什么是 jdk、jre？

JDK,Java Development Kit,

编写 Java 程序使用的软件,其中包含了编写 Java 程序时所需的类库、编译运行工具等

JRE　Java Runtime Environment，运行 Java 程序使用的软件.

JRE 在 JDK 中

classpath、path、java_home 作用分别是什么？

PATH：执行路径，是操作系统搜索本地可执行文件的目录列表

CLASSPATH：类加载路径，查找类的目录列表

类搜索顺序

1、当前目录

2、rt.jar

3、CLASSPATH

4、报错

JAVA_HOME：JDK 的安装根目录。其他 java 编写的软件在使用时，会搜索该环境变量以找到 jre

Java 数据类型有哪些？

8 基本类型：byte、short、char、int、long、float、double、boolean

引用类型：存储的是引用，通过引用访问对象

null 类型：值 null，可以赋给任意引用类型变量，表示未知、不存在。不可以赋给基本类型变量

Java 基本数据类型有哪些？精度范围从小到大是什么？对应的包装类有哪些？

精度从小到大：byte、char、short、int、long、float、double boolean

对应包装类型：Byte、Character、Short、Integer、Long、Float、Double、Boolean

&和&&有什么区别？

&&　逻辑运算符。

短路与，第一个操作数为 false，则不再计算第二个操作数

&　位运算符

非短路与，两个操作数都会被计算

Switch（k） k 的类型能是什么？

byte、short、int、char、枚举、字符串(Java7 新增)

Char 能不能存储中文？

能，一个 Unicode 字符占两个字节，char 类型的空间是 2 个字节

== 和 equals 的区别？

在 equals 方法没有被重写的时候，两个没有区别，都是比较栈中的值

如果 equals 被重写，则 equals 是按照重写规则进行比较

数组的数据结构是什么？

一块连续的存储空间　　线性表

方法重载要求有哪些？

在同一个类中，方法名相同，参数列表不同(个数、类型、顺序)，与返回值无关

在 1.6 编译版本下，如果参数中有泛型，则与返回值有关

```
public void someMethod(List<Integer> list){};
public int someMethod(List<String> list){return 1};
```

封装的理解

将对象的数据和行为组合在一起，把实现细节隐藏起来，不允许外部直接访问；把方法暴露出来，让方法来操作这些数据。两个方面的含义：该隐藏的隐藏，该暴露的暴露。

实现封装的关键在于决不能让其他类的方法直接地访问一个类的属性，应仅能通过该对象的方法操作该对象的数据

递归方法

递归方法：一个方法自己调用自己，方法的递归包含了隐式的循环，会重复执行某段代码

递归方法必须在某个时刻不再调用它自己，否则就变成了无穷递归，递归一定要向已知方向递归

常量池

常量池，在 Java 中用于保存 在编译期就可以确定的值，是已编译的 class 文件中的一份数据（常量表），执行器产生的常量也会放入常量池，故认为常量池是 JVM 的一块特殊的内存空间。

常量池，既是一块特殊的内存（运行时常量池），也是 class 文件中的一段描述（静态常量池）

方法重写要求有哪些？

在父子类中

返回类型必须与被重写方法的返回类型相同

重写方法不能比被重写方法限制有更严格的访问级别。

参数列表必须与被重写方法的相同

重写方法不能抛出新的异常或者比被重写方法声明的检查异常更广的检查异常。但是可以抛出更少，更有限或者不抛出异常。

static 修饰方法和属性 如何访问？

通过类名访问

Final 修饰类、方法、属性、局部变量分别有什么作用

1.修饰类

当用 final 修饰一个类时，表明这个类不能被继承。

2.修饰方法

当用 final 修饰一个方法时，表明这个方法不能被重写。

3.修饰变量

栈中的值不能被修改

构造方法什么时候调用？

1、创建对象

new Class.forName("").newInstance()

2、创建子类对象

this 代表什么？Super 代表什么？

this 表示的是当前对象的引用，哪个对象调用了 this 所在的函数，this 就代表哪个对象；

super 表示的是当前类的父类。

面向对象有几个特征？

封装,继承,多态

接口和抽象类的区别？

1.抽象类可以提供成员方法的实现细节，而接口中只能存在 public abstract 方法；

2.抽象类中的成员变量可以是各种类型的，而接口中的成员变量只能是 public static final 类型的；

3.接口中不能含有静态代码块以及静态方法，而抽象类可以有静态代码块和静态方法；

4.一个类只能继承一个抽象类，而一个类却可以实现多个接口。

什么是继承？一个类继承另一个类关键字是什么？

继承是 is a 的关系，A 继承 B 首先 A 是 B （比如 婴儿 是 人 Baby extends Person）

A 继承 B,A 拥有了 B 中的属性和方法

extends

什么是封装？Boolean 类型的 get、set 方法应该怎么写？

就是把对象的属性和操作（或服务）结合为一个独立的整体，并尽可能隐藏对象的内部实现细节

对于类型为 boolean 的属性 abc，其 get 方法写做 isAbc

什么是多态？

父类的引用指向子类的实例

class B extends A

A a = new B();

a 是 A 类型。即使用多态后，对象仍然是父类类型

数组的初始化方式有几种？

```
int[] i = new int[]{1,2,3,4,5};
int[] j = {1,2,3,4,5};
int[] k = new int[10];
for(){}
```

值传递和引用传递的区别

在 java 中，对象传递永远传递的是栈内存中的值，对于基本数据类型，栈中的值就是实际的值（所谓的值传递），对于引用类型，栈中的值是对象在堆中的地址（所谓的引用传递）

线程中 sleep()和 wait()的区别

在调用 sleep()方法的过程中，线程不会释放对象锁。

在调用 wait()方法的过程中，线程会释放对象锁。

在调用 sleep()方法,线程会自己醒来

在调用 wait()方法 ，线程需要被其他线程唤醒

在调用 sleep()方法前,不需要获取任何东西

在调用 wait()方法前，要先获取调用 wait 方法的对象的对象锁

RuntimeException 和 Exception 有什么区别

RuntimeException 继承自 Exception

RuntimeException 不需要程序员通过代码处理

Exception 需要处理

处理方式 1 try-catch

处理方式 2 throws

抛出 是将异常抛给调用者，如果调用者没有处理的话，将会继续向上抛。最终将抛给 jvm

接口是否可继承接口？抽象类是否可实现(implements)接口？抽象类是否可继承具体类?

