Java从很多方面来说，就是简化版的C++。

——Michael Feldman

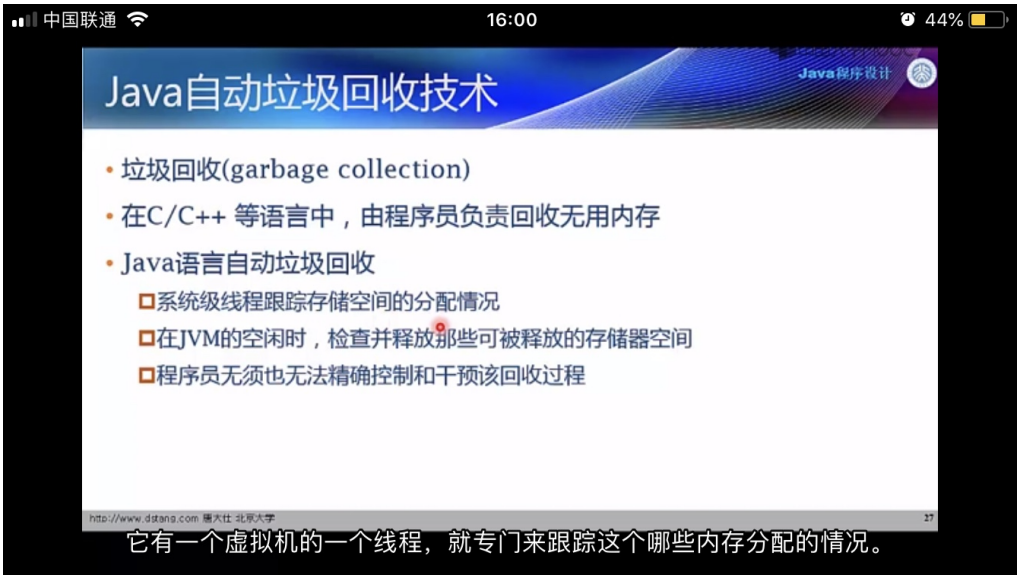
Java与C++最大的区别——只能单根继承

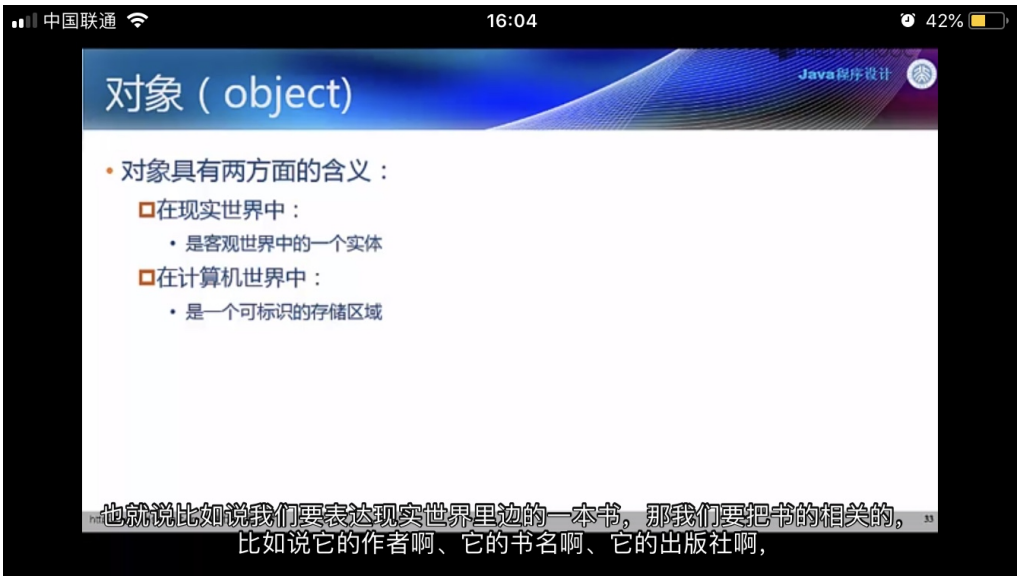
Java是C++--

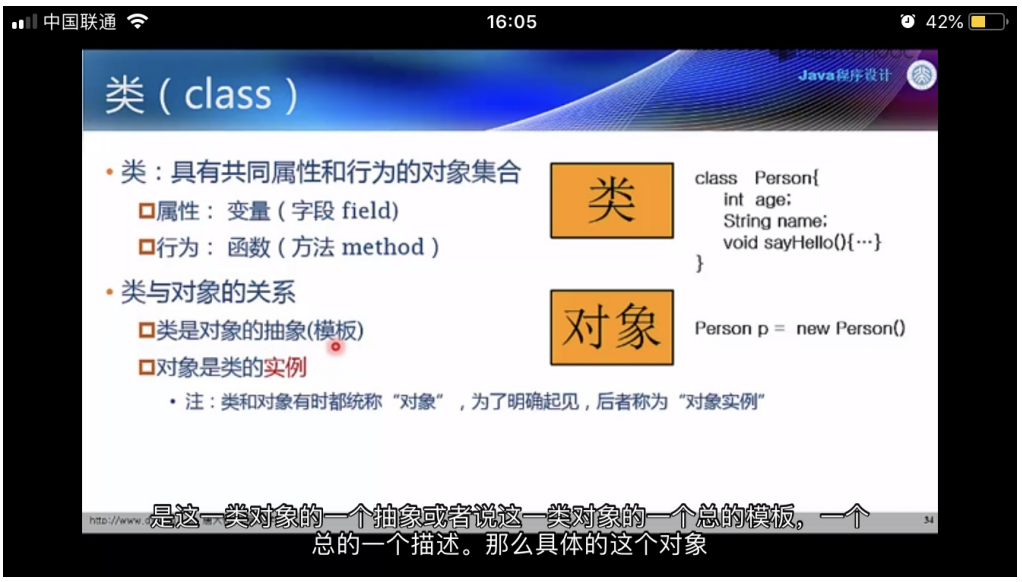
☛ 注意什么是对象

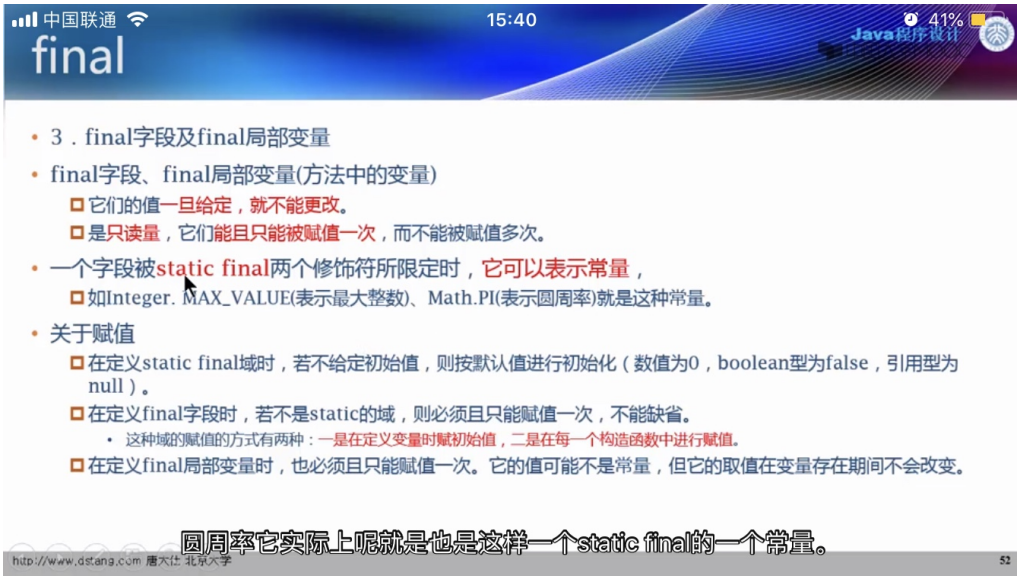
```
1    MyClass a = new MyClass();
