# C#与Java的对比

1，命名空间与包

C#为了把实现相似功能的类组织在一起，引入了命名空间的概念（namespace）

Java中与此对应的东西叫做包（package）

2，类的访问控制方面的不同

C#只有两种：public和默认(同internal)

public能被所有的类（同项目中和不同项目中）访问

internal（在class关键字前不加控制符时默认为internal），表明类只能在同一项目中访问

Java也只有两种：public和默认

public能被所有的类访问

默认（在class关键字前不加控制符时）只能被同一包中的所有类访问

3，类成员的访问控制

C#中有四种：public,protected,private（默认）,internal（注意这里internal和默认是不同的）

public能被所有的类访问

protected只能被子类访问

private（也即不写任何控制符时的默认情况下）只能类内部访问

internal能被同一项目中的类访问

Java也有四种：public,protected,private和默认

public能被所有的类访问

protected既能被同一个中的其他类访问，也可以被不同包中的子类访问

private只能在类内部使用

默认可以被这个包中的其他类访问，如果一个子类与父类位于不同的包中，子类也不能访问父类中的默认访问控制成员

4，C#中类的继承用通过冒号:实现，在Java中用extends

C#中实现接口通过冒号:实现，在Java中用implements

C#中密封类用sealed实现，在Java中用final

C#中常数用const实现，在Java中用final

C#中属性用set,get代码块实现，在Java中一般用类似于C#中的字段代表属性，或者用setter,getter构造器实现

C#中有部分类（partial）的概念，Java中没有

C#中有readonly修饰属性只读，Java中没有

C#中有virtual和override修饰虚方法和重写方法，Java中没有，Java中默认父类中的方法都是virtual的

Java中有static{},synchroized{}代码块的概念，C#中没有

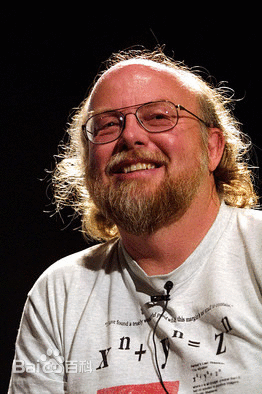
Java中有标签（如labelA:）的概念，C#中没有

C#中子类调用父类的方法用base.method()，Java中用super.method()

C#中用is来判断一个实例是否是某一个类的，Java中用instanceof

C#中用foreach(int i in array)来对数组中每一个元素的遍历，Java中用for(int i : array)

# 8月29日



**JAVA之父：**

在1984年到2010年间，供职于Sun微系统公司。

在2010年甲骨文收购Sun后不久，宣布离职，并在2011年初加入谷歌。

2011年8月30日，离开谷歌，加入开展海洋探测业务的机器人制造公司Liquid Robotics，任首席软件架构师。负责传感器软件开发和自主导航设计，数据中心海量数据处理。

命令行常用命令：

dir（directory）列出当前目录下的文件以及文件夹

cd（change directory）: 进入指定目录

cd.. : 退回到上一级目录

cd \: 退回到根目录

**实验步骤：**

1、安装JDK（jdk-7u25-windows-i586.exe）；

2、设置path环境变量：C:\Program Files\Java\jdk1.7.0\_25\bin

3、编写源代码文件Test.java，存放到d:\

4、在命令行模式中切换到Test.java所在文件夹d:\

d:

回车符

5、运行javac Test.java

6、运行 java Test

代码编辑器统一使用Notepade++

计算机系统的内存：

内存是一片存储空间，其基本存储单元是字节byte(8个二进制位)。这些基本的存储单元需要进行编号，这些编号就是内存地址。编号的范围就是内存寻址范围。

如果系统有4位地址线，则寻址空间为2的4次方，即16

**JAVA数据类型**：

1)基本数据类型byte char short int long float double boolean

2)引用类型接受对象引用

\*内存地址：变量、对象、各种数据结构在内存中的位置编号

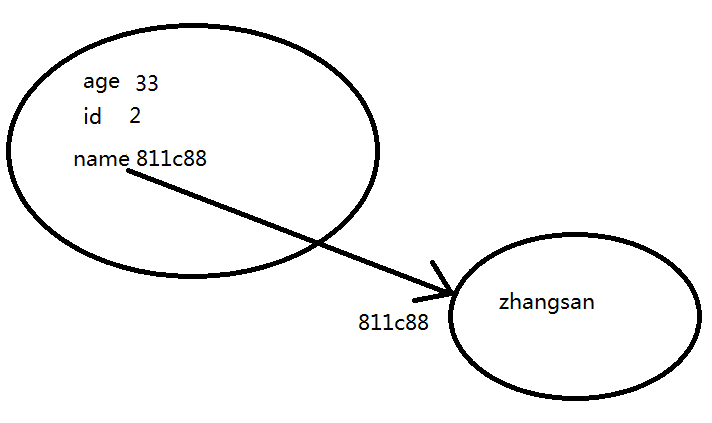
Person p = null;

p = new Person();

p.age = 33;

p.id = 2;

p.name = "zhangsan";



交换两个整数的值：

int x = 7, y = 9;

int z = x;

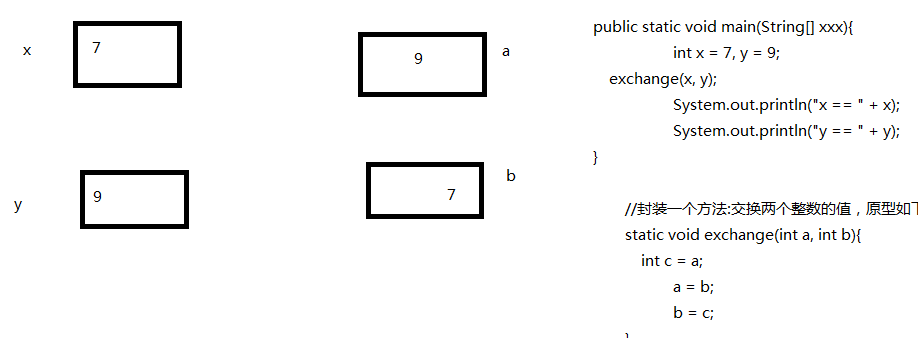
x = y;

y = z;