接口可以继承接口。抽象类可以实现(implements)接口，

抽象类是可以继承具体类。抽象类中可以有静态的 main 方法。

记住抽象类与普通类的唯一区别就是不能创建实例对象和允许有 abstract 方法

Object 类中有几个方法？每个方法做什么的？

构造方法 1 个，其他方法 11 个

构造方法：Object()

clone() equals(Object obj) finalize() getClass() hashCode()

notify() notifyAll() toString() wait() wait(long timeout) wait(long timeout, int nanos)

notify() notifyAll()

如果重写了一个类的的 hashCode 方法，还要考虑什么？

考虑重写 equals 方法

Exception RuntimeException Error Throwable 之间的父子关系是什么样子的？

Throwable

Exception

RuntimeException

Error

列出 String 中至少 15 个方法（非继承于 Object 的）。说明这些方法的作用

charAt(int index)

contains(CharSequence s)

endsWith(String suffix)

equalsIgnoreCase(String anotherString)

getBytes()

indexOf(int ch)

isEmpty()

lastIndexOf(int ch)

length()

matches(String regex)

replace(char oldChar, char newChar)

replaceAll(String regex, String replacement)

split(String regex)

startsWith(String prefix)

substring(int beginIndex)

substring(int beginIndex, int endIndex)

toCharArray()

toLowerCase()

toUpperCase()

trim()

String 如何与 byte[] char[] int double boolean 进行转换？

1.string 和 byte[]之间的相互转化

string ->byte[]

```
byte[] midbytes=isoString.getBytes("UTF8");
```

byte[]->string

```
String isoString = new String(bytes);
String srt2=new String(midbytes,"UTF-8");
```

2.string 和 char[]之间转换

```
String s1 = "abc";
char[] ch = { 'a', 'b', 'c' };
String s3 = new String(ch);
char[] c = s1.toCharArray();
```

int 和 string 之间的相互转换

int -> String

```
int i=12345;
String s="";
第一种方法：s=i+"";
第二种方法：s=String.valueOf(i);
```

String -> int

```
s="12345";int i;
第一种方法：i=Integer.parseInt(s);
第二种方法：i=Integer.valueOf(s).intValue();
```

StringBuffer 是什么？ 和 StringBuilder 有什么区别？

StringBuffer 可变字符串

StringBuffer 线程安全

StringBuilder 非线程安全

String 在内存中如何存储？

String 的值存在于字符串常量池中。当新建一个 String 对象的时候，会先到常量池中查找，如果常量池中没有，则会新建一个放到常量池中，并将地址返回，如果常量池中有，则直接返回常量池中的对象。

什么是接口？

接口就是一些方法特征的集合------接口是对抽象的抽象。

接口是规范

用于定义实现类中的方法

用于解耦合，分离方法的调用者和实现者

集合如何分类？

collection

set

HashSet

TreeSet

list

ArrayList

LinkedList

Vector

Map

HashMap

TreeMap

Hashtable

ArrayList 的数据结构是什么？底层是通过什么来实现的？

线性表，底层通过数组实现，数组初始长度为 10

LinkedList 的数据结构是什么？

双向链表。

HashSet 的底层是什么实现的？

对于 HashSet 而言，它是基于 HashMap 实现的，

HashSet 底层采用 HashMap 来保存所有元素

6.List 和 set 有哪些区别？

Set 元素不允许重复，无序（与放入顺序无关）

List 允许重复，有序

三种遍历 Map 的方式分别是什么？

ketySet values entrySet

流如何分类？

按照输入的方向分，输入流和输出流

按照处理数据的单位不同分，字节流和字符流

按照功能的不同分，分节点流和包装流

Java 实现一个线程有几种方式？

第一种是继承 Thread 类　实现方法 run()　不可以抛异常 无返回值

第二种是实现 Runnable 接口 实现方法 run() 不可以抛异常 无返回值

H5 的新特性

1.本地存储特性（Class: OFFLINE & STORAGE）

2.设备兼容特性 (Class: DEVICE ACCESS)

3.连接特性（Class: CONNECTIVITY）

4.Server-Sent Event（允许服务端推送数据到客户端）

5.WebSocket

6.网页多媒体特性(Class: MULTIMEDIA)

7.三维、图形及特效特性（Class: 3D, Graphics & Effects）

8.性能与集成特性（Class: Performance & Integration）

9.新的表单控件：比如 calendar、date、time、email、url、search，以及新的表单属性等。

10.新的特殊内容元素：比如 article、footer、header、nav、section。

Hibernate 工作原理及优点?