```

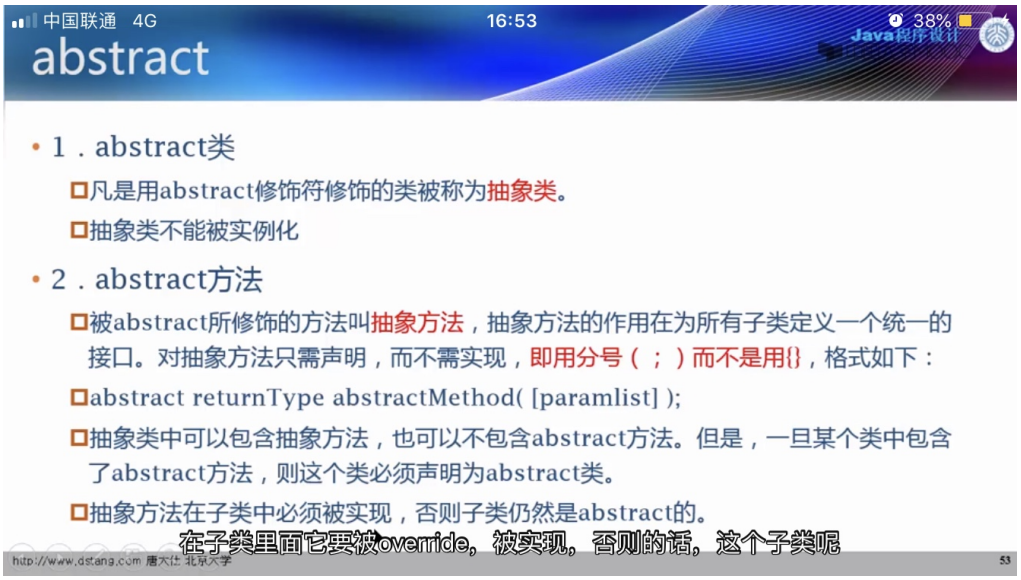
此时 a 是指向对象的指针，而不能说 a 是对象。指针存储在栈中，对象存储在堆中，操作实例实际上是通过指针间接操作对象。多个指针可以指向同一个对象。




中国联通
16:04
42%
对象（object）
Java程序设计
• 对象具有两方面的含义：
□在现实世界中：
• 是客观世界中的一个实体
□在计算机世界中：
• 是一个可标识的存储区域
也就说比如说我们要表达现实世界里边的一本书，那我们要把书的相关的，
比如说它的作者啊、它的书名啊、它的出版社啊，


中国联通
16:05
42%
类（class）
Java程序设计
• 类：具有共同属性和行为的对象集合
□属性： 变量（字段 field）
□行为： 函数（方法 method）
• 类与对象的关系
□类是对象的抽象(模板)
□对象是类的实例
• 注：类和对象有时都统称“对象”，为了明确起见，后者称为“对象实例”
类
class Person{
int age;
String name;
void sayHello(){…}
}
对象
Person p = new Person()
是这一类对象的一个抽象或者说这一类对象的一个总的模板，一个
总的一个描述。那么具体的这个对象