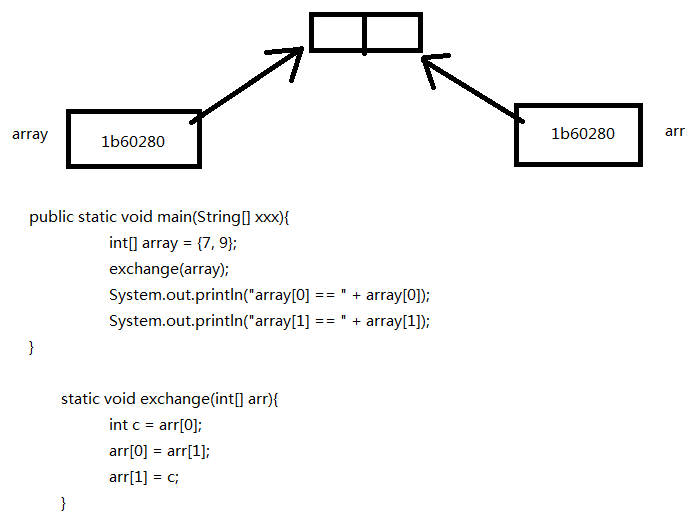
**方法调用时的参数传递**

方法调用时，会发生参数传递，也就是实参的值复制给形参。实参和形参分别有各自独立的内存空间。

参数为基本类型：



参数为引用类型：



# 9月2日

**方法重载Overload**

为啥返回值不参与方法重载的判断？

**冒泡排序算法：**

无序的原数组：{17,7,6,3,9,56,2,23,5,44}

算法思路：

第一步：把最大的那个数，放到数组最后一个单元中。具体实现：

从头开始，把相邻两个数，进行比较，如果前面的数大于后面的那个数，则把两个数进行交换。比较次数为9

{7,17,6,3,9,56,2,23,5,44} 第1次比较

{7,6,17,3,9,56,2,23,5,44} 第2次比较

{7,6,3,17,9,56,2,23,5,44} 第3次比较

{7,6,3,9,17,56,2,23,5,44} 第4次比较

{7,6,3,9,17,56,2,23,5,44} 第5次比较

{7,6,3,9,17,2,56,23,5,44} 第6次比较

{7,6,3,9,17,2,23,56,5,44} 第7次比较

{7,6,3,9,17,2,23,5,56,44} 第8次比较

{7,6,3,9,17,2,23,5,44,56} 第9次比较

第二步：把第二大的那个数，放到数组最后倒数第二个单元中。具体实现：

从头开始，把相邻两个数，进行比较，如果前面的数大于后面的那个数，则把两个数进行交换。比较次数为8

{6,7,3,9,17,2,23,5,44,56} 第1次比较

{6,3,7,9,17,2,23,5,44,56} 第2次比较

{6,3,7,9,17,2,23,5,44,56} 第3次比较

{6,3,7,9,17,2,23,5,44,56} 第4次比较

{6,3,7,9,2,17,23,5,44,56} 第5次比较

{6,3,7,9,2,17,23,5,44,56} 第6次比较

{6,3,7,9,2,17,5,23,44,56} 第7次比较

{6,3,7,9,2,17,5,23,44,56} 第8次比较

。。。。

。。。。

。。。。

第九步：把第9大的那个数，放到数组最后倒数第9个单元中。具体实现：

从头开始，把相邻两个数，进行比较，如果前面的数大于后面的那个数，则把两个数进行交换。比较次数为1

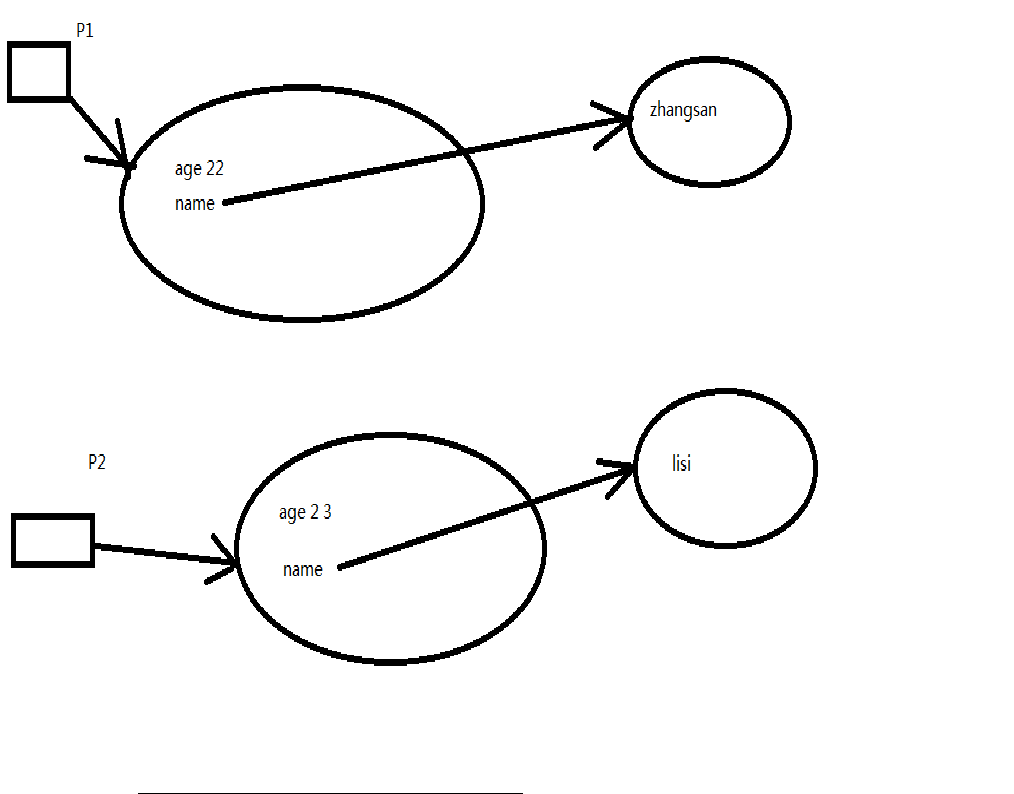
# 9月5日

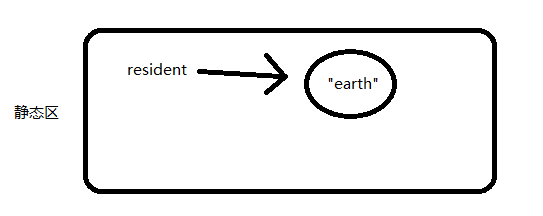
什么是垃圾对象（数组）?