原理： 1.读取并解析配置文件

2.读取并解析映射信息，创建 SessionFactory

3.打开 Session

4.创建事务 Transation

5.持久化操作

6.提交事务

7.关闭 Session

8.关闭 SesstionFactory

优点： 1. 对 JDBC 访问数据库的代码做了封装，大大简化了数据访问层繁琐的重复性代码。

2. Hibernate 是一个基于 JDBC 的主流持久化框架，是一个优秀的 ORM 实现。他很大程度的简化 DAO 层的编码工作

3. hibernate 使用 Java 反射机制，而不是字节码增强程序来实现透明性。

4. hibernate 的性能非常好，因为它是个轻量级框架。映射的灵活性很出色。它支持各种关系数据库，从一对一到多对多的各种复杂关系。

线程生命周期有哪些状态？什么情况下回进入该状态？

生命周期的 7 种状态

新建（new Thread）

当创建 Thread 类的一个实例（对象）时，此线程进入新建状态（未被启动）。

就绪（runnable）

线程已经被启动，

正在等待被分配给 CPU 时间片，也就是说此时线程正在就绪队列中排队等候得到 CPU 资源。

运行（running）

线程获得 CPU 资源正在执行任务（run()方法），此时除非此线程自动放弃 CPU 资源或者有优先级更高的线程进入，线程将一直运行到结束。

死亡（dead）

当线程执行完毕或被其它线程杀死，线程就进入死亡状态，

这时线程不可能再进入就绪状态等待执行。

堵塞（blocked）

由于某种原因导致正在运行的线程让出 CPU 并暂停自己的执行，即进入堵塞状态。

锁池

当线程没有获取到对象锁的时候，会进入锁池

等待池

调用 wait 方法的时候，会进入等待池

什么是反射？通过反射能够获取一个类中的哪些信息？

反射定义：

在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；

对于任意一个对象，都能够调用它的任意一个方法和属性；

这种动态获取信息以及动态调用对象的方法的功能称为 java 语言的反射机制。

能获取

类所在包,类名，父类，实现的接口

属性：

修饰符，类型，名字，值

方法：

修饰符，返回值类型，名字，抛出的异常、参数列表

构造方法

修饰符

Jdbc 访问数据库有哪些步骤？

1、注册 JDBC 驱动：
2、获得连接对象
3、获得状态集创建一个 Statement
4、执行 SQL 语句
5、获取结果集 处理结果
6、关闭 JDBC 对象

什么是事务？事务有哪些特征？多线程并发情况下，如果没有事务会出现哪些问题？

什么是事务

一次不可再分的操作

事务具有四个特征：（这四个特性简称为 ACID 特性）

原子性（ Atomicity ）、

一次不可再分的操作，事务中包含的各操作要成功都成功，要失败都失败

一致性（ Consistency ）、

事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。

隔离性（ Isolation ）

一个事务的执行不能其它事务干扰。

持续性（ Durability ）。

也称永久性，指一个事务一旦提交，它对数据库中的数据的改变就应该是永久性的。

并发问题可归纳为以下几类:

A.丢失更新：撤销一个事务时，把其他事务已提交的更新数据覆盖（A 和 B 事务并发执行，A 事务执行更新后，提交；B 事务在 A 事务更新后，B 事务结束前也做了对该行数据的更新操作，然后回滚，则两次更新操作都丢失了）。

B.脏读：一个事务读到另一个事务未提交的更新数据（A 和 B 事务并发执行，B 事务执行更新后，A 事务查询 B 事务没有提交的数据，B 事务回滚，则 A 事务得到的数据不是数据库中的真实数据。也就是脏数据，即和数据库中不一致的数据）。

C.不可重复读：一个事务读到另一个事务已提交的更新数据（A 和 B 事务并发执行，A 事务查询数据，然后 B 事务更新该数据，A 再次查询该数据时，发现该数据变化了）。

D. 覆盖更新：这是不可重复读中的特例，一个事务覆盖另一个事务已提交的更新数据（即 A 事务更新数据，然后 B 事务更新该数据，A 事务查询发现自己更新的数据变了）。

E.虚读（幻读）：一个事务读到另一个事务已提交的新插入的数据（A 和 B 事务并发执行，A 事务查询数据，B 事务插入或者删除数据，A 事务再次查询发现结果集中有以前没有的数据或者以前有的数据消失了）。

如何在 jdbc 中使用事务？

```
Connection conn = DriverManager.getConnection("url","用户名","密码");
conn.setAutoCommit(false);
conn.commit();
conn.rollback();
```

三层结构是什么？每一层怎么用？

三层结构是代码设计范型。

action

表示层，实现显示逻辑，与用户进行数据交互（包括数据合法性校验）

service

业务层，用于处理核心业务逻辑

dao

持久层，用于数据持久化操作（访问数据库）

单例模式，工厂模式，代理模式，适配器模式，观察者模式，模版模式分别是什么？

单例模式（Singleton）

在 Java 应用中，单例对象能保证在一个 JVM 中，该类只有一个实例存在。

懒汉模式

```
1.          //懒汉式单例类.在第一次调用的时候实例化自己
2.public class Singleton {
3.   private Singleton() {}
4.   private static Singleton single=null;
5.   //静态工厂方法
6.   public static Singleton getInstance() {
7.       if (single == null) {
8.          single = new Singleton();
9.       }
10.      return single;
11.   }
12.}
```

恶汉模式

```
1.//饿汉式单例类.在类初始化时，已经自行实例化
2.public class Singleton1 {
3.   private Singleton1() {}
4.   private static final Singleton1 single = new Singleton1();
5.   //静态工厂方法
6.   public static Singleton1 getInstance() {
7.      return single;
8.   }
9.}
```

