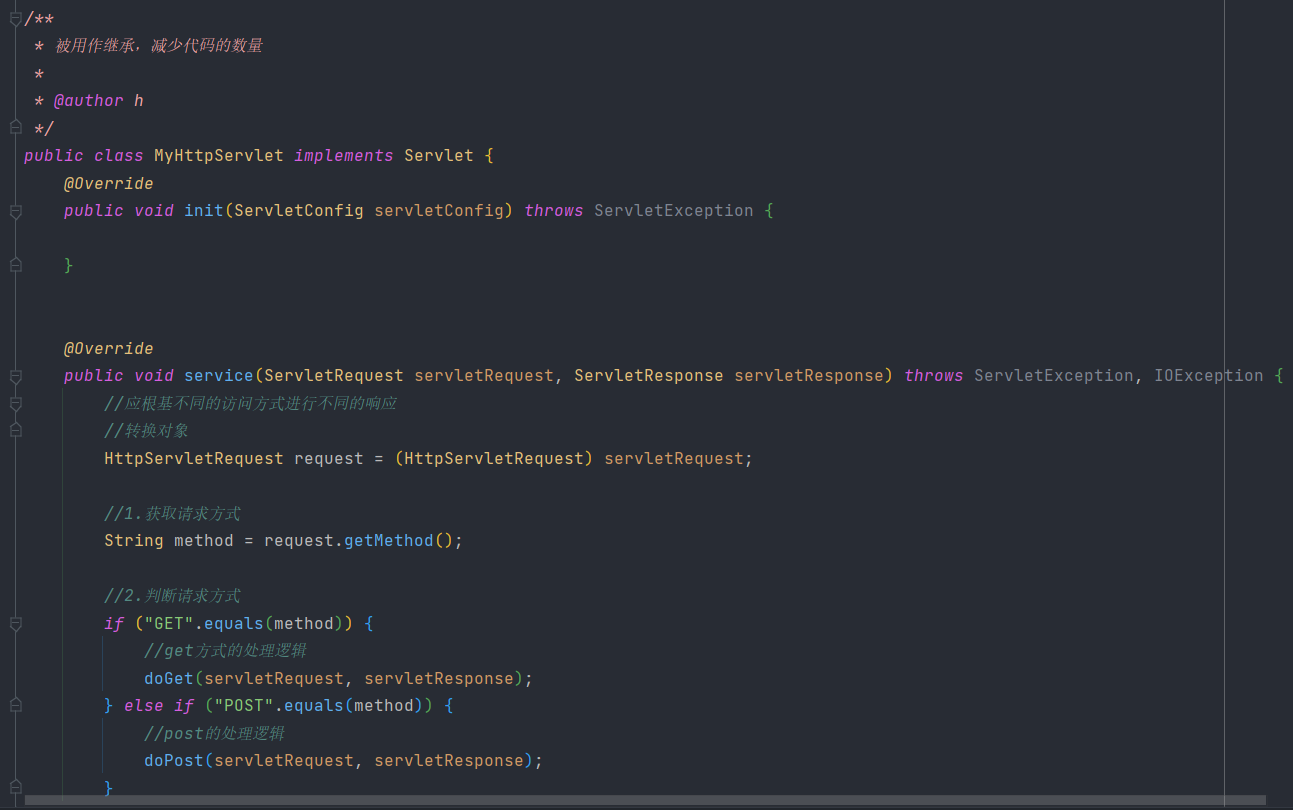


Service方法：

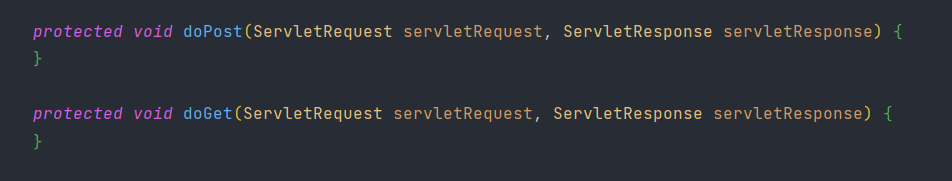


为了避免每次创建都写这些逻辑代码，可以把他封装成一个类，然后继承即可：

封装的类(MyHttpServlet)中：



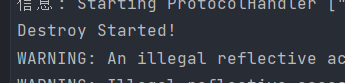
Protected:在子类中可以访问



调用时：只需要重写父类方法即可实现-



Destroy方法：

### HttpServlet父类：

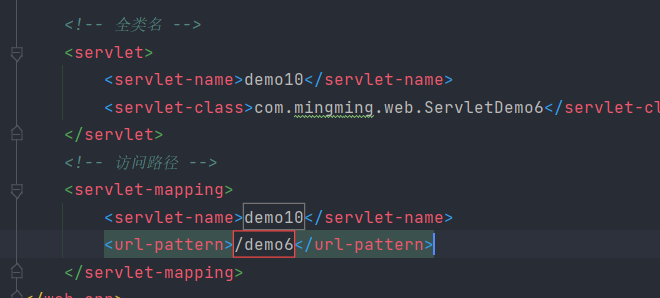




### XML配置servlet:

少用

之前的都是注解配置 @WebServlet(urlPattern = { }, loadOnStartup = )