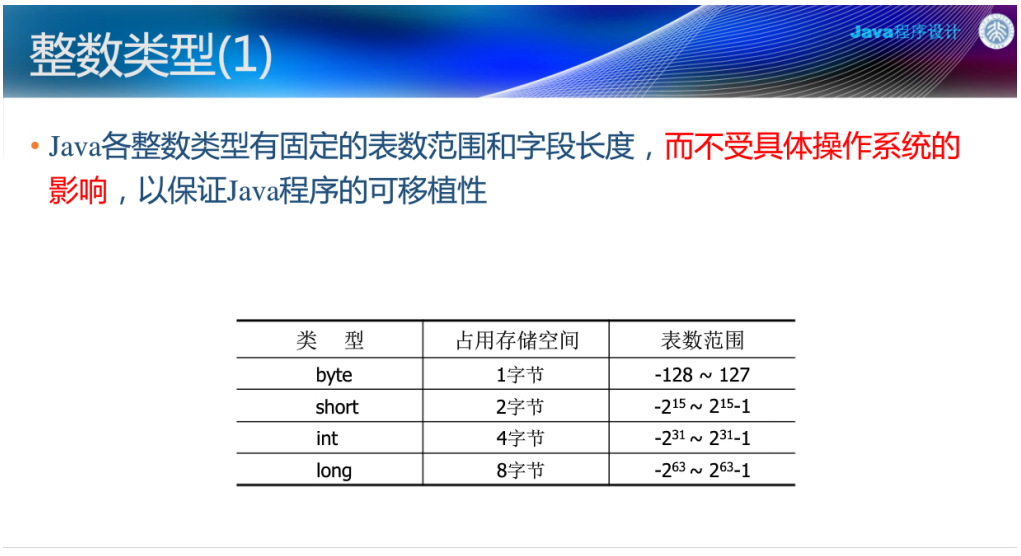

中国联通
15:40
41%
final
Java程序设计
• 3．final字段及final局部变量
• final字段、final局部变量(方法中的变量)
□它们的值一旦给定，就不能更改。
□是只读量，它们能且只能被赋值一次，而不能被赋值多次。
• 一个字段被static final两个修饰符所限定时，它可以表示常量，
□如Integer. MAX_VALUE(表示最大整数)、Math.PI(表示圆周率)就是这种常量。
• 关于赋值
□在定义static final域时，若不给定初始值，则按默认值进行初始化（数值为0，boolean型为false，引用型为null）。
□在定义final字段时，若不是static的域，则必须且只能赋值一次，不能缺省。
• 这种域的赋值的方式有两种：一是在定义变量时赋初始值，二是在每一个构造函数中进行赋值。
□在定义final局部变量时，也必须且只能赋值一次。它的值可能不是常量，但它的取值在变量存在期间不会改变。
圆周率它实际上呢就是也是这样一个static final的一个常量。
http://www.dstang.com 唐大仕 北京大学
52



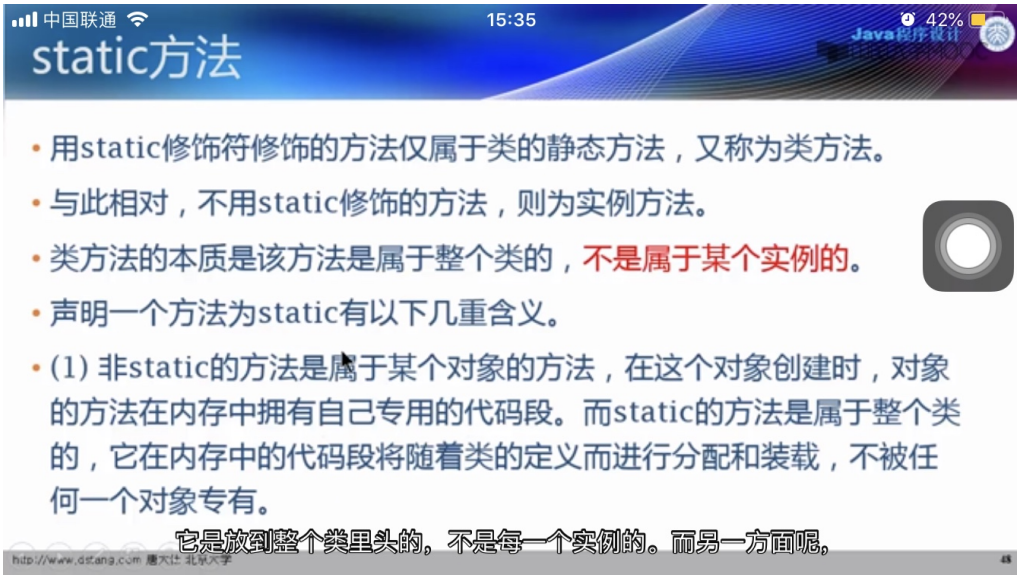
在CMD窗口中输入 CD\（就是返回根目录） 回车 在输入 D: 即可在D盘操作状态

cd ..返回上一级目录

boolean类型数据只允许取值true或false 默认false



类　型	占用存储空间	表数范围
byte	1字节	-128 ~ 127
short	2字节	$-2^{15} \sim 2^{15}-1$
int	4字节	$-2^{31} \sim 2^{31}-1$
long	8字节	$-2^{63} \sim 2^{63}-1$