工厂模式（Factory Pattern）

客户类和工厂类分开。消费者任何时候需要某种产品，只需向工厂请求即可。消费者无须修改就可以接纳新产品。缺点是当产品修改时，工厂类也要做相应的修改

即将对象的创建交给第三个类

代理模式（Proxy Pattern）

代理模式给某一个对象提供一个代理对象，并由代理对象控制对源对象的引用。代理就是一个人或一个机构代表另一个人或者一个机构采取行动。

适配器模式（Adapter Pattern）

把一个类的接口变换成客户端所期待的另一种接口，从而使原本因接口原因不匹配而无法一起工作的两个类能够一起工作。适配类可以根据参数返还一个合适的实例给客户端。

观察者模式（Observer Pattern）

观察者模式定义了一种一队多的依赖关系，让多个观察者对象同时监听某一个主题对象。这个主题对象在状态上发生变化时，会通知所有观察者对象，使他们能够自动更新自己。

模板模式（Template Pattern）

模板方法模式准备一个抽象类，将部分逻辑以具体方法以及具体构造子的形式实现，然后声明

一些抽象方法来迫使子类实现剩余的逻辑。不同的子类可以以不同的方式实现这些抽象方法，从而对剩余的逻辑有不同的实现。先制定一个顶级逻辑框架，而将逻辑的细节留给具体的子类去实现。

表单提交方式有几种？区别是什么？

表单提交方式为 GET 和 POST 方式，

对于 GET 和 POST 请求的区别，一般都认为有一下几点

1、GET 使用 URL 或 Cookie 传参。而 POST 将数据放在 BODY 中。

2、GET 的 URL 会有长度上的限制，则 POST 的数据则可以非常大。

3、POST 比 GET 安全，因为数据在地址栏上不可见。

Jquery 元素选择器有哪些？

JQuery 选择器包含：

1、基本选择器

a)ID 选择器（#id）

b)标签选择器(element)

c)类样式选择器(.class)

d)全部选择器（*）

2、层级选择器

a)ancestor descendant

b)parent > child

c)prev + next

d)prev ~ siblings

3、条件选择器

a):first

b):last

c):not(selector)

d):even

e):odd

f):eq(index)

g):gt(index)

h):lt(index)

i):header

j):animated

k):focus1.6+

l):contains(text)

m):empty

n):has(selector)

o):parent

p):hidden

q):visible

r):enabled

s):disabled

t):checked

u):selected

4、子元素选择器

a):nth-child

b):first-child

c):last-child

d):only-child

5、表单选择器

a):input

b):text

c):password

d):radio

e):checkbox

f):submit

g):image

h):reset

i):button

j):file

k):hidden

6、属性选择器

a)[attribute]

b)[attribute=value]

c)[attribute!=value]

d)[attribute^=value]

e)[attribute$=value]

f)[attribute*=value]

g)[attrSel1][attrSel2][attrSelN]

Jquery 访问 css　访问属性　访问 innerHTML 动态绑定事件

JQuery 访问 CSS

```
$().css("名","值")
$().css("名")
```

JQuery 访问属性

```
$().attr("名","值")
$().attr("名")
$().prop("名","值")
$().prop("名")
```

JQuery 访问 HTML 代码/文本

```
$().html("名")
$().text("名")
```

JSP 9 大内置对象是哪些？这九个内置对象分别对应 Servlet 中的什么 java 类型？

out	JspWriter
response	HttpServletResponse
request	HttpServletRequest
session	HttpSession
application	ServletContext
pageContext	PageContext
exception	Exception
config	ServletConfig 对象
page	当做 this，相当于当前的 jsp 实例

Servlet 生命周期是什么？

1.创建 Servlet 实例

由服务器创建(不需要用户去 new)

默认第一次请求到达时创建对象

配置了<load-on-startup>参数大于等于 0，服务器启动时创建对象

创建的实例是单例的

2.初始化

调用 Servlet 中 init(ServletConfig config)方法

服务器自动调用

3.当请求到达时调用 service(ServletRequest request,ServletResponse response)方法

对于 HttpServlet 提供了 service(HttpServletRequest request,HttpServletResponse response)方法

以及 doGet 和 doPost 方法(servcie 就包括了这两个方法)

4.当服务器销毁 Servlet 实例时

调用 destroy()方法

服务器自动调用

Request 对象获取请求参数的几种方法　分别返回什么类型？

```
String name = request.getParameter("name")
String[] hobbies = request.getParameterValues("hobbis");
Map<String,String[]> map = request.getParameterMap();
```

Jsp 4 个作用域对象是什么？

作用域	范围
pageContext	当前页面
request	同一个请求
session	同一个会话
application	整个应用

转发和重定向区别是什么？

转发

只能在同一个服务器内部进行转发

转发涉及到的多个组件，可以共享 request 和 response

浏览器只能看到 1 次请求和响应，所以浏览器的地址栏不会发生变化

可以访问 WEB-INF 下的资源

重定向

地址栏随着重定向地址的改变而改变

重定向的请求都是 get 请求,相当于在地址栏输入了一个新的地址

可以跨域访问

EI 表达式能做什么？

1.能访问作用域

pageContext

request

session

application

jsp 引擎先执行 pageContext.getAttribute(key)找数据

如果找不到，则依次 request,session,application 区域找数据

如果在其中某个区域找到数据

则不再向下查找

找到以后作为 html 输出到浏览器

如果所有区域都没有找到

则什么都不显示

也可以使用

pageScope

requestScope

sessionScope

applicationScope

访问指定区域的数据

el 表达式隐式对象(11 个)

pageScope	取得 page 范围的属性名称所对应的值
requestScope	取得 request 范围的属性名称所对应的值
sessionScope	取得 session 范围的属性名称所对应的值
applicationScope	取得 application 范围的属性名称所对应的值
pageContext	表示此 JSP 的 pageContext
param	相当于 ServletRequest.getParameter(String name)
paramValues	相当于 ServletRequest.getParameterValues(String name)
header	相当于 ServletRequest.getHeader(String name)

headerValues	相当于 ServletRequest.getHeaders(String name)
cookie	相当于 HttpServletRequest.getCookies()
initParam	如同 ServletContext.getInintParamter(String name)