没有一个有效的引用指向的对象（数组）。

# 9月7日

静态成员的理解：





//单例设计模式：设计一个类，其对象实例最多只能生成一个。

class Moon{

static Moon m = null;

private Moon(){

}

static Moon getInstance(){

if(m == null){

m = new Moon();

}

return m;

}

}

# 9月9日

字符编码：ASCII ---------- 1 byte

Unicode ----------- 2 bytes

char 类型有效范围: 0---------2的16次方减一

# 9月12日

1. String 工具类和 char[] 数组的关系：String类有一个类型为char[]的成员变量（即一个字符数组），String类对字符串的存储以及各种操作就是通过这个字符数组来进行的。

作业：利用String类解析：协议，主机，端口号，资源路径

http://www.sina.com.cn:8080/sports/index.jsp

胡老师教学群2：183887234

# 9月14日

***作业：***

1）基础题：用String 工具类，从以下这个网络资源路径（URL）解析出，协议，主机号，端口号，资源路径。

[http://www.sina.com.cn:80/index.html](http://www.sina.com.cn/index.html)



String url = "ftp://www.baidu.com:88899/sports/index.jsp";

**int** index1 = url.indexOf("://");

System.*out*.println(index1);

String protocol = url.substring(0, index1);

System.*out*.println("protocol:" + protocol);

//int index2 = url.indexOf(":", index1 + 3);

**int** index2 = url.lastIndexOf(':');

System.*out*.println(index2);

String host = url.substring(index1 + 3, index2);

System.*out*.println("host:" + host);

**int** index3 = url.indexOf("/", index1 + 3);

System.*out*.println(index3);

String port = url.substring(index2+1, index3);

System.*out*.println("port:" + port);

String path = url.substring(index3 + 1);

System.*out*.println("path:" + path);

1. 提高题：DIY自己的String工具类-------------MyString，并提供以下方法：

MyString(char[] str)

char charAt(int index)

返回指定索引处的 char 值。

intindexOf(intch)

返回指定字符在此字符串中第一次出现处的索引。

int length()

返回此字符串的长度。

booleanstartsWith(char prefix)

测试此字符串是否以指定的前缀开始。

boolean contains(char c)

当且仅当此字符串包含指定的 char 值序列时，返回 true。

MyString replace(char target, char replacement)

使用指定的字面值替换序列替换此字符串所有匹配字面值目标序列的子字符串。

1. StringBuilder和String的区别

String对象的内容，一旦初始化就不能被修改了。

虽然String类提供了replace，toLowerCase() 等方法，但这些方法都没有修改本身的字符串内容，而是返回了一个新的String对象。

通过查看String类的源码，发现：1）存放字符串的成员变量char数组，是final的，并且是private的；2）String类也没有通过任何一个可以修改这个char数组的成员方法。所以我们无法改变String的内容。

JDK为什么这样设计String类？为了内存优化！因为：1）字符串类型数据大量存在于我们的内存；2）大部分情况下，我们对字符串的操作都是读操作，而非修改操作。所以对于若干个内容完全一样的字符串，就用一个String对象来表示，而不会占用若干个String对象了，这样节省了内存开销。

StringBuilder类则能够真正修改自己的字符串。

在需要频繁修改、或者拼接字符串的场合，应该使用StringBuilder类，而不要用String类。为什么？因为用String类，会导致大量的String对象的生成（以及随后的销毁），消耗大量系统资源。

***作业***

设计代码来验证，上面阐述的关于StringBuilder和 String的区别比较。

# 9月18日

面向对象三大基石：封装继承多态

封装：为了代码的复用，将功能逻辑封装到类中，类的成员变量表示属性，类的成员方法表示行为。

多态三要素：

1. 子类重写父类的方法；
2. 父类引用接受子类对象实例；
3. 通过父类引用调用那个被重写的方法，运行子类重写的那个版本（而非运行父类的版本）。

思考：Java为什么要支持多态？

只有子类、父类之间才能强制转换！

# 9月23日

方法的返回值不参与判断是否是方法重载。为什么？因为调用方法的时候，不一定接收返回值，从而编译器无法据此进行绑定。

\*方法重载 Overload. （编译时绑定静态绑定）

某个类中，两个（或者若干个）成员方法的方法名一样，但参数类型不一样，或者参数的个数不一样。

多态（运行时绑定动态绑定）

必须进行方法重写(Override)：子类把父类的方法重新实现了一次。

\*什么时候使用抽象方法

某个类的某种行为无法具体确定，只有在子类中才能做具体定义。可以把该行为定义

为抽象方法，等待它的子类去具体定义该方法。

\*一个类一旦有一个抽象方法（用abstract关键字修饰，并且没有方法体的方法），

它就成为了抽象类，必须用abstract关键字修饰

\*某个类一旦继承了某个抽象类（或者实现某个接口），则必须做出两种选择：

要么把自己声明为抽象的（不实现父类的抽象方法），要么实现父类的抽象方法。

# 9月26日

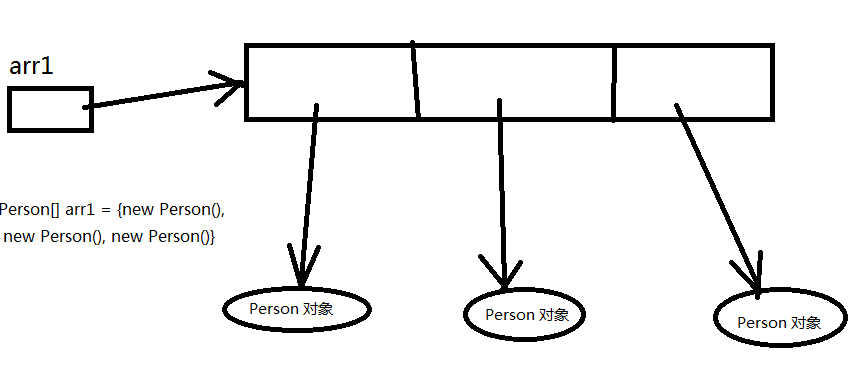
阶段性项目：学生成绩系统，包含一下功能模块：

成绩录入模块：从命令行录入学生成绩信息（学号，性别，年龄，各科成绩（数学，物理，网络，Java））

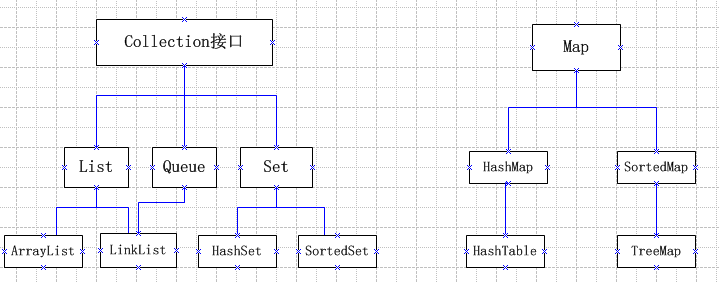
成绩处理模块：按总成绩排序。

成绩输出模块：在控制台输出学生成绩信息。

对象数组的内存模型：



容器框架：



1. 泛型的引入、概念、使用

数组提供了编译时的类型安全，但缺乏灵活性；

集合能够适配各种类型（因为其数据域定义为Object类型），但不能提供编译时的类型安全检查。Generic（泛型）解决了这个问题。

1. 面向对象思想的基本元素：封装、继承、多态。运用这些思想进行软件设计，在大量的软件开发实践中，有些固定的设计套路，有先辈对这些套路进行了总结，这就是设计模式（比如：单例模式，观察者模式，代理模式，等等）。