可以看到，nextLine()自动读取了被next()去掉的Enter作为他的结束符，所以没办法给s2从键盘输入值。经过验证，我发现其他的next的方法，如double nextDouble()， float nextFloat()， int nextInt()等与nextLine()连用时都存在这个问题，解决的办法是：在每一个 next()、nextDouble() 、 nextFloat()、nextInt() 等语句之后**加一个nextLine()语句，将被next()去掉的Enter结束符过滤掉。**

```
public class NextTest{
    public static void main(String[] args) {
        String s1,s2;
        Scanner sc=new Scanner(System.in);
        System.out.print("请输入第一个字符串：");
        s1=sc.next();
        sc.nextLine();
        System.out.print("请输入第二个字符串：");
        s2=sc.nextLine();
        System.out.println("输入的字符串是："+s1+" "+s2);
    }
}
```

next() 与 nextLine() 区别

next()：

- 1、一定要读取到有效字符后才可以结束输入。
- 2、对输入有效字符之前遇到的空白，next() 方法会自动将其去掉。
- 3、只有输入有效字符后才将其后面输入的空白作为分隔符或者结束符。
- next() 不能得到带有空格的字符串。

nextLine()：

- 1、以Enter为结束符,也就是说 nextLine()方法返回的是输入回车之前的所有字符。
- 2、可以获得空白。

Argument 专用于 Actual Argument（实际参数，实参），Parameter 专用于 Formal Parameter（形式参数，形参）。

若在声明数组时进行赋值即初始化称为静态内存分配。

▶ Java 程序运行在 JVM 上，JVM 是程序与操作系统之间的桥梁。
▶ JVM 实现了 Java 的平台无关性。
▶ JVM 是内存分配的前提。

所有的局部变量都是放在栈内存中保存的，不管其实基本类型的变量还是引用类型的变量，都是存储在各自的方法栈区中；但引用类型变量所引用的对象（包括数据，普通java对象）则总是存储在堆内存中。

String 对象一旦被创建就是固定不变的，对 String 对象的任何操作都不影响到原对象，而是会生成新的对象。

static 标记的属性或方法由整个类（所有实例）共享

静态变量各种类型的缺省值：
boolean: false
byte: 0
char: ''（两个单引号之间无内容）
short: 0
int: 0
float:0.0
long: 0
double:0.0
Reference: null （对象引用）

java数组是静态的，必须经过初始化后才能使用，并且一旦初始化指定了数组的长度，该长度是不可变的。
两种初始化的方式：
1.静态初始化：初始化时由程序猿显示指定每个数组元素的初始值，由系统决定数组的长度
举个栗子：
String[] names = new String[]{"孙猴子","唐僧","猪八戒"};
2.动态初始化：就是与静态初始化相对的撒，23333。其实动态初始化就是在初始化的时候指定数组长度（这时已经分配内存）

举个栗子：

```
String[] names = new String[3];
names[0]="孙猴子";
names[1]="唐僧";
names[2]="猪八戒";
```

但是初始化只有这两种方式吗？我们需要理解到真正的初始化到底做了什么工作。

先了解一下初始化后它在内存中是这个样子的。

在《Java编程思想》第86页有这样一段话：

“static方法就是没有this的方法。在static方法内部不能调用非静态方法，反过来是可以的。而且可以在没有创建任何对象的前提下，仅仅通过类本身来调用static方法。这实际上正是static方法的主要用途。”

这段话虽然只是说明了static方法的特殊之处，但是可以看出static关键字的基本作用，简而言之，一句话来描述就是：方便在没有创建对象的情况下来进行调用（方法/变量）。很显然，被static关键字修饰的方法或者变量不需要依赖于对象来进行访问，只要类被加载了，就可以通过类名去进行访问。

static可用来表示全局变量

创建出来的对象只包含属于各自的成员变量，并不包括成员方法。因为同一个类的对象拥有各自的成员变量，存储在各自的堆内存中，但是他们共享该类的方法，并不是每创建一个对象就把成员方法复制一次。