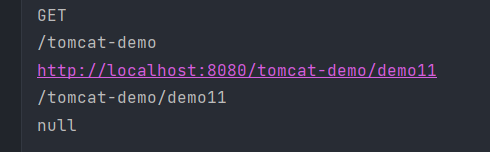


## Request

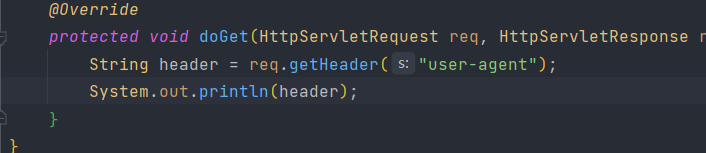


### 请求行



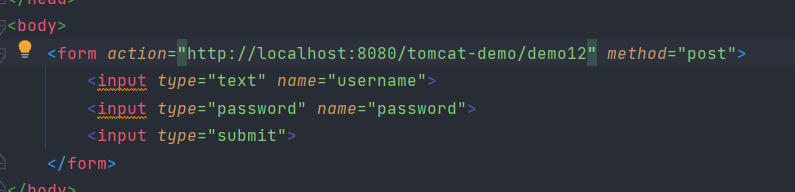


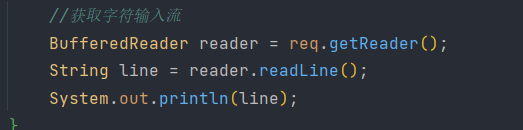
### 请求头：





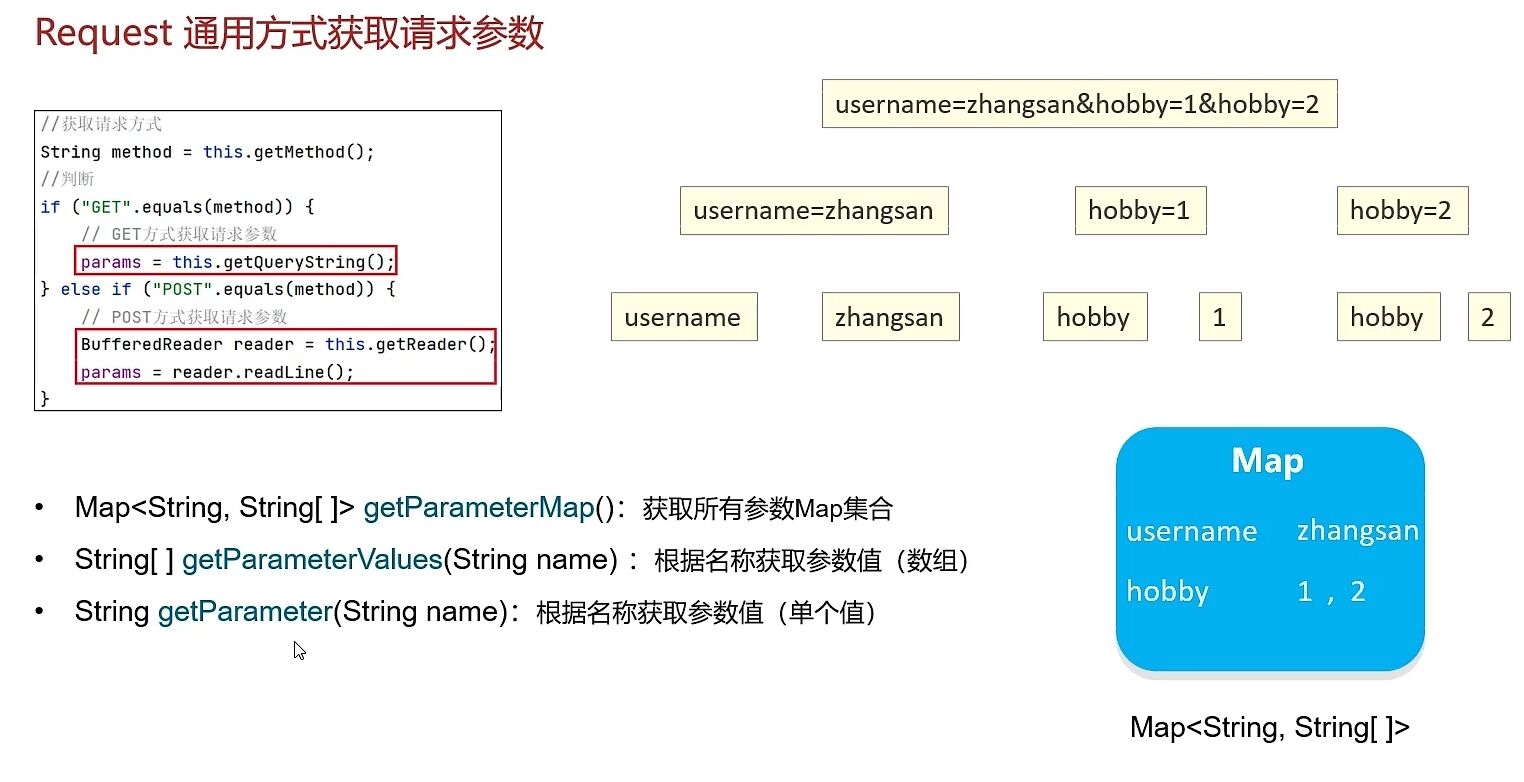
### 请求体post:



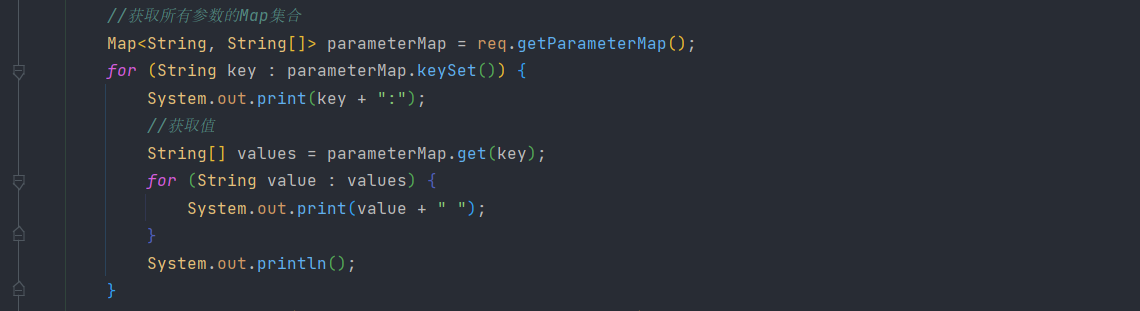


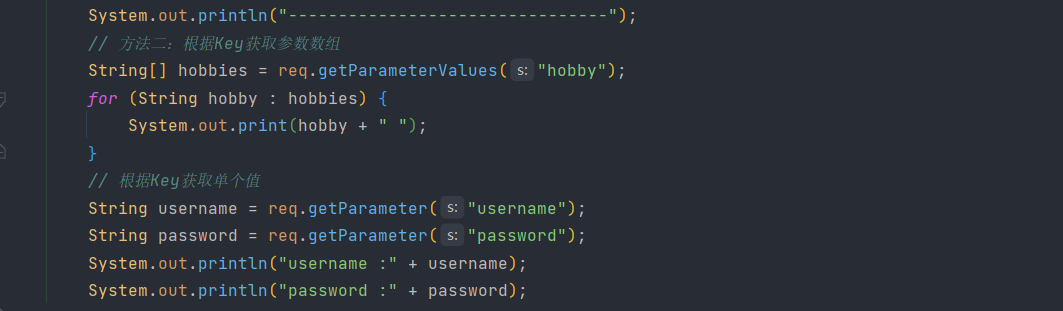


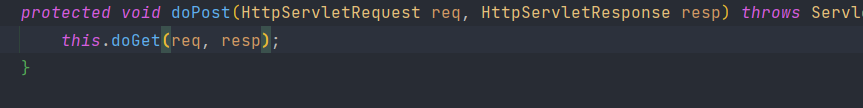
### \*\*统一获取请求参数的方式：



用统一的好处：只需要写一次代码就可以，在另个函数中只需调用就可以，例如图三







#### \*\*中文乱码处理：

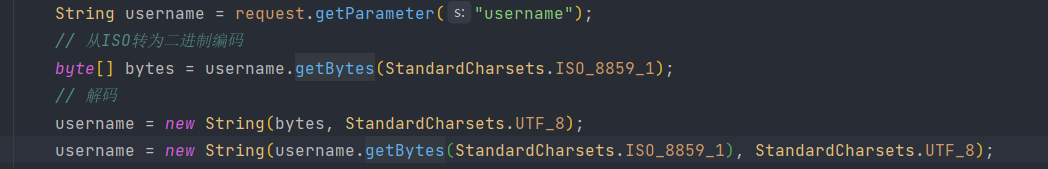
Post：



Get:



一行代码解决



### 请求转发：



跳转到/For2

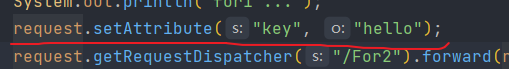
特点：

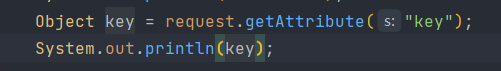
1. 浏览器地址栏不发生变化

2. 只能转发到服务器内的资源

3. 一次请求，可以在转发的资源间共享requesr资源

#### 请求转发之间共享数据：

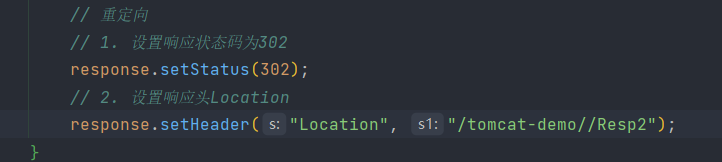




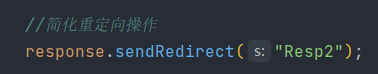
## Response：



### 重定向：



简化：

 // 应加虚拟目录，不知道为啥这个也行

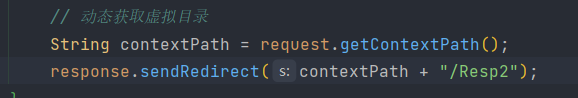
虚拟目录的加与不加看给谁使用，给浏览器使用就要加，



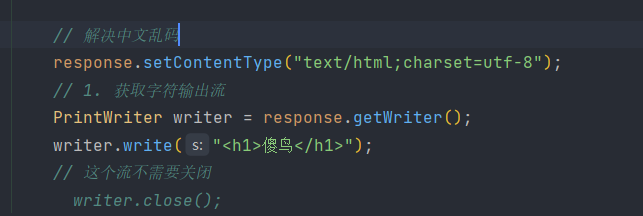
特点：

1. 地址栏发生变化
2. 可以重定向到任意资源，服务器内外皆可
3. 两次请求，不能再多个资源共享request的共享数据





### 响应字符数据：



### SqlSessionFactory代码优化：

把