# 9月28日

Object 类的equals() 方法是比较两个对象的引用。子类可以重写它，以定制自己的比较规则。一般来说，比较规则都是比较两个对象的成员变量的值。

Object的hashCode方法，返回一个整数，它唯一地标识一个对象。最通常的做法就是返回对象的内存地址（为了安全，需要把对象内存地址转换为对象引用）。子类可以重写hashCode方法，采用的哈希算法：

1）内容一样的对象(equals方法返回true)，哈希值一定一样（hashCode()返回值一样）。

2）尽量满足：内容不同的对象，其哈希值尽量不一样。

思考：

结合String类的hashCode方法的实现思路，为Person类重写hashCode方法。

# 9月30日

哈希(散列)算法的数学理论基础:

所有散列函数都有如下一个基本特性：如果两个散列值是不相同的（根据同一函数），那么这两个散列值的原始输入也是不相同的。这个特性是散列函数具有确定性的结果，具有这种性质的散列函数称为单向散列函数。但另一方面，散列函数的输入和输出不是唯一对应关系的，如果两个散列值相同，两个输入值很可能是相同的，但也可能不同，这种情况称为“哈希碰撞”，这通常是两个不同长度的输入值，刻意计算出相同的输出值。输入一些数据计算出散列值，然后部分改变输入值，一个具有强混淆特性的散列函数会产生一个完全不同的散列值。

HashSet的 add 方法，首先会调用传入的对象的hashCode() 方法，逐个和集合中已有对象的hashCode方法的返回值进行比较，如果都不等，直接把该对象加入HashSet; 如果和某个对象的hashCode返回值相等，则继续调用equals方法进行判断，如果相等则不加入，否则加入。

## 10月9日

作业：词频统计：统计文本中，每个单词出现的次数。

原始数据：hello world hello java math tom

输出：hello 2

world 1

java 1  
 tom 1

math 1

用两种方式实现：

1. 不用集合（只用面向对象知识，数组，链表）；
2. 用集合。

## 10月12日

ArrayList，LinkedList元素可重复，元素遍历的顺序就是之前元素添加顺序；

HashSet元素不可重复，元素遍历的顺序是随机的；

TreeSet可以对元素（Person类型的对象）进行排序，并且程序员可以自定义比较规则。自定义比较规则有两种方式：

1. Person 实现Comparable接口，并重写compareTo() 方法；
2. 为Person类型定制一个比较器，这个比较器要实现 Comparator接口。---------- 该方案实际上用到了策略设计模式。该方案更好，因为它不用修改Person类型源码。

作业：

词频统计的结果，分别按照单词的字母升序、降序打印出来。采用两种排序方案：

1. 用TreeSet/Comparable/Comparator 实现；（基础题）
2. 用原始的冒泡排序实现。（选作题）

## 10月14日

HashMapTreeMap

HashMap是无序的，而TreeMap是有序的（用KEY排序）

所以，如果要把某个类型作为KEY装入TreeMap，比如保证该类型可比较。

## 10月17日

异常分两种：

RuntimeException（）可以不try catch

其他 Exception 必须try catch, 否则无法通过编译

# 10月17日

1. IO流：内存和输入输出设备（IO设备）之间的数据传递
2. 按数据流向分：输入流输出流，按数据格式：字节流（二进制数据）字符流（文本数据）
3. 字符编码怎么回事？

如果仅仅是英文，ASCII编码没有问题。但还有其他国家地区语言

中文：GB2312 GB18030 BIG5 GBK

韩文：EUC-KR

日文：SJIS

各搞一套，在本地使用没有问题，一旦出现在网络中，由于不兼容，互相访问就出现了乱码现象。

为了统一UNICODE UTF-32 UTF-16 UTF-8

JAVA 内部是unicode编码

编码在线查询：<http://bm.kdd.cc/>

1. 对IO进行处理的类库，JAVA最优秀

字节流InputStreamOutputStream

字符流 Reader Writer

文本文件输入流：FileReader

文本文件输出流：FileWriter

二进制文件输入流：FileInputStream

二进制文件输出流：FileOutputStream

1. 文本文件输出流FileWriter：把内存中的文本写到硬盘文件上。
2. 作业：

1）在内存中建一个String（字符串）对象，其内容为

<html>

<body>

<h1>我的主页</h1>

<font size="13" color="red">欢迎访问</font>

</body>

</html>

把这些文本写到磁盘上一个文件，命名为 index.html

## 10月19日

FileReader文本文件输入流

作业：用FileReader、FileWriter实现文本文件复制。

FileReader中有个方法用于读取单个字符数据：

publicint**read**()throws [IOException](mk:@MSITStore:E:\Tools\JDK_API_1_6_zh_CN.CHM::/java/io/IOException.html)

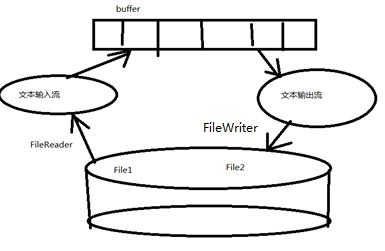
该方法的返回值设为int, 而不能是char，为啥？

文本数据可以用二进制流处理（InputStream, OutputStream的子类），但如果涉及字符显示等场合，就必须自己处理转码（所以为了方便，就直接用文本流工具类---------Reader，Writer）；

二进制数据不能用文本流来处理，因为文本流会对二进制数据进行转码（这种转换是毫无意义，并且经常找不到对应的字符编码）。

## 10月21日

文件拷贝



## 10月26日

FileInputStreamFileOutputStream

FileInputStream中有个方法用于读取单个字节数据：

publicint**read**()throws [IOException](mk:@MSITStore:E:\Tools\JDK_API_1_6_zh_CN.CHM::/java/io/IOException.html)