2.访问对象中的属性

语法

${对象.属性}

这里属性指的是对应的 get/set 方法

取属性的数据实际上就是 get 方法

如果是一个对象的集合,想取出其中的对象

${emps[1]}

3.计算

在 el 表达式内部做计算

字符串比较使用 eq

${"aaa" eq "bbb" }

判断是否为空

${empty emp }

若为空返回 true

若不为空返回 false

4.获得请求参数

${param.参数名}

5.访问自定方法

Cookie 和 session 区别是什么？

cookie

数据保存在客户端

只能保存字符串（编码 ISO8859-1）

安全性较差

客户端分担了服务器的压力

浏览器可以禁用 cookie

cookie 大小有限制

session

数据保存在服务器端

保存任意类型对象

相对安全

所有数据在服务器内存，增加了服务器的压力

session 虽然使用 cookie 实现，但可以通过传 jsessionid 找到原来的 session（url 重写）

只要服务器内存够，随便使用

Mvc 分别是什么？一个请求到响应的过程是什么？

是一种软件架构的开发模式

将一个操作分成三种不同的部分

model：封装了业务逻辑，对于业务数据按一定的规则进行加工

view：封装显示逻辑

cotroller：协调视图和模型

请求不再直接访问模型和视图

而是统一交给控制器(所有请求进的都是控制器)

由控制器来分发给合适的模型来处理

模型处理之后，也不再返回个视图

而是交给控制器

由控制器选择合适的视图，将结果展示给用户

Ajax 是什么？如何通过 JavaScript 发送 ajax 请求？如何通过 jquery 发送 ajax 请求

Asynchronous javascript and xml

异步的 javaScript 和 xml

js 访问:

```
function hello(){
    var xmlHttpRequest = new XMLHttpRequest();

    xmlHttpRequest.onreadystatechange=function(){
        if(xmlHttpRequest.status==200 && xmlHttpRequest.readyState==4){
            var result = xmlHttpRequest.responseText;
            alert(result);
        }
    }

    xmlHttpRequest.open("get", "${pageContext.request.contextPath}/sayHello", true);
    xmlHttpRequest.send(null);


}
```

jQuery 访问

ajax

get

post

getJSON

load

Mybatis 是什么？

持久层框架

java 实现数据库操作只能是 JDBC,

mybatis 的底层也是实现的 JDBC

MyBatis 的前身就是 iBatis ，是一个支持普通 SQL 查询，存储过程和高级映射的优秀持久层(ORM)框架。

MyBatis 消除了几乎所有的 JDBC 代码和参数的手工设置以及对结果集的检索封装。

MyBatis 可以将向 PreparedStatement 中的输入参数自动进行输入映射，将查询结果集进行输出映射，映射成 java 对象。

Mybatis Config 配置文件的作用是什么？能配置些什么？

用来配置 Mybatis 的基本信息，能配置

settings 配置

properties 属性文件

typeAliases 类型的别名（实体类型别名）；

typeHandlers 类型句柄

ObjectFactory 对象工厂

plugins 插件

environments 环境

mappers 映射器

Mybatis Mapper 配置文件的作用是什么？

配置数据库里的表与实体类的映射关系以及 SQL 语句和 Dao 接口的映射关系

Mybatis resultMap 是什么？

MyBatis 中在查询进行 select 映射的时候，返回类型可以用 resultType，也可以用 resultMap，resultType 是直接表示返回类型的，

而 resultMap 则是对外部 ResultMap 的引用，用来定义列和属性的映射关系

Mybatis resultType 是什么？

MyBatis 中在查询进行 select 映射的时候，返回类型可以用 resultType，也可以用 resultMap，resultType 是直接表示返回类型

Mybatis 如何获取保存的数据的主键？

在 insert 中添加

useGeneratedKeys="true"

keyProperty="id"

keyColumn=""

三个属性

Mybatis 只需要写 dao 的接口？为什么可以调用方法？

程序员只需要写 dao 接口(Mapper)，而不需要写 dao 实现类，

由 mybatis 根据 dao 接口和映射文件中 statement 的定义生成接口实现代理对象。

jsp/servlet 关系

jsp 本质上 就是 Servlet

jsp 会由 jsp 引擎将其转换为 java 类，并编译成 class 文件。

tomcat 如何配置，配置文件在哪里，如何修改端口？

配置文件一般都在 conf 文件夹里，

主要有 server.xml，context.xml，tomcat_user.xml，web.xml 四个常用配置文件，

server 主要是服务器设置的，例如端口设置，路径设置

要修改端口的话，找到 server.xml 中 protocol 值为 http1.1 的 connector 标签修改 port 参数即可。

Hibernate 有几种配置文件？每种配置文件的作用是什么？

xxx.hbm.xml 映射文件 配置类和表的映射关系

hibernate.config.xml 总配置文件配置一些 hibernate 底层的配置

如何在 hibernate 中配置一对多关系？

一方中在 hbm 配置文件中配置<one-to-many>，多的一方配置<many-to-one>

Hibernate 一级缓存和二级缓存分别是什么？

一级缓存是 session 级别的。

也就是只有在同一个 session 里缓存才起作用。

而二级缓存是 sessionFactory 级别的。

其缓存对同一个 sessionFactory 生产出来的 session 都有效

Spring 的两个核心技术是什么？分别能做什么事情？

AOP 和 IOC.