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

放入一个静态代码块，可以保证不执行多次

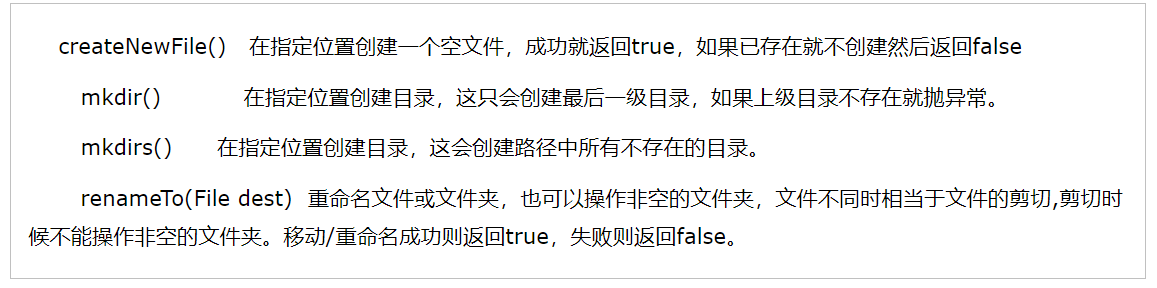


在类中只需这么写就可以



# FILE



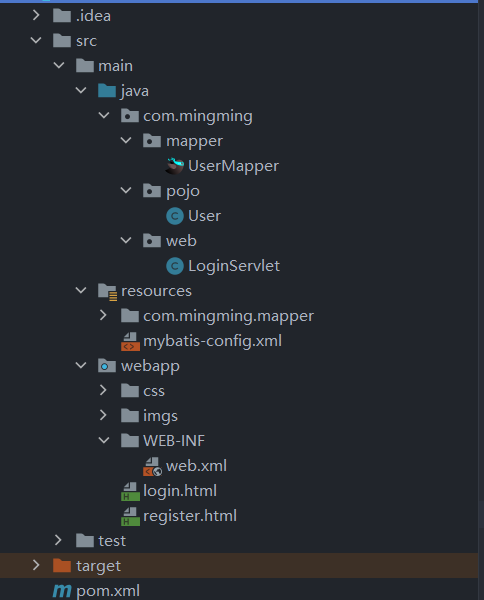


# 实战经验：

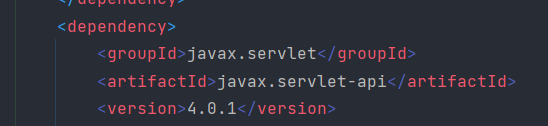
## 新建文件没有webapp?

刷新maven试试选中你的web项目再试试

## 项目结构不清楚？



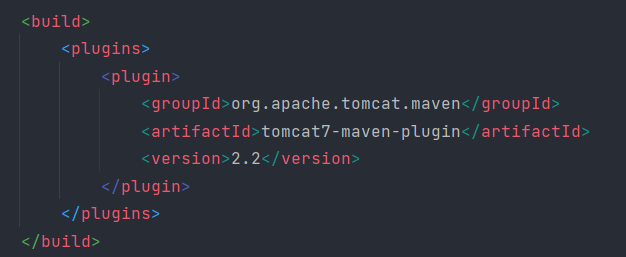
## 新建没有servlet?



## 报错你的xxx不是一个servlet?



## Tomcat7插件用不了？



## 表单不输入时参数为空字符串而不是null!

## 发现数据取不出来？

在xml映射文件中设置resultMap试试

## 查询的数据乱码？

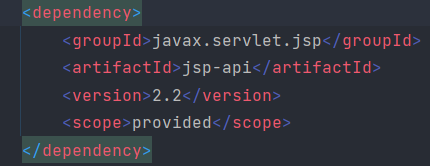
Post: request.setCharaterEncoding(“utf-8”);