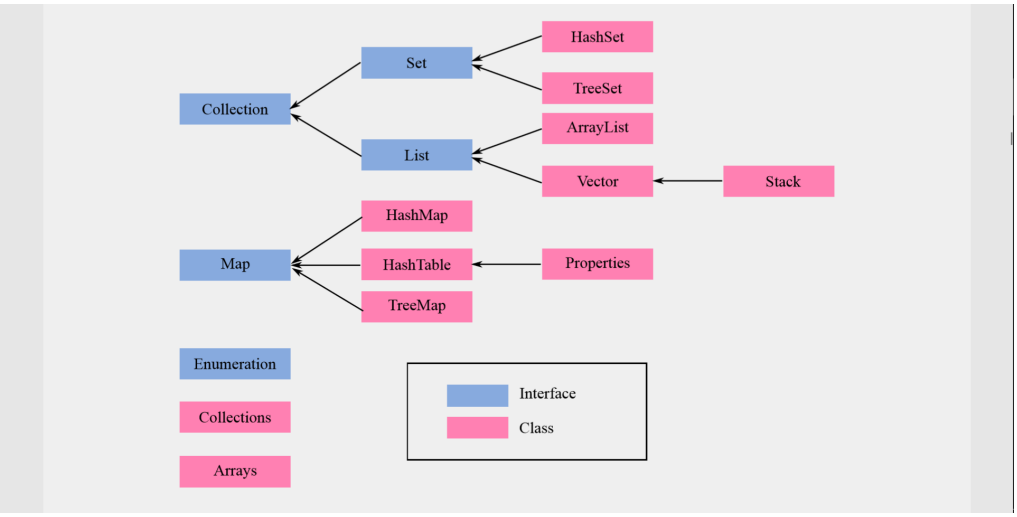
抽象机制是开发人员在开发过程中使用的机制，而动态绑定机制是Java虚拟机运行时提供的机制。

abstract可以修饰类和成员方法

类继承抽象类类实现多接口

抽象类是对类的抽象（可以抽象但不宜实例化），而接口是对行为的抽象。

泛型的实质是将原本确定不变的数据类型参数化。



以上容器内只存放对象的引用 而不是基本数据类型

利用 `<? extends Fruit>` 形式的通配符，可以实现泛型的向上转型

Java 集合可分为 Collection 和 Map 两种体系

Collection常用接口：

Set：元素无序、不可重复的集合

List：元素有序，可重复的集合

Collection接口

```
|----Collection
    |----List
        |----ArrayList(主要实现类)：底层是用数组实现的，线程不安全的，比Vector 效率高。增删慢，查找快。
        |----Vector：底层是用数组实现的，线程安全的，效率低
        |----LinkedList:底层是用链表实现的，增删快，查找慢
    |----Set : 存储的元素是无序的且不可重复的
        |----HashSet(主要实现类)：底层是创建了hashMap对象
        |----LinkedHashSet：继承了HashSet，底层实现原理和HashSet一样。
                但是LinkedHashSet可以按照元素添加的顺序进行遍历。因为LinkedHashSet
                底层维护了一对指针（链表）用来记录元素添加的顺序。
        |----TreeSet：可以对对象中的属性进行排序
```

遍历Collection的两种方式： Iterator和foreach循环

增强for循环：只能用于数组和集合

Iterator 接口：用来遍历集合中的元素

增强for循环：(foreach循环)

格式：

for(元素的类型 临时变量 : 数组、集合的对象名){

}

Java中的Iterator功能比较简单，并且只能单向移动：

(1) 使用方法iterator()要求容器返回一个Iterator。第一次调用Iterator的next()方法时，它返回序列的第一个元素。注意：iterator()方法是java.lang.Iterable接口,被Collection继承。

(2) 使用next()获得序列中的下一个元素。

(3) 使用hasNext()检查序列中是否还有元素。

(4) 使用remove()将迭代器新返回的元素删除。

Iterator是Java迭代器最简单的实现，为List设计的ListIterator具有更多的功能，它可以从两个方向遍历List，也可以从List中插入和删除元素。

迭代器应用：

```
list l = new ArrayList();
l.add("aa");
l.add("bb");
l.add("cc");
for (Iterator iter = l.iterator(); iter.hasNext();) {
  String str = (String)iter.next();
  System.out.println(str);
}
/*迭代器用于while循环
Iterator iter = l.iterator();
while(iter.hasNext()){
  String str = (String) iter.next();
  System.out.println(str);
}
*/
```