IOC 在 Spring 里的实现是 Dependency Injection 依赖注入 就是说对象之间的依赖关系在后期通过配置文件（典型为 XML 文件）生成， Spring 里实现了两种注入方式：构造函数注入、 Setter 方法注入。我们可以这样理解这种技术带来的好处，前期我们只需要关注单个对象（组件）的功能实现，具体的业务实现是通过后期配置出来的，不同的配置可以产生不同的业务功能。

AOP 大大降低了对象之间的耦合程度，与 IoC 一样，能够通过后期的配置动态为对象增加新的特性，甚至能够为对象动态增加方法。在 Spring 下， AOP 的实现不需要借助专门的 AOP 定义语言，只需要普通的 Java 对象和 XML 配置文件即可

如何使用 spring 管理数据库事务？

使用事务管理器 transactionmanager

使用 AOP 技术将事务切面织入到我们的目标对象上

配置又包括两个版本：

1 transessionproxyfactrybean 来获得有事务的代理对象

2 使用 spring ts 的命名空间做<ts:advice>在通过 AOP:config 做事务的 advisor

也可以通过注解形式@Transactional 来配置事务。

如何实现 aop？

配置可以通过 xml 文件来进行，大概有四种方式：

1.配置 ProxyFactoryBean，显式地设置 advisors, advice, target 等

2.配置 AutoProxyCreator，这种方式下，还是如以前一样使用定义的 bean，但是从容器中获得的其实已经是代理对象

3.通过<aop:config>来配置

4.通过<aop: aspectj-autoproxy>来配置，使用 AspectJ 的注解来标识通知及切入点

也可以直接使用 ProxyFactory 来以编程的方式使用 Spring AOP，通过 ProxyFactory 提供的方法可以设置 target 对象, advisor 等相关配置，最终通过 getProxy()方法来获取代理对象

使用 spring 进行 web 项目整合的时候。需要在 web.xml 中配置哪些东西?

<context-param>
<param-name>contextConfigLocation</param-name>
<param-value>classpath:applicationContext.xml</param-value>
</context-param>配置 spring 配置文件的路径在 Classpath 目录下
<listener>
<listenerclass>org.springframework.web.context.ContextLoaderListener</listener-class>
配置监听
</listener>
<filter>
<filter-name>encodingFilter</filter-name>
<filterclass>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
<init-param>
<param-name>encoding</param-name>
<param-value>UTF-8</param-value>
</init-param>
</filter>
<filter-mapping>
<filter-name>encodingFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>配置乱码解决

SpringMVC 中有哪些注解？分别有什么作用？

1、@Controller

2、@RequestMapping

3、@RequestParam,@PathVariable

4、@CookieValue

5、@RequestBody

6、@autowrite、@Qualifie、@Rsource

7、@ModelAttribute

8、@SessionAttributes

9、@RequestHeader 获取请求的头部信息

10、@Repository

11、@Transactional

1、@Controller 用于标记在一个类上，使用它标记的类就是一个 SpringMVC Controller 对象。分发处理器将会扫描使用了该注解的类的方法，并检测该方法是否使用了@RequestMapping 注解。

如果只是使用注解的话，spring 无法管理的，所以在 springMvc 容器即配置文件中将此我们配置的文件位置告诉 spring，交给 spring 管理

配置如下：

```
<!--映射 HandlerMapping 和适配器 HandlerAdapter...其他映射信息-->
<mvc:annotation-driven></mvc:annotation-driven>
<!--扫包-->
<context:component-scan base-package="此处为你项目中配置了 controller 注解的类的包名
"></context:component-scan>
```

2、@RequestMapping 是一个用来处理请求地址映射的注解，可用于类或方法上。用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径，用于方法上则是表示此方法处理改地址的请求。

RequestMapping 注解有六个属性，三类进行说明。

1、 value， method；

value： 指定请求的实际地址，

method： 指定请求的 method 类型， GET、POST、PUT、DELETE 等；

2、 consumes，produces；

consumes：指定处理请求的提交内容类型（Content-Type），例如 application/json, text/html;

produces: 指定返回的内容类型，仅当 request 请求头中的(Accept)类型中包含该指定类型才返回

3、 params，headers；

params： 指定 request 中必须包含某些参数值是，才让该方法处理。

headers： 指定 request 中必须包含某些指定的 header 值，才能让该方法处理请求。

3、@RequestParam 将 request 中的参数绑定到我们的控制器的参数上

在@RequestParam 中除了指定绑定哪个参数的属性 value 之外，还有一个属性 required，

它表示所指定的参数是否必须在 request 属性中存在，默认是 true，表示必须存在，当不存在时就会报错。

在下面代码中我们指定了参数 name 的 required 的属性为 false，而没有指定 age 的 required 属性，这时候如果我们访问/requestParam.do 而没有传递参数的时候，

系统就会抛出异常，因为 age 参数是必须存在的，而我们没有指定。而如果我们访问/requestParam.do?age=1 的时候就可以正常访问，因为我们传递了必须的参数 age，

而参数 name 是非必须的，不传递也可以。

```
public String testRequestParam(@RequestParam(required=false) String name,
@RequestParam ("age") int age) {
    return age;
}
```

@PathVariable URL 是这样的：http://host:port/path/参数值

@RequestParam URL 是这样的：http://host:port/path?参数名=参数值

两者的作用都是将 request 里的参数的值绑定到 contorl 里的方法参数里

```
public @ResponseBody
User printUser(@RequestParam(value = "id", required = false, defaultValue = "0")
int id) {
        User user = new User();
        user = userService.getUserById(id);
        return user;
}


public @ResponseBody
   User printUser2(@PathVariable int id) {
          User user = new User();
          user = userService.getUserById(id);
          return user;
}

@RequestMapping(value="/user/{id:\\d+}",method = RequestMethod.GET)
public @ResponseBody
User printUser2(@PathVariable int id) {
      User user = new User();
      user = userService.getUserById(id);
      return user;
}
```