## Js只能获取c:foreach的第一个数据？





# JSP：Java Server Pages

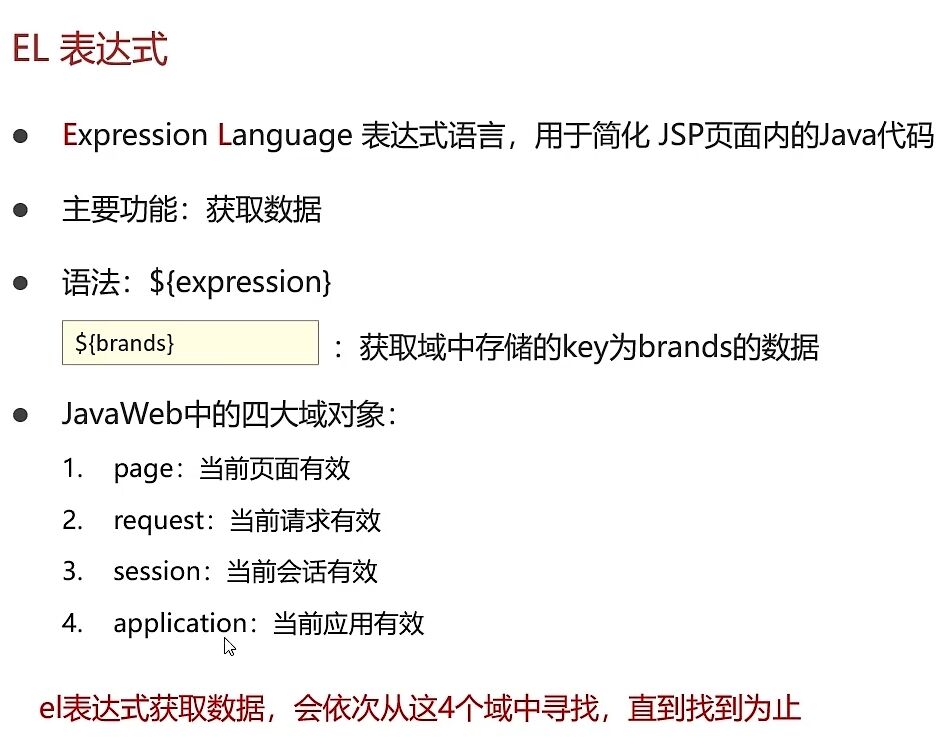




Java代码可被截断



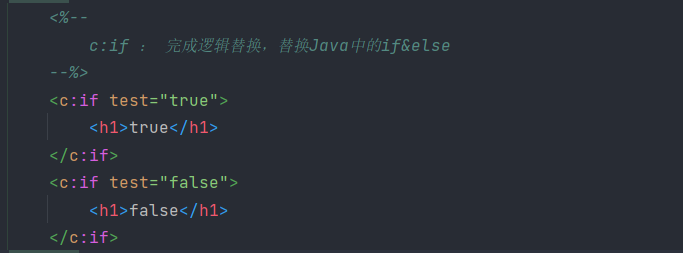
## EL表达式：Expression Language

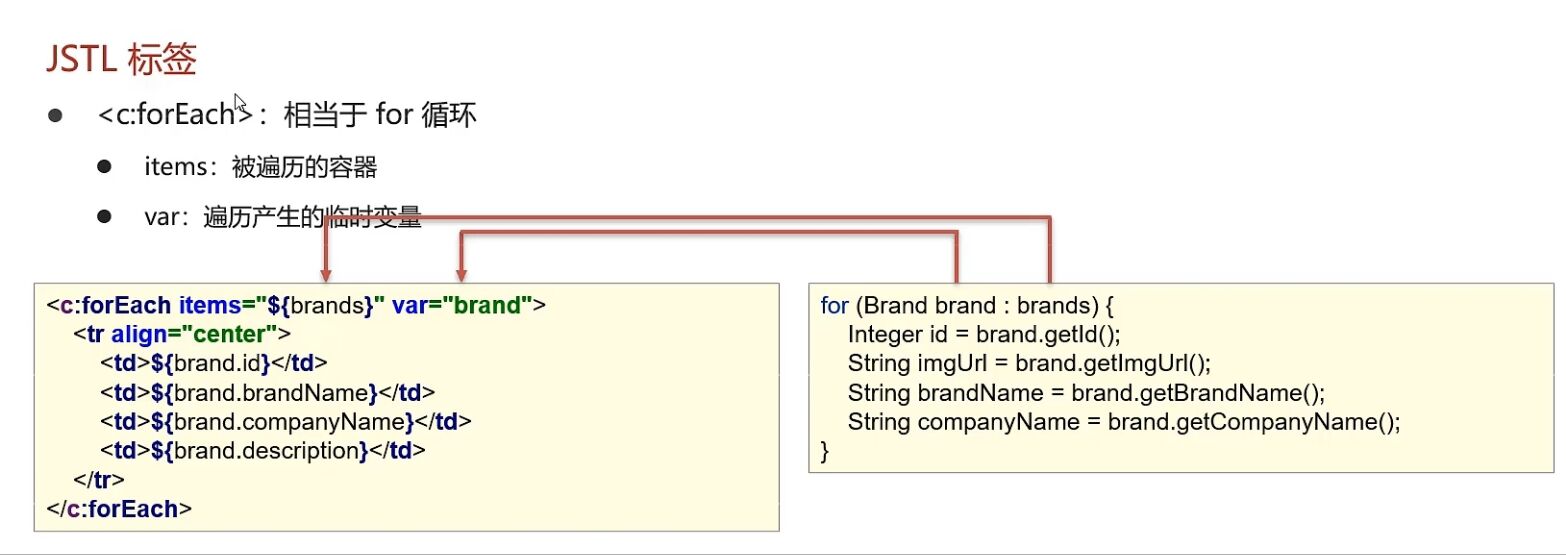


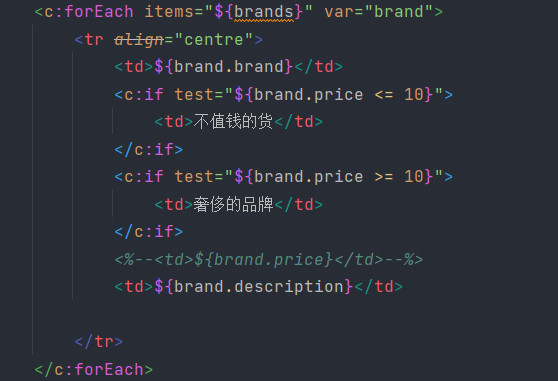
## JSTL：Jsp Standard Tag Library



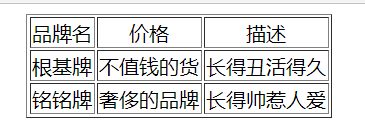






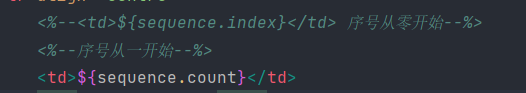




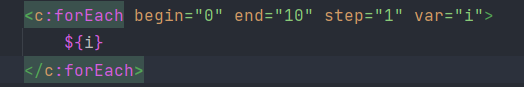




