B) 领域模型命名规约

1） 数据对象：xxxDO，xxx 即为数据表名。

2） 数据传输对象：xxxDTO，xxx 为业务领域相关的名称。

3） 展示对象：xxxVO，xxx 一般为网页名称。

4） POJO 是 DO/DTO/BO/VO 的统称，禁止命名成 xxxPOJO。

word内 shift+f3 单元格内大写小写字母转换

sql查询表结构

SELECT

COLUMN\_NAME 列名,

COLUMN\_TYPE 数据类型,

DATA\_TYPE 字段类型,

CHARACTER\_MAXIMUM\_LENGTH 长度,

IS\_NULLABLE 是否为空,

COLUMN\_DEFAULT 默认值,

COLUMN\_COMMENT 备注

FROM

INFORMATION\_SCHEMA.COLUMNS

where

-- developerclub为数据库名称，到时候只需要修改成你要导出表结构的数据库即可

table\_schema ='hy'

AND

-- article为表名，到时候换成你要导出的表的名称

-- 如果不写的话，默认会查询出所有表中的数据，这样可能就分不清到底哪些字段是哪张表中的了，所以还是建议写上要导出的名名称

table\_name = 'hy\_account'

命令行连接远程数据库 注意大小写

mysql -u 用户名 -p密码 -h 服务器IP地址 -P 服务器端MySQL端口号 -D 数据库名

函数式接口：接口中仅含有一个抽象方法的接口，该接口中可以含有多个非抽象的方法（默认方法或静态方法）

函数式接口可以隐式转换为lambda表达式

定义

@FunctionalInterface

interface GreetingService

{

void sayMessage(String message);

}

实例化

GreetingService greetService1 = message -> System.out.println("Hello " + message);

java8 后接口中可以含有默认方法和静态方法

为什么要有默认方法

想象一个场景，假如在没有默认方法的情况下我们需要对集合框架增加一个新的迭代功能（forEach），那么我们就需要对其 Iterable 接口增加一个新的抽象方法，但是这样做的后果就是该接口下的所有实现类都必须去实现该方法。可是由于集合框架体系的庞大，实现类有非常多个，一改动全身，影响实在是太大，并不可取。

但是当有默认方法时，就解决了这个问题，就算在接口中增加了新的默认方法，具体的实现类也无需改动就有了新功能。

默认方法的出现就是为了保证原有代码兼容性的同时在接口中增加新的功能。

获取到项目的文件根目录

String basepath = request.getSession().getServletContext().getRealPath("/");

**JSTL <C:forEach>标签**

**<c:forEach**

**items="<object>"**

**begin="<int>"**

**end="<int>"**

**step="<int>"**

**var="<string>"**

**varStatus="<string>">**

**${userStatus.index} 此项的索引，从0开始**

**${userStatus.count} 此项的计数序号，从1开始**

**${userStatus.first} 此项是否是第一项，布尔值**

**${userStatus.last} 此项是否是最后一项，布尔值**

**${userStatus.begin} 此次迭代的起始索引，对应<c:foreach>中begin属性值**

**${userStatus.end} 此次迭代的终止索引，对应<c:foreach>中end属性值**

**${userStatus.step} 此次迭代的跳跃步伐，对应<c:foreach>中step属性值 。**

1. **sql文件表名为大写，mysql运行时不区分大小写造成表名不一致**

**在mysql安装目录中找到my.ini文件，添加**

**#区分大小写 1区分 2区分**

**lower\_case\_table\_names=2**

**配置项即可**

1. **excel数据生成sql插入语句**

**在数据行末尾找一个单元格，选中，在函数栏中输入（示例）**

**="insert into EAMS\_ARCHIVE\_FILED2(ID,ITEMNUM,TITLE,DEP,SDATE,EDATE,REMARK) values('"&A3&"','"&A3&"','"&B3&"','"&F3&"','"&G3&"','"&H3&"','"&J3&"');"**

**点击√确定后，字符串中定义的变量会转换成数据值，**

insert into EAMS\_ARCHIVE\_FILED2(ID,ITEMNUM,TITLE,DEP,SDATE,EDATE,REMARK) values('1','1','钱江御府项目计划、用地、规划、各专项审查、勘察设计合同等前期文件','杭州市房地产开发集团有限公司','2010.03.15','2016.08.02','');

