[**第99集 13**](#_Toc438805889)

[**容器当中接口的层次图如下： 14**](#_Toc438805890)

[**容器当中接口的特点如下： 14**](#_Toc438805891)

[**.Collection接口 14**](#_Toc438805892)

[**.List接口： 15**](#_Toc438805893)

[**.SET接口 15**](#_Toc438805894)

[**第100集 16**](#_Toc438805895)

[**容器之List类的特征及常见子类的特点： 16**](#_Toc438805896)

[**第101与102集 19**](#_Toc438805897)

[**通过对List接口的实现类ArrayList的常见方法的应用 19**](#_Toc438805898)

[**第103-104集 23**](#_Toc438805899)

[**手动封装SxtLinkedList类代码如下： 24**](#_Toc438805900)

[**第105集 28**](#_Toc438805901)

[**Map的基本用法 28**](#_Toc438805902)

[**第106集 30**](#_Toc438805903)

[**自定义实现Map 30**](#_Toc438805904)

[**第109集 33**](#_Toc438805905)

[**List\_Map 33**](#_Toc438805906)

[**第110集 33**](#_Toc438805907)

[**Set接口的特点以及常用方法 33**](#_Toc438805908)

[**第111集 34**](#_Toc438805909)

[**自定义个SxtHashSet类的代码为： 34**](#_Toc438805910)

[**第112集 36**](#_Toc438805911)

[**容器之数据存储综合练习（JavaBean）： 36**](#_Toc438805912)

[**.Employee类 36**](#_Toc438805913)

[**.Test01测试类代码如下： 38**](#_Toc438805914)

[**第113集 40**](#_Toc438805915)

[**数据封装之用容器封装数据测试代码如下： 40**](#_Toc438805916)

[**第114集 42**](#_Toc438805917)

[**Iterator接口： 42**](#_Toc438805918)

[**第115集 44**](#_Toc438805919)

[**Iterator迭代器如何遍历Map: 44**](#_Toc438805920)

[**第117集 46**](#_Toc438805921)

[**泛型的继承 46**](#_Toc438805922)

[**泛型的擦除 49**](#_Toc438805923)

[**总结： 52**](#_Toc438805924)

[**第118集 52**](#_Toc438805925)

[**自定义泛型的深入2 52**](#_Toc438805926)

[**泛型的上限 54**](#_Toc438805927)

[**super,泛型的下限 57**](#_Toc438805928)

[**泛型的嵌套。 59**](#_Toc438805929)

[**第119集 63**](#_Toc438805930)

[**迭代器与分拣 63**](#_Toc438805931)

[**.容器深入 65**](#_Toc438805932)

[**.自定义泛型的应用。 71**](#_Toc438805933)

[**第120集 74**](#_Toc438805934)

[**HashMap深入原理。 74**](#_Toc438805935)

[**第121集 80**](#_Toc438805936)

[**HashMap经典存储，分拣思路与面向对象的思路组合结题。 80**](#_Toc438805937)

[**.测试类代码如下： 83**](#_Toc438805938)

[**第122与123集 86**](#_Toc438805939)

[**冒泡排序 86**](#_Toc438805940)

[**.冒泡排序晋级版 91**](#_Toc438805941)

[**.冒泡排序最终版本 92**](#_Toc438805942)

[**第124集 94**](#_Toc438805943)

[**引用类型的排序： 94**](#_Toc438805944)

[**泛型的应用。 100**](#_Toc438805945)

[**第125集 104**](#_Toc438805946)

[**排序的进阶 104**](#_Toc438805947)

[**第126集 111**](#_Toc438805948)

[**引用类型的排序，新闻排序 111**](#_Toc438805949)

[**排序的策略： 111**](#_Toc438805950)

[**1、实体类 java.lang.Comparable+compareTo 111**](#_Toc438805951)

[**2、业务排序java.util.Comparator+compare 111**](#_Toc438805952)

[**测试 114**](#_Toc438805953)

[**商品的排序： 115**](#_Toc438805954)

[**实现Comparable接口，提高比较的条件。 117**](#_Toc438805955)

[**测试 118**](#_Toc438805956)

[**根据收藏量进行排序的标准 118**](#_Toc438805957)

[**第127集 119**](#_Toc438805958)

[**引用类型\_排序容器\_TreeSet与TreeMap 119**](#_Toc438805959)

[**测试代码： 121**](#_Toc438805960)

[**案例2,Worker代码如下： 123**](#_Toc438805961)

[**测试类代码如下： 125**](#_Toc438805962)

[**TreeMap:要求键可以排序，与上TreeSet同理 126**](#_Toc438805963)