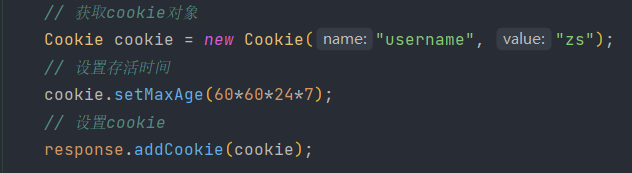
利用varStatus进行排序操作

、

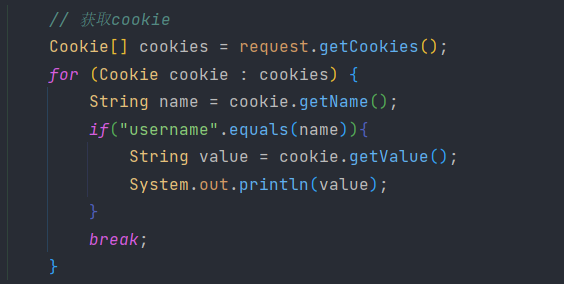
熟悉的for循环



# Cookie







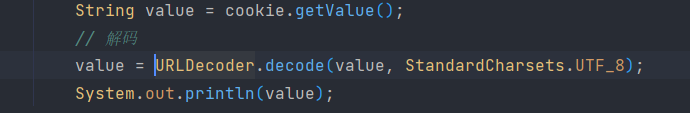




想使用中文要进行URL编码

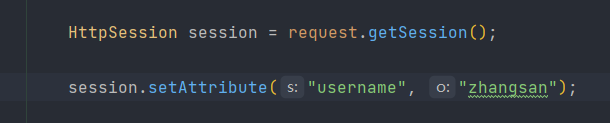


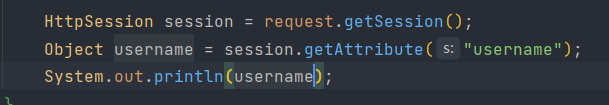
获取的时候解码



# Session

Session的实现依靠cookie





Session的钝化：在服务器关闭时会自动将session写入硬盘



Session的活化：在服务器开启时将session读到服务器，并删除文件

**注：服务器的关闭是正常关闭**

Session正常会在30min后失效，可以通过配置文件修改