**点击单元格右下角下拉，使此列每一个单元格都填充好内容，复制到数据库中执行即可插入数据。**

**如果单元格内容需要拆分开，选中此列单元格，选择“数据”操作菜单，选择“分列”操作，按照提示拆分单元格为两列或更多列。**

**操作过程中遇到“若要执行此操作，所有合并单元格需大小相同”，可以全选单元格，在“开始”中选择清除格式，继续操作即可。**

1. **mysql复制表**

**1.只复制表结构： CREATE TABLE tablename (LIKE source\_table\_name)**

**完整语句格式（CREATE TABLE IF NOT EXISTS tablename (LIKE source\_table\_name)）**

**2.复制表结构及数据 CREATE TABLE tablename SELECT \* FROM source\_table\_name**

**完整语句结构 （CREATE TABLE IF NOT EXISTS tablename SELECT ... (some valid select or union statement)）**

**SELECT复制的结果只包含了select的字段和数据，其它表属性都有系统的配置文件决定；包括存储引擎、默认字符集等都是有系统的默认配置所决定。**

**真正的表结构复制方法是LIKE方法，如果不需要考虑表原本的属性包括存储引擎、备注、主键、索引等那么select复制方法是个不错的方法并且还能连同数据一起复制。**

1. **String 与 byte[]互转**

**byte[] b = str.getBytes();**

**String a = new String(b);**

1. **数据转换标签配置**

**import** org.springframework.format.annotation.DateTimeFormat;

**import** com.fasterxml.jackson.annotation.JsonFormat;

@DateTimeFormat(pattern="yyyyMMdd")

@JsonFormat(pattern="yyyyMMdd",timezone="GMT+8")

1. mysql 查看进程列表 show processlist

停止进程 kill [进程号]

1. ctrl+shift+L--查看快捷键列表

ctrl+shift+L 连续两侧 --修改快捷键

\*\*\*\*append()--prepend()

\*\*\*\*before()--after()

\*\*\*\*前端获取input(file)文件名

var filename = $("input[name=ddd]")[0].files[0].name;

\*\*\*\*Java中 VO、 PO、DO、DTO、 BO、 QO、DAO、POJO的概念

PO（persistant object）: 持久对象，o/r（对象关系）映射之中出现的概念，对象对应数据库模型

DO(domain object):领域对象，现实世界抽象出来的有形或者无形的实体，一般与数据中的表结构对应

TO(Transfer object)数据传输对象，在应用程序不同关系之间的传输对象

DTO（Data Transfer Object）数据传输对象

这个概念来源于J2EE的设计模式，原来的目的是为了EJB的分布式应用提供粗粒度的数据实体，以减少分布式调用的次数，从而提高分布式调用的性能和降低网络负载，但在这里，泛指用于展示层与服务层之间的数据传输对象。

VO(view object) 值对象

视图对象，用于展示层，它的作用是把某个指定页面（或组件）的所有数据封装起来。

BO(business object) 业务对象

从业务模型的角度看 , 见 UML 元件领域模型中的领域对象。封装业务逻辑的 java 对象 , 通过调用 DAO 方法 , 结合 PO,VO 进行业务操作。 business object: 业务对象 主要作用是把业务逻辑封装为一个对象。这个对象可以包括一个或多个其它的对象。 比如一个简历，有教育经历、工作经历、社会关系等等。 我们可以把教育经历对应一个 PO ，工作经历对应一个 PO ，社会关系对应一个 PO 。 建立一个对应简历的 BO 对象处理简历，每个 BO 包含这些 PO 。 这样处理业务逻辑时，我们就可以针对 BO 去处理。

POJO(plain ordinary java object) 简单无规则 java 对象

纯的传统意义的 java 对象。就是说在一些 Object/Relation Mapping 工具中，能够做到维护数据库表记录的 persisent object 完全是一个符合 Java Bean 规范的纯 Java 对象，没有增加别的属性和方法。我的理解就是最基本的 Java Bean ，只有属性字段及 setter 和 getter 方法！。

DAO(data access object) 数据访问对象

是一个 sun 的一个标准 j2ee 设计模式， 这个模式中有个接口就是 DAO ，它负持久层的操作。为业务层提供接口。此对象用于访问数据库。通常和 PO 结合使用， DAO 中包含了各种数据库的操作方法。通过它的方法 , 结合 PO 对数据库进行相关的操作。夹在业务逻辑与数据库资源中间。配合 VO, 提供数据库的 CRUD 操作

87332752

\*\*\*\*js设置readio选中和取消

$("input[name=is\_advance][value=1]").removeAttr("checked");

$("input[name=is\_advance][value=2]").attr("checked",**true**);

\*\*在Oracle中提供了三种类型的集合操作： 并(UNION)、交（INTERSECT）、差（MINUS）

UNION：将多个查询的结果组合到一个查询结果之中，并去掉重复值

UNION ALL：将多个查询结果组合到一个查询结果中，但是包含重复值

INTERSECT：返回多个查询结果相同的部分

MINUS：返回两个查询结果的差集

mysql中只支持union、unionall操作(参考：<https://blog.csdn.net/sanzhongguren/article/details/76615464>)

\*\*mysql 字符串转日期 **STR\_TO\_DATE**(accepttime,'%Y-%m-%d %H:%i:%s')