[**第128集 128**](#_Toc438805964)

[**Collections工具类常见方法的使用 128**](#_Toc438805965)

[**第129集 131**](#_Toc438805966)

[**容器的收尾工做： 131**](#_Toc438805967)

[**使用队列实现自己定义堆栈：（弹，压，获取头） 133**](#_Toc438805968)

[**第130集 135**](#_Toc438805969)

[**Enumeration接口： 135**](#_Toc438805970)

[**第131集 138**](#_Toc438805971)

[**Hashtable与HashMap区别： 138**](#_Toc438805972)

[**Properties代码的如下： 138**](#_Toc438805973)

[**.作用：读取资源配置文件 138**](#_Toc438805974)

[**.键与值只能为字符串 138**](#_Toc438805975)

[**.方法：setProperty(String key,String value); 139**](#_Toc438805976)

[**.输出到文件 140**](#_Toc438805977)

[**第132集 144**](#_Toc438805978)

[**引用的分类：（强，软，弱，虚）： 144**](#_Toc438805979)

[**weakHashMap 146**](#_Toc438805980)

[**IdentityHashMapDemo代码如下： 147**](#_Toc438805981)

[**EnumMap. 148**](#_Toc438805982)

[**第133集 149**](#_Toc438805983)

[**同步控制： 150**](#_Toc438805984)

[**只读设置： 151**](#_Toc438805985)

[**第134集 153**](#_Toc438805986)

[**工作当中的常见组件： 153**](#_Toc438805987)

[**函数式编程 154**](#_Toc438805988)

[**约束条件的加入 158**](#_Toc438805989)

[**集合的引入：交集，差集，并集 160**](#_Toc438805990)

[**第135集 162**](#_Toc438805991)

[**Guava的进阶 162**](#_Toc438805992)

[**实例2 164**](#_Toc438805993)

[**Guava对HashMap的进一步实现 166**](#_Toc438805994)

[**第138集 170**](#_Toc438805995)

[**Apache Commons Collections 170**](#_Toc438805996)

[**函数式编程 170**](#_Toc438805997)

[**函数式编程之2 174**](#_Toc438805998)

[**第139集 180**](#_Toc438805999)

[**函数式编程之闭包 180**](#_Toc438806000)

[**第140集 187**](#_Toc438806001)

[**Commons对集合进行的操作： 188**](#_Toc438806002)

[**交集，并集，差集 188**](#_Toc438806003)

[**Commons对队列的扩展： 190**](#_Toc438806004)

[**第141集 192**](#_Toc438806005)

[**Commons对迭代器的扩展： 193**](#_Toc438806006)

[**Commons双向的Map 197**](#_Toc438806007)

[**Bag 包 允许重复 199**](#_Toc438806008)

[**第142集 202**](#_Toc438806009)

[**对集合这一章的总结： 202**](#_Toc438806010)

[**第143集 204**](#_Toc438806011)

[**IO入门之File类代码如下： 204**](#_Toc438806012)

[**File类进阶： 205**](#_Toc438806013)

[**第144集 207**](#_Toc438806014)

[**File类的常见方法代码如下： 207**](#_Toc438806015)

[**第145集 211**](#_Toc438806016)

[**对目录的操作： 212**](#_Toc438806017)

[**文件夹子目录的遍历： 214**](#_Toc438806018)

[**第146集 216**](#_Toc438806019)

[**IO流的基本概念和原理： 216**](#_Toc438806020)

[**第147集 217**](#_Toc438806021)

[**读取文件： 217**](#_Toc438806022)

[**写出文件： 220**](#_Toc438806023)

[**第148集 222**](#_Toc438806024)

[**文件的拷贝： 222**](#_Toc438806025)

[**工具类的封装： 225**](#_Toc438806026)

[**第149集 229**](#_Toc438806027)

[**文件的拷贝： 230**](#_Toc438806028)

[**第150集 233**](#_Toc438806029)

[**参考150集视频，对以前的代码的健壮性进行了扩展即可了： 233**](#_Toc438806030)

[**第151集 233**](#_Toc438806031)

[**字符流,读取纯文本 233**](#_Toc438806032)

[**文件的写出： 235**](#_Toc438806033)

[**第152集 237**](#_Toc438806034)

[**IO之处理流： 237**](#_Toc438806035)

[**处理流字符流： 240**](#_Toc438806036)

[**第153与154集 242**](#_Toc438806037)

[**处理流之转换流： 242**](#_Toc438806038)

[**一、编码《-----》解码 242**](#_Toc438806039)

[**二、转换流: 字节流 转为字符流   处理乱码(编码集、解码集) 243**](#_Toc438806040)