如何向HashSet中添加数据？或HashSet的底层实现原理？

当我们向HashSet中存放数据a时，会先根据该对象中的hashCode方法返回的值决定存放在数组中的位置。如果存放的位置没有其它元素那么直接存放。如果存放的位置已经有了其它元素b时，会调用a的equals方法进行内容的比较。如果返回的是true那么认为两个元素是相同的则不能再次存放。如果返回的是false那么认为两个元素不同。以链表的形式进行存放。

存储元素所在类的要求：要求必须重写hashCode和equals方法

C：无语言层面的异常机制

C++：注意内存回收

Java：有垃圾回收

面向方面（AOP）：Java EE过滤器
面向事件：Java EE监听器
建立概念：Servlet和JSP是同一个东西 只是写法不同

JSP(Java Servlet Page)在服务器端运行

浏览器端数据均是字符串

URL映射的两种方式：web.xml配置;
基于@WebServlet()注解

不管用什么语言实现，从客户端到浏览器都要遵循HTTP协议：即通过请求-响应模式，
一、参数的方式放到URL里;
二、放在请求体里;
三、放在请求头里。

请求可以forward（转发），一直向后面的servlet传递。

会话间隔用于判断两个请求-响应是否属于同一个会话。

不变的部分封装起来 变的部分拿出来放到随改随用的配置文件中，无需重新build！！！

用户再次请求时，系统判断若.jsp修改时间新于.class的修改时间，则会compile .java 使其变成.class

forward仅在浏览器与服务器之间产生一次请求，而redirect会产生两次请求

JSP脚本里嵌入的代码
<%
int a = 0;
%>
在转换成servlet时，被嵌入到doGet、doPost方法

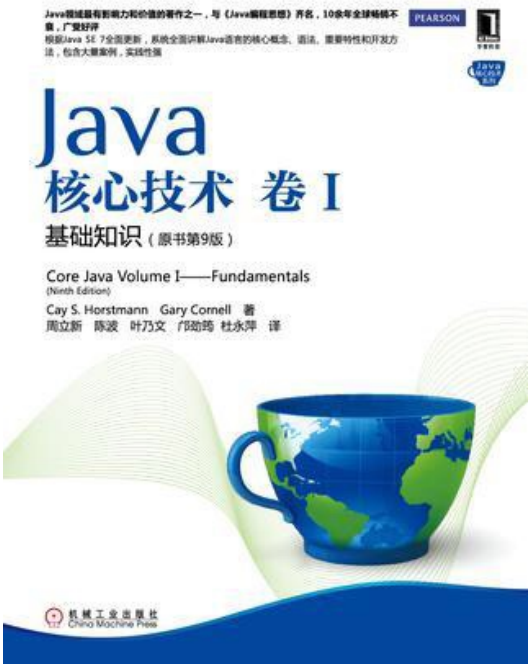
servlet中的response.getWriter()对应JSP中的out对象

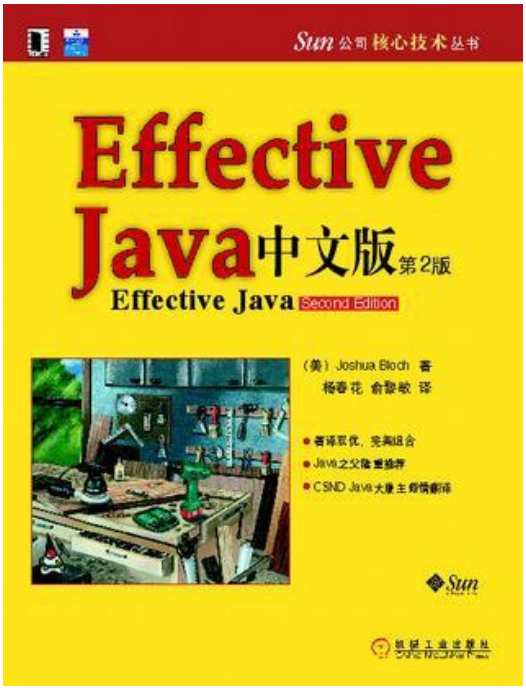
servlet中的ServletContext与JSP中的application对应（全生命周期）

MVC分别对应数据、页面显示和业务逻辑

——

How2J网站 w3school





《Effective Java》、《Java Performance》、《Design Pattern》、《Head First Statistics》

《Java虚拟机》

Bruce, Thinking in Java