如何访问链表中间节点

对于这个问题，我们首先能够想到的就是先遍历一遍整个的链表，然后计算出链表的长度，进而遍历第二遍找出中间位置的数据。这种方式非常简单。若题目要求只能遍历一次链表，那又当如何解决问题？

可以采取建立两个指针，一个指针一次遍历两个节点，另一个节点一次遍历一个节点，当快指针遍历到空节点时，慢指针指向的位置为链表的中间位置，这种解决问题的方法称为快慢指针方法。

jstl <forEach> 动态条件终止循环（不确定终止索引的情况下）

<c:set var="flag" value="true"/>//循环进行标志

<c:forEach items="${listdata}" var="item">

<c:if test="${flag}">

//循环体

</c:if>

<c:if test="${循环结束条件}">

<c:set var="flag" value="false"/>//循环终止

</c:if>

</c:forEach>

【强制】不要在foreach循环里进行元素的remove/add操作。remove元素请使用Iterator方式，如果并发操作，需要对Iterator对象加锁。

foreach遍历集合的本质也是使用迭代器来进行遍历，那为什么直接使用迭代器就不会出错呢？原因在于删除方法使用的是集合的（list.remove(index i)）还是迭代器( iter.remove() )的. Itr类是ArrayList集合的内部类，实现了Iterator接口，在内部类中维护了两个变量modCount和expectedModCount,

modCount 表示：集合添加或者删除元素的个数

expectedModCount 表示：预期集合将会被修改的次数

java fail-fast机制会在两个变量不相等时抛出异常，认为集合的元素被外界因素修改，在遍历开始时，两个变量显然是相等的，当使用了集合的删除方法（list.remove()）时，会改变modCount的值（+1），从而在 foreach中的 iter.next()调用时检查出错误（java.util.ConcurrentModificationException），如果直接使用迭代器遍历，使用iter.remove方法删除元素，会修改相应的expectedModeCount变量（+1），从而使两个变量相等。

关于包装类Integer

1. int 和Integer在进行比较的时候，Integer会进行拆箱，转为int值与int进行比较。

2. Integer与Integer比较的时候，由于直接赋值的时候会进行自动的装箱，那么这里就需要注意两个问题，一个是-128<= x<=127的整数，将会直接缓存在IntegerCache中，那么当赋值在这个区间的时候，不会创建新的Integer对象，而是从缓存中获取已经创建好的Integer对象。二：当大于这个范围的时候，直接new Integer来创建Integer对象。

3. new Integer(1) 和Integer a = 1不同，前者会创建对象，存储在堆中，而后者因为在-128到127的范围内，不会创建新的对象，而是从IntegerCache中获取的。那么Integer a = 128, 大于该范围的话才会直接通过new Integer（128）创建对象，进行装箱。

4.Integer.parseInt(s)与Integer.valueOf(s)的联系

Integer.parseInt(s)是把字符串解析成int基本类型，Integer.valueOf(s)是把字符串解析成Integer对象类型

5.Integer equals()源码

public boolean equals(Object obj) {

if (obj instanceof Integer) {

return value == ((Integer)obj).intValue();

}

return false;

}

所以不能使用Integer对象.equals("字符串整数")去判断是否相等

执行sql时mysql报错：Lock wait timeout exceeded; try restarting transaction

原因：事务未执行完成就将tomcat关闭掉

解决：使用show processlist查看当前正在执行的进程，可以将所有本地正在操作的进程使用 kill [id]的方式关闭掉

使用jquary选取dom元素时，如果属性值中包含[、]等需要转义的值时，可以使用另外一种选取方法

例 $("select[name=node[0].node\_id]).val()--------控制台报错

解决 $("select[name$='node[0].node\_id']").val()------选取成功----注意不要忽略符号$以及'

贴一段为下拉框添加元素的代码

**function** appendOptionTo($o,k,v,d){//$o:目标下拉框，k：文字，v：值，d：默认值

**var** $opt=$("<option>").text(k).val(v);

**if**(v==d){

$opt.attr("selected", "selected");

}

$opt.appendTo($o);

}

Q2：计算机运算为何要使用补码？

A2：可以简化计算机的运算步骤，且只用设加法器，如做减法时若能找到与负数等价的正数来代替该负数，就可以把减法操作用加法代替。而采用补码，就能达到这个效果。

hashmap的默认大小是 16

ArrayList默认大小是10

//mybatis使用resultMap映射实体类一对多关系时，没有gettter、setter方法也可以正常映射，但是在实体类转换json时，集合属性如果没有访问方法就会导致转换信息丢失

ctrl+shift+/ 折叠代码快捷键 ctrl+shift+\* 放开代码快捷键

使用接口开发的意义：接口定义了一种规范，规定了系统中的哪些方法用于外界调用，接口的调用者不用关心接口中具体的实现，接口的实现者不用担心调用者调用了接口中定义以外的方法，具有良好的封装性，在多人协作开发中，接口保证了开发的效率和规范，而且还具有良好的拓展性。(关于拓展性：假设不使用定义接口实现的方法，直接定义一个方法，然后写具体的实现功能的步骤,那么对于日后想要更换业务逻辑，或者既想保留原来的实现逻辑，又要在其他地方实现另一种逻辑（功能相同，逻辑不同（微信支付，支付宝支付。。。）），这时只能写另一个类，如果实现接口,就可以写另一个实现类，在实例化时区别开即可)