该方法的返回值设为int, 而不能是byte为啥？提示，考虑-1的补码表示。计算机系统内部采用补码表示数。

两种增强工具流：缓冲流转换流

1）缓冲流（带缓冲区的流，增加读写效率）：

BufferedReaderBufferedWriterBufferedInputStreamBufferedOutputStream

这些缓冲流内部管理了一个缓冲数组，使用缓冲流进行的读写操作，其实是对缓冲数组进行读写操作，而不是IO设备了，所以读写效率会大大提升。

缓冲数组管理的细节：

对于输入流（对于输入流BufferedReaderBufferedInputStream），先一次性从输入设备读数据填满缓冲数组，然后之后程序的读操作，都读的是这个缓冲数组，如果缓冲数组中的数组都读取完毕，则缓冲数组会再向输入设备读取数据，一次性填满缓冲区，如此往复。

对于输出流（对于输入流BufferedWriterBufferedOutputStream），程序对流的写操作，都写到了这个缓冲数组，如果缓冲数组写满了，则缓冲数组会一次性把数据写到输出设备中，如此往复。

缓冲流还提供了一些有用的操作。

2）转换流（字节流转成字符流）：InputStreamReader OutputStreamWriter

## 10月28日

1. 系统的标准输入流，输出流

System.in System.out.

课题作业：

从键盘反复录入数据，并将数据写到一个文件中。

# 10月31日

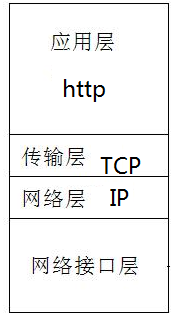
1、程序员的论坛：<http://bbs.csdn.net/><http://stackoverflow.com/>