4、@CookieValue

@CookieValue 绑定 cookie 的值到 Controller 方法参数

例：

```
public String testCookieValue(@CookieValue("hello") String cookieValue,
        @CookieValue String hello) {
        System.out.println(cookieValue + "----------" + hello);
        return "cookieValue";
    }
```

在上面的代码中我们使用@CookieValue 绑定了 cookie 的值到方法参数上。上面一共绑定了两个参数，一个是明确指定要绑定的是名称为 hello 的 cookie 的值，一个是没有指定。

使用没有指定的形式的规则和@RequestParam 的规则是一样的，即在 debug 编译模式下将自动获取跟方法参数名同名的 cookie 值。

5、@RequestBody

@RequestBody 是指方法参数应该被绑定到 HTTP 请求 Body 上（常用的就是添加 jackson.jar，然后直接将对象转换成 json 字符串）

6、@autowrite、@Qualifie、@Rsource

@autowrite 以类型自动匹配注入

@Qualifie、@Rsource 当一个接口有多种实现类时，选择性注入

```
@Service("service"--相当于在 spring 中配置 bean 的 id)
public class EmployeeServiceImpl implements EmployeeService {
    public EmployeeDto getEmployeeById(Long id) {
```

```
            return new EmployeeDto();
        }
    }
    public class EmployeeInfoControl {

        @Autowired
        @Qualifier("service"--以类型名即 spring 中 bean 的 id 选择性注入)或者
@Resource("service"--以类型名即 spring 中 bean 的 id 选择性注入)
        EmployeeService employeeService;

        @RequestMapping(params = "method=showEmplayeeInfo")
        public void showEmplayeeInfo(HttpServletRequest request,
HttpServletResponse response, EmployeeDto dto) {
            #略
        }
    }
    spring 容器中配置
    扫包（将 service 交个 spring 管理）
    <context:component-scan base-package="service"></context:component-scan>
```

7、@ModelAttribute 应用于方法,将任何一个拥有返回值的方法标注上 @ModelAttribute，使其返回值将会进入到模型对象的属性列表中,基本数据类型不起作用，比如 int.

例如：

Java 代码

```
@ModelAttribute("items")//<——①向模型对象中添加一个名为 items 的

属性
public List<String> populateItems() {
        List<String> lists = new ArrayList<String>();
        lists.add("item1");
        lists.add("item2");
        return lists;
}
@RequestMapping(params = "method=listAllBoard")
public String listAllBoard(@ModelAttribute("currUser")User user,

ModelMap model) {
        bbtForumService.getAllBoard();
        //<——②在此访问模型中的 items 属性
        System.out.println("model.items:" + ((List<String>)

model.get("items")).size());
        return "listBoard";
}
```

在 ① 处，通过使用 @ModelAttribute 注解，populateItem() 方法将在

任何请求处理方法执行前调用，Spring MVC 会将该方法返回值以“items

”为名放入到隐含的模型对象属性列表中。
所以在 ② 处，我们就可以通过 ModelMap 入参访问到 items 属性，当执

行 listAllBoard() 请求处理方法时，② 处将在控制台打印

出“model.items:2”的信息。当然我们也可以在请求的视图中访问到模型

对象中的 items 属性。

8、Spring 允许我们有选择地指定 ModelMap 中的哪些属性需要转存到 session 中，以便下一个请求属对应的 ModelMap 的属性列表中还能访问到这些属性。这一功能是通过类定义处标注 @SessionAttributes 注解来实现的。@SessionAttributes 只能声明在类上，而不能声明在方法上。

例如

```
@SessionAttributes("currUser") // 将 ModelMap 中属性名为 currUser 的属性
@SessionAttributes({"attr1","attr2"})
@SessionAttributes(types = User.class)
@SessionAttributes(types = {User.class,Dept.class})
@SessionAttributes(types = {User.class,Dept.class},value={"attr1","attr2"})
```

9、@RequestHeader 获取请求的头部信息

10、@Repository 持久层注解，如果配置为（"xxx"）相当于给持久层类起别名

11、

事物传播行为介绍:

@Transactional(propagation=Propagation.REQUIRED) ：如果有事务，那么加入事务，没有的话新建一个(默认情况下)

@Transactional(propagation=Propagation.NOT_SUPPORTED) ：容器不为这个方法开启事务

@Transactional(propagation=Propagation.REQUIRES_NEW) ：不管是否存在事务,都创建一个新的事务,原来的挂起,新的执行完毕,继续执行老的事务

@Transactional(propagation=Propagation.MANDATORY) ：必须在一个已有的事务中执行,否则抛出异常

@Transactional(propagation=Propagation.NEVER) ：必须在一个没有的事务中执行,否则抛出异常(与 Propagation.MANDATORY 相反)

@Transactional(propagation=Propagation.SUPPORTS) ：如果其他 bean 调用这个方法,在其他 bean 中声明事务,那就用事务.如果其他 bean 没有声明事务,那就不用事务.