了解一个前端框架layui、bootstrap

通常如果一样的东西需要编码，就说明这样的东西并不适合传输。至于原因有多种多样，size过大，包含隐私数据等等。

要特别注意相等运算符==。JavaScript在设计时，有两种比较运算符：

第一种是==比较，它会自动转换数据类型再比较，很多时候，会得到非常诡异的结果；

第二种是===比较，它不会自动转换数据类型，如果数据类型不一致，返回false，如果一致，再比较。

由于JavaScript这个设计缺陷，不要使用==比较，始终坚持使用===比较。

layer获取当前窗口层数，父页面是0，第一层是1，，，

**var** index = parent.layer.getFrameIndex(window.name);

**var** t=window.parent.frames[index-2].document.getElementById("category").value;//获取父页面元素值

eclipse生成方法注释：【Alt】+【Shift】+【j】（光标移动到public前面）

大数据全套资料 <https://pan.baidu.com/s/1RFYlWXLpvHK-GnomiFqvsg> 密码：p481（内含开发工具）

父窗口选中iframe ： 1.在创建iframe时指定id值 sss

2.$("#id值 iframe")[索引值，从0开始] var abc=$("#sss iframe")[0] -->选中第一个iframe

父窗口调用iframe方法 ： 选中的iframe.contentWindow.方法名() abc.contentWindow.bcd();

父窗口选中iframe元素 ： 选中的iframe.contents().find("#元素id") $("#sss iframe").contents().find("#sss");

\*\*\*一定要指定iframe的id ^^^^注意写法

通常对程序调试最多时使用的几个快捷键就是这几个，其中：

F5：单步进入，进入方法内部；

F6：下一行代码，逐行步进；

F7：返回至调用处的下一行代码

F8：继续运行，跳过当前中断，运行至下一个断点，若当前中断的后面没有断点，则运行程序不会中断。

静态方法不可以重写

接口由全局常量和抽象方法组成，接口不能有静态方法

\*\*\*\*实体类接受时间格式问题@DateTimeFormat()

ctrl+shift+r：打开资源 输入文件名快速检索

ctrl+o：快速outline 列出当前类所有属性和方法的列表

ctrl+e：快速转换编辑器 所有打开的文件列表

查看check与uncheck异常说明

对象浅克隆与深克隆 利用流来进行深度克隆

了解SqlServer

方法重写---运行时多态

<c:forEach items=*"*${columnsList }*"* var=*"item"*>

<c:choose>

<c:when test=*"*${fn:contain(item.columnName,'|--') }*"*>

<option value=*"*${item.id }*"*>${item.columnName }</option>

</c:when>

<c:otherwise>

//下拉列表只可看不可选

<optgroup label=*"*${item.columnName }*"*></optgroup>

</c:otherwise>

</c:choose>

</c:forEach>

为什么说java只有值传递而没有引用传递？

当x是基本数据类型传递给方法时，方法拿到的只是x的拷贝值而并非x本身，所以此时即使改变x的拷贝值，那么x的值也是不变的；

当x是引用数据类型时，x是某个对象的引用，是这个对象的地址值，那么传递给方法的是地址值的拷贝，此时修改地址值也不会改变原地址的值，但是利用地址值去修改对象的值是会影响到原对象的，但因为传递的仍然是地址值，是一个值，那么仍然属于值传递而非引用传递。

孙宏民:

tar -zxvf 文件名

孙宏民:

tailf 文件名具有动态查看文件内容的功能，常用于查看服务器启动日志

孙宏民:

/etc目录下存放系统配置文件

孙宏民:

linux 对系统权限要求严格，普通用户只对home/用户名下的文件具有操作权限，而root用户具有所有文件的读写权限

孙宏民:

clear 清屏

孙宏民:

rm -f 文件 强制删除

孙宏民:

rm -r 文件夹名

孙宏民:

rm 文件名

孙宏民:

cp -r 源文件夹名 目标路径＋文件夹名

孙宏民:

cp 源文件名 目标路径＋文件名

孙宏民:

打开环境变量配置文件 /etc/profile

export JAVA\_HOME=/usr/local/jdk

export PATH=$JAVA\_HOME/bin:$PATH

expoty CLASS\_PATH=$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

孙宏民:

source /etc/profile 解析文件