mobilecloud.sinaapp.com

JAVA 视频教程毕向东

<JAVA编程思想><高效JAVA编程>

2、TCP/IP网路协议栈

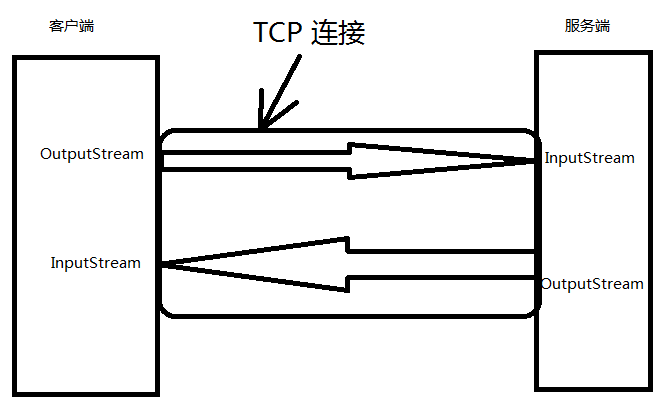


4、TCP通讯模型

Socket（插座）通信：把底层复杂的网络通信逻辑封装成库函数（C语言），或者类库（JAVA，C#）

Socket分两种：TCP Socket, UDP Socket

TCP 链接的概念：就是客户端和服务器之间的通信管道，这个管道的通信是双向的。



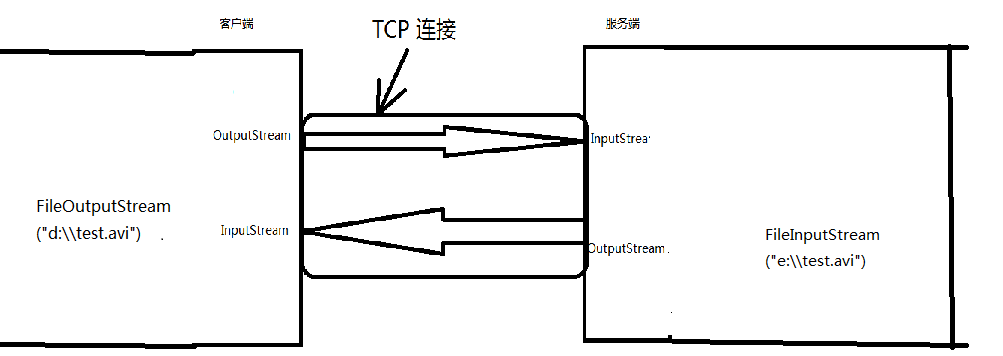
作业：

1. 理解课堂代码；
2. 在课堂代码基础上添加功能：

文件上传：将客户端的某个文件，通过TCP连接，发送到服务器上，并存为一个文件；

文件下载：将服务器的某个文件，通过TCP连接，发送到客户端上，并存为一个文件。

下图为文件下载的原理图：

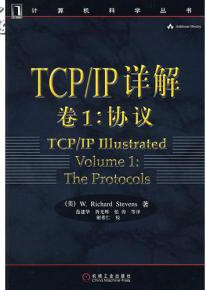


TCP 协议，是基于连接的协议。Server支持的连接数是有限的，连接不用的时候，需要关闭，以释放网络IO资源。

TCP/IP 的发明者：Vinton G. Cerf

 1970年代在业界普及

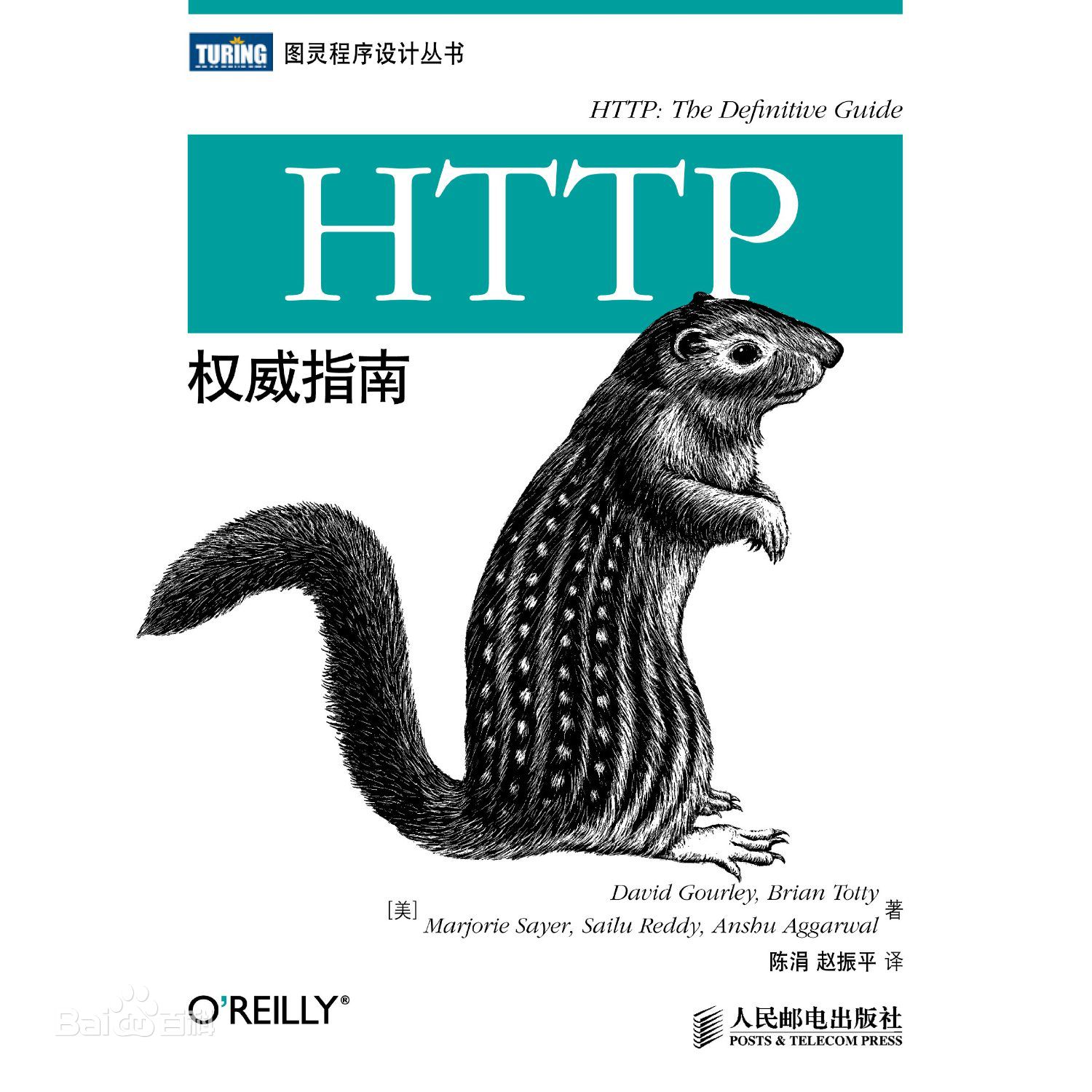
TCP/IP的经典教材：TCP/IP详解卷1 协议 W.Richard Stevens



http协议作者 Tim Berners-Lee 1990年发明



http经典教材《HTTP权威指南》 by Gourley



## 11月2日

Socket通讯的本质：跨进程通讯手段。

如果是本机内的不同进程之间，则Socket通讯就是本机内部的跨进程通讯；

如果是网络不同主机上的进程之间，则Socket通讯就是网络通讯。

网络通讯要解决的问题就是：网络上不同主机之间分享资源（文本，视频，音频。。。）。一般采用客户机-服务器（Client-Server）模型。客户机发出访问请求，服务器应答访问请求。基于TCP Socket的通讯模型：

1）客户机发出TCP连接请求，服务器接收确认连接请求；

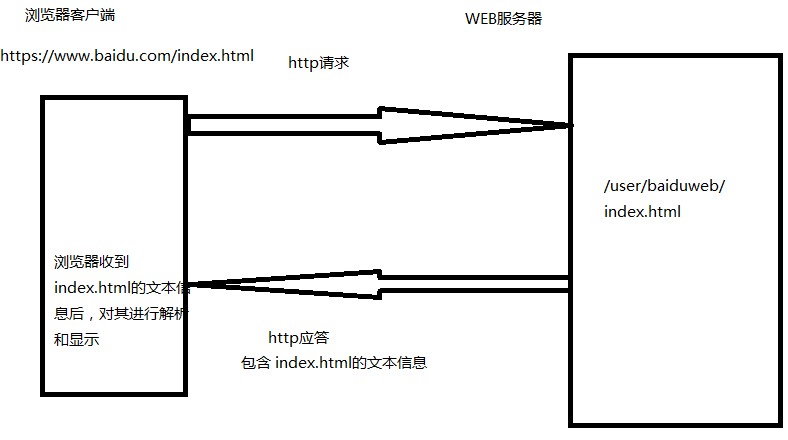
2）客户机发出具体的访问意图（要访问的资源的URL，等等）；

3）服务器解析客户机发来的访问意图，并组织数据，回送给客户端。

HTTP服务器框架：Tomcat，Apache，JBoss，WebService，Jetty，可以基于这些框架开发各种HTTP服务器应用：各种网站（新浪，阿里巴巴主页，，），各种服务器应用（QQ服务器，微信服务器，淘宝服务器）。

HTTP 客户端：浏览器（IE，FireFox，Chrome, Opera），各种客户端APP（ qq客户端，微信客户端，淘宝购物客户端）。

下图为浏览器客户端和WEB服务器应用交互的基本原理图：



浏览器请求一个复杂的网页，需要多次HTTP事务。

一次HTTP事务由一个HTTP请求（request）和一个HTTP应答（response）构成。

一个请求就是一个HTTP 请求协议包；一个应答就是一个HTTP应答协议包。

协议包由协议头和协议体构成。

一次HTTP事务访问一个网络资源，而网络资源是由URL标识的。

比如百度网页的URL: <http://www.baidu.com/index.html>

主页上的LOGO图片的URL: <https://www.baidu.com/img/bd_logo1.png>

请求包的格式：

**GET /Login/LoginCl?user\_name=zhang&pass\_word=123 HTTP/1.1**

Accept: image/jpeg, application/x-ms-application, image/gif, application/xaml+xml, image/pjpeg, application/x-ms-xbap, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, \*/\*

Accept-Language: zh-CN

User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3)