事物超时设置:

@Transactional(timeout=30) //默认是 30 秒

事务隔离级别:

@Transactional(isolation = Isolation.READ_UNCOMMITTED)：读取未提交数据(会出现脏读，不可重复读) 基本不使用

@Transactional(isolation = Isolation.READ_COMMITTED)：读取已提交数据(会出现不可重复读和幻读)

@Transactional(isolation = Isolation.REPEATABLE_READ)：可重复读(会出现幻读)

@Transactional(isolation = Isolation.SERIALIZABLE)：串行化

MYSQL：默认为 REPEATABLE_READ 级别

SQLSERVER：默认为 READ_COMMITTED

脏读 ：一个事务读取到另一事务未提交的更新数据

不可重复读 ：在同一事务中，多次读取同一数据返回的结果有所不同，换句话说，后续读取可以读到另一事务已提交的更新数据. 相反,"可重复读"在同一事务中多次读取数据时，能够保证所读数据一样，也就是后续读取不能读到另一事务已提交的更新数据

幻读 ：一个事务读到另一个事务已提交的 insert 数据

rollbackFor 该属性用于设置需要进行回滚的异常类数组，当方法中抛出指定异常数组中的异常时，则进行事务回滚

ollbackForClassName 该属性用于设置需要进行回滚的异常类名称数组，当方法中抛出指定异常名称数组中的异常时，则进行事务回滚

readOnly 该属性用于设置当前事务是否为只读事务，设置为 true 表示只读，false 则表示可读写，默认值为 false

propagation 该属性用于设置事务的传播行为

SpringMVC 模式下，请求相应的流程是什么？

1、 首先用户 发送请求—— >DispatcherServlet ， 前端控制器收到请求后自己不进行处理，而是委托给其他的解析器进行处理，作为统一访问点，进行全局的流程控制；

2、 DispatcherServlet —— >HandlerMapping ， HandlerMapping 将会把请求映射为 HandlerExecutionChain 对象（包含一个 Handler 处理器（页面控制器）对象、多个 HandlerInterceptor 拦截器）对象，通过这种策略模式，很容易添加新的映射策略；

3、 DispatcherServlet —— >HandlerAdapter ， HandlerAdapter 将会把处理器包装为适配器，从而支持多种类型的处理器，即适配器设计模式的应用，从而很容易支持很多类型的处理器；

4、 HandlerAdapter —— > 处理器功能处理方法的调用， HandlerAdapter 将会根据适配的结果调用真正的处理器的功能处理方法，完成功能处理；并返回一个 ModelAndView 对象（包含模型数据、逻辑视图名）；

5、 ModelAndView 的逻辑视图名—— > ViewResolver ， ViewResolver 将把逻辑视图名解析为具体的 View，通过这种策略模式，很容易更换其他视图技术；

6、 View —— > 渲染 ，View 会根据传进来的 Model 模型数据进行渲染，此处的 Model 实际是一个 Map 数据结构，因此很容易支持其他视图技术；

7、 返回控制权给 DispatcherServlet ， 由 DispatcherServlet 返回响应给用户，到此一个流程结束。

点餐系统概要说明

开发环境: ubuntu 系统
开发语言：java
开发工具: MyEclipse8
数据库:mysql
使用的技术:swing,jdbc,自封装的 factory .jar 包
开发周期:5 天
开发人数:1 人

点餐系统是餐厅提供给客户点菜服务与后台菜品管理的系统,
一共分为两个系统.前台的点菜系统(order),后台菜品管理系统(orderManager)
前台给客户使用，用于点餐和加菜。
后台是管理系统，用于管理所有的菜品类型 ， 所有的菜，所有的订单等

P2P 金融理财系统

开发环境: ubuntu
开发语言:java
开发工具: MyEclipse 10
数据库:mysql/oracle
开发人数:1-2 人
开发周期:5 天
服务器： Tomcat6.0

P2p 金融理财是个人与个人间的小额借贷交易，一般需要借助电子商务专业网络平台帮助借贷双方确立借贷关系并完成相关交易手续。借款者可自行发布借款信息，包括金额、利息、还款方式和时间，实现自助式借款。借出者根据借款人发布的信息，自行决定借出金额，实现自助式借贷。

主要功能包括
前台系统功能简介:
进入前台界面首先以列表形式展示所有融资产品,并且提供分页。在抬头有产品类型,贷款周期，贷款利率，贷款规模四个查询条件点击不同的查询条件可以按相应条件查询。

当用户点击相应产品的查询详情按钮后,会跳转详情界面左边显示产品详情,右边显示申请人数和申请人列表。点击申请跳出申请界面输入数据后提交申请成功,刷新该界面。

点击每日新闻连接进入新闻展示界面,新闻做列表展示。
后台系统功能简介:
进入后台登录界面输入帐号密码即可登录。（没有后台登录用户管理的模块,所以后台用户数据是自己手动插入数据库的）

1.融资产品管理:

在线申请产品管理

2.新闻管理

3.元数据管理:（1）产品周期管理

（2）产品类型管理

4.企业管理（1）企业管理可以展示所有注册企业信息并且添加

（2）企业报表按年份展示企业融资情况。

机票分销管理系统

开发环境: ubuntu 系统

开发语言：java

开发工具: MyEclipse8

数据库: oracle

使用的技术: spring+springmvc+hibernate

开发周期: 13-14 天

开发人数: 7-8 人一组

整个系统由三个子系统组成，票务系统，分销系统，销售系统。

三个子系统之间 通过 WebService 的 Hessian 实现数据通信

票务系统主要实现

航班管理、票务管理以及用户订单的保存

分销系统

这是整个系统的核心，主要用于机票对分销商的分配以及其他基础功能模块

如：

用户管理

权限管理

分销商管理

分配计划管理

分销商报表

机票追加

元数据管理： 分销商等级管理等

销售系统：

用于机票的销售，该子系统原则上来说，每个分销商都应该拥有自己的销售系统

但是我们在做开发的时候，只做一个实现，用于模拟完整的数据流程

用户购买完机票后，通过 webservice 与上行系统进行数据交互，将购买信息提交给分销平台，

分销平台记录购买信息后，最终将信息提交给票务系统。