Accept-Encoding: gzip, deflate

Host: www.sina.com.cn

Connection: Keep-Alive

user\_name=huheng&pass\_word=123

应答包的格式：

**HTTP/1.1 200 OK**

Content-Type: text/html

Vary: Accept-Encoding

X-Powered-By: shci\_v1.03

Content-Encoding: gzip

Server: nginx

Date: Fri, 13 Nov 2015 00:12:15 GMT

Last-Modified: Fri, 13 Nov 2015 00:10:05 GMT

Expires: Fri, 13 Nov 2015 00:13:15 GMT

Cache-Control: max-age=60

Age: 43

Content-Length: 116632

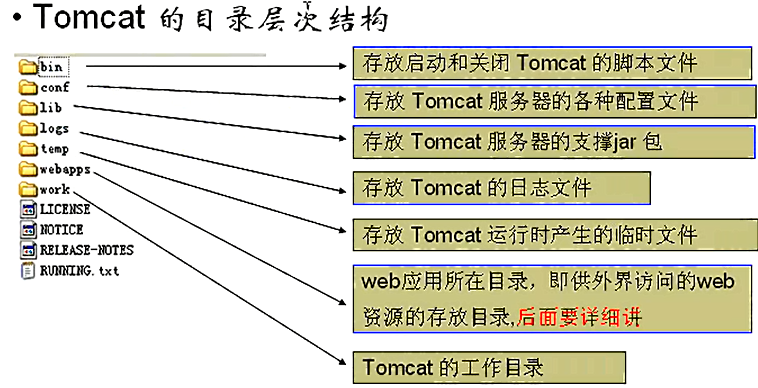
X-Cache: HIT from ctc.cd.1cf1.33.spool.sina.com.cn

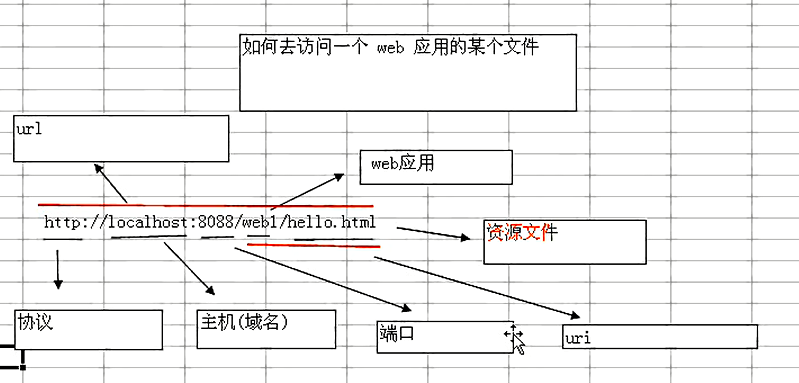
FAILED/SUCESS

一款流行的HTTP服务器 TOMCAT

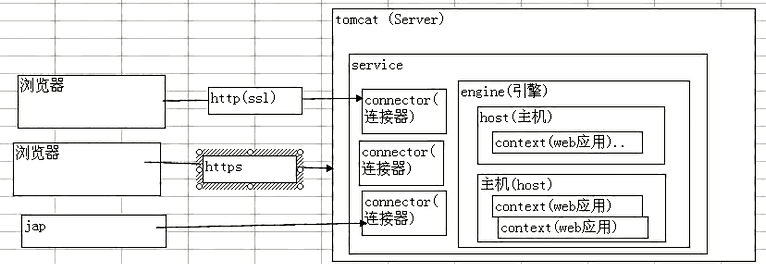
<http://tomcat.apache.org/>

apache-tomcat-7.0.65-src\java\org\apache\tomcat\util\net\DefaultServerSocketFactory.java

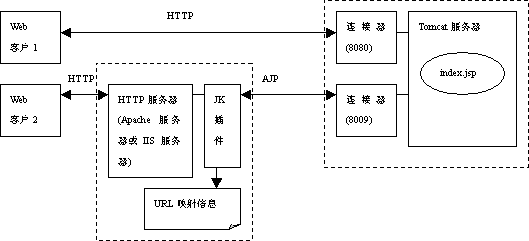




tomcat 体系架构



第一连接器监听8080端口，负责建立HTTP连接。在通过浏览器访问Tomcat服务器的Web应用时，使用的就是这个连接器。第二个连接器监听8009端口，负责和其他的HTTP服务器建立连接。在把Tomcat与其他HTTP服务器集成时，就需要用到这个连接器。 Web客户访问Tomcat服务器上JSP组件的两种方式如图22-1所示



静态网页：一个只包含静态信息的html文档。

动态网页：既有静态信息，又有动态信息的网页，则必须使用Java Servlet进行开发。

命令行编译、运行Servlet

实验步骤（具体请参照课堂项目: webapps目录下的eleWeb）

1、在webapps下建立项目自己的项目目录，在项目目录下再建立如下目录层次结构：

WEB-INF =》classes、web.xml

1. 把写好的类TestServlet放在classes下，包括类的包
2. 把apache-tomcat-7.0.72\lib\servlet-api.jar所在路径添加到classpath中，

4、在命令行执行编译：javac–d .TestServlet.java

5、在web.xml中配置servlet

6、重启tomcat服务器（执行bin目录下的 startup.bat）

7、在浏览器中访问该Servlet

TOMCAT 环境配置：

解压后，运行 bin\startup.bat

MyEclispe 破解版

百度“myeclipse tomcat JAVA WEB 开发环境搭建”

**11月11日**

用MyEclipse开发Servlet

将项目部署好后，启动tomcat后报错:

java.lang.NoClassDefFoundError: org/apache/juli/logging/LogFactory

解决方法：

一、打开myeclipse，Preferentces->MyEclipse->Servers->Tomcat->Tomcat 6.x

,载入Tomcat7，在"Paths"下，点击“Add JAR/ZIP”,加载tomcat7下的……/bin/tomcat-juli.jar。点击“OK”。

登陆成功后，切换到一个主页面，这个主页面往往很复杂，我们最好单独写一个servlet来定义这个界面--MainPage。如何切换到这个新的主界面? 有两种方式可选：

1、重定向

response.sendRedirect(“/TestServlet/MainPage”) 的作用：在响应头加了一个Location属性，指定了一个资源的URL

Location: <http://127.0.0.1:8088/TestServlet/MainPage>

浏览器收到这响应后，解析出Location属性，就会向这个URL发出一次新的HTTP请求。

可重定向很灵活，可以重定向到世界上任何一个URL，比如http://www.sina.com.cn

1. 请求转发

getServletContext().getRequestDispatcher("/MainPage").forward(request, response);

请求转发是在服务器内部的切换，效率比重定向高。

请求转发的限制：只能转发到本应用的其他资源。

**11月21日**

Servlet之间共享数据，举例：把LoginServlet中的用户名传递到LoginSuccess中。

1、如果页面之间是转发（forward）的关系，可以用request对象来承载数据的传递，具体做法：

在LoginServlet.java中：request.setArrribute(“USER\_NAME”, name);

在LoginSuccess.java中：String name = request.getArrribute(“USER\_NAME”);

页面之间如果是重定向关系，则：

在LoginServlet.java中：

2、response.sendRedirect("/UserManager56/LoginSucess?USER\_NAME="+name);

在LoginSuccess.java中：request.getParameter(“USER\_NAME”), 也就是从浏览器传过来的参数。

数据库系统通常也是采用SOCKET TCP 通信，它由一个客户端(mysql workbench/navigator for mysql/ cmd）和一个服务器（mysql）组成。

1. 客户端接收用户的查询语句，通过TCP连接发送到服务器端；
2. 服务器端解析这些查询语句，并且对数据库做出相应的操作（增删改查），再把结果通过TCP连接返回给客户端；
3. 客户端把接受的服务器端返回来的结果呈现给用户。

Java 中操作数据库的模块 JDBC

如何在JavaWeb应用中使用JDBC

1. 建表
2. 把驱动复制到工程
3. 书写JDBC相关代码

## 11月25日

页面之间的跳转和数据传递：

1、重定向浏览器发了 GET 请求：

在LoginServlet.java中：

response.sendRedirect("/UserManager56/LoginSucess?USER\_NAME="+name);

在LoginSuccess.java中：request.getParameter(“USER\_NAME”), 也就是从浏览器传过来的参数。

2、转发浏览器不参与，仅仅是服务器不同Servlet之间的方法调用跳转

可以用request对象来承载数据的传递，具体做法：

在LoginServlet.java中：request.setArrribute(“USER\_NAME”, name);

在LoginSuccess.java中：String name = request.getArrribute(“USER\_NAME”);

3、超链接浏览器发了 GET 请求

"/UserManager56/LoginSucess?USER\_NAME="+name);

在目标Servlet中：request.getParameter(“USER\_NAME”), 也就是从浏览器传过来的参数。

4、FORM表单 action 发了GET/POST 请求

<form action="/UserManager56/LoginServlet" method="get">

用户名<input type="text" name="user\_name"><br/>

密码<input type="password" name="pass\_word"><br/>

<input type="submit" name value="Login"><br/>

</form>

在目标Servlet中：request.getParameter(“user\_name”), 也就是从浏览器传过来的参数。

MVC设计模式\框架

Model数据对数据库操作等 UserInfo JdbcUtils

View用户界面界面显示，用户交互 LoginFailure LoginSucess UpdateOk UpdateFailure UserList

Controller控制器控制程序执行流程、跳转 LoginServlet、UpdateUSer

## 11月28日

JavaBean规范

## 11月30日

Servlet 生命周期：

一个Servlet第一次被用到的时候，会被初始化并创建一个对象实例：它的构造方法、init()方法, doGet/doPost会被调用到。Servlet对象实例会在适当的时候(比如：服务器内存不够或者Web应用关闭时)被销毁：它的destroy()方法会被调用到。

Servlet单实例多线程模型：

对于若干次请求，一个Servlet只生成一个对象实例（服务器为了节省内存空间），并且一个请求对应一个子线程（因为服务器需要异步处理大量请求）。

ServletContext对象的生命周期就是整个WEB应用的生命周期，它代表整个WEB 应用。它是一个单例对象。

ServletContextListener的使用：监视ServletContext的生命周期。

## 12月2日

Cookie存在于浏览器客户端（内存中或者磁盘上）

一）在应答包中回写cookie数据：response.addCookie(cookie);

对应的应答包格式如下：

HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Set-Cookie: LAST\_TIME="2016-12-02T14:38:02.599+0800"; Version=1

Content-Type: text/html;charset=utf-8

Date: Fri, 02 Dec 2016 06:38:02 GMT

二）如果浏览器客户端有Cookie数据，则浏览器发起请求时，会携带Cookie数据：

对应的请求包格式如下：

POST /UserManager78/LoginServlet HTTP/1.1

Host: 127.0.0.1:8080

Cookie: LAST\_TIME="2016-12-02T14:37:33.132+0800"

Servlet通过请求包获取Cookie数据：request.getCookies();

12月5日

Html 静态网页 简单

Servlet 动态网页 麻烦

JSP（Java Server Page） 直接编辑网页，但可以开发动态网页

ASP（Active Server Page）

JSP 兼顾了HTML的简单和Servlet的动态特性

JSP = html+css+js+java+jsp标签

jsp声明

<%! %>在这里面声明的变量是成员变量

<% %>在这里面声明的变量是方法内的局部变量

***具体语法***

1. 指令元素（编译指令）
2. page指令

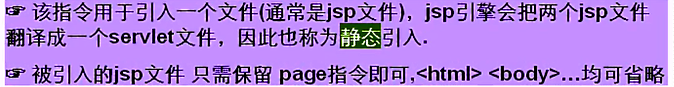
pl:<%@pagelanguage=*"java"*import=*"java.util.\*"*pageEncoding=*"utf-8"*contentType=*"text/html; charset=utf-8"*%>

常用的属相有 import，contentType，errorPage

1. include 指令

<%@include file=*"路径"*%>

*Ps:路径是相对路径。由于该方法引来的页面会和当前页编写成一个serlet文件，所以被引页和当前页的页面编码得相同。*



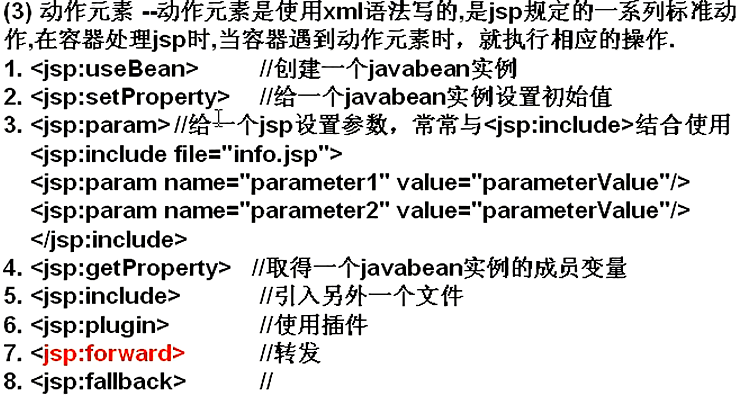
1. 脚本元素

<%=java表达式 %>

<% java代码//int i = 10 局部变量%>

<%！ int j = 20 // 全局变量%>

1. 动作元素（运行指令）



*ps:<jsp:include****page****=”” /> 上面的错了。*