[**编码-》解码 244**](#_Toc438806041)

[**第155集 246**](#_Toc438806042)

[**IO图形化小结： 247**](#_Toc438806043)

[**第156-157集 248**](#_Toc438806044)

[**其他流的补充： 248**](#_Toc438806045)

[**字节流： 248**](#_Toc438806046)

[**案例2; 251**](#_Toc438806047)

[**第158集 255**](#_Toc438806048)

[**其他流之处理流： 255**](#_Toc438806049)

[**DataInputStream: 255**](#_Toc438806050)

[**DataOutputStream: 255**](#_Toc438806051)

[**案例2： 258**](#_Toc438806052)

[**第159集 262**](#_Toc438806053)

[**处理流 262**](#_Toc438806054)

[**案例： 264**](#_Toc438806055)

[**第160集 266**](#_Toc438806056)

[**编写工具类实现关闭的方法： 267**](#_Toc438806057)

[**测试; 268**](#_Toc438806058)

[**第161集 272**](#_Toc438806059)

[**打印流： 272**](#_Toc438806060)

[**buffer: 276**](#_Toc438806061)

[**第162集 277**](#_Toc438806062)

[**IO流之设计模式（装饰模式）： 277**](#_Toc438806063)

[**·类与类之间的关系： 278**](#_Toc438806064)

[**·测试： 279**](#_Toc438806065)

[**第163集 280**](#_Toc438806066)

[**ruboCopy的应用： 280**](#_Toc438806067)

[**1.环境的配置跟java的环境变量配置一样： 280**](#_Toc438806068)

[**2.不要把父目录拷贝到子目录当中: 281**](#_Toc438806069)

[**3.删除超长目录; 281**](#_Toc438806070)

[**第164集 284**](#_Toc438806071)

[**Io之，RandomAccessFilel类： 284**](#_Toc438806072)

[**第165->167集 286**](#_Toc438806073)

[**文件的分割与合并： 286**](#_Toc438806074)

[**第168集 294**](#_Toc438806075)

[**.IO总结： 294**](#_Toc438806076)

[**第169集 297**](#_Toc438806077)

[**Thread线程的入门： 297**](#_Toc438806078)

[**第170集 297**](#_Toc438806079)

[**java事件如何实现多线程： 298**](#_Toc438806080)

[**第171集 300**](#_Toc438806081)

[**静态代理模式： 300**](#_Toc438806082)

[**第172集 302**](#_Toc438806083)

[**使用Runnable创建线程： 303**](#_Toc438806084)

[**测试： 304**](#_Toc438806085)

[**web12306代码： 305**](#_Toc438806086)

[**第173集 306**](#_Toc438806087)

[**多线程的第三种实现方式（Callable）: 306**](#_Toc438806088)

[**第174集 310**](#_Toc438806089)

[**线程的状态： 310**](#_Toc438806090)

[**如何干涉线程的状态以及停止方法： 310**](#_Toc438806091)

[**第175集 312**](#_Toc438806092)

[**线程的合并，阻塞： 312**](#_Toc438806093)

[**Yield方法（）： 313**](#_Toc438806094)

[**第176集 314**](#_Toc438806095)

[**线程的Sleep（）； 314**](#_Toc438806096)

[**案例2 316**](#_Toc438806097)

[**第177集 318**](#_Toc438806098)

[**线程的基本信息： 318**](#_Toc438806099)

[**测试： 319**](#_Toc438806100)

[**线程的优级别： 320**](#_Toc438806101)

[**第178集 322**](#_Toc438806102)

[**线程的同步： 322**](#_Toc438806103)

[**第179集 327**](#_Toc438806104)

[**单例的设计模式： 327**](#_Toc438806105)

[**线程的同步： 327**](#_Toc438806106)

[**单例模式的两种情况： 331**](#_Toc438806107)

[**第180集 333**](#_Toc438806108)

[**一.线程死锁问题： 333**](#_Toc438806109)

[**二、同步方法synchronized 333**](#_Toc438806110)

[**三、死锁: 过多的同步容易造成死锁 333**](#_Toc438806111)

[**第181集 336**](#_Toc438806112)

[**生产者与消费者模式： 336**](#_Toc438806113)

[**生产者： 339**](#_Toc438806114)

[**消费者： 340**](#_Toc438806115)

[**测试; 341**](#_Toc438806116)

[**馒头案例的代码如下： 341**](#_Toc438806117)

[**第182集 344**](#_Toc438806118)

[**任务调度： 344**](#_Toc438806119)

[**Time定时器类 344**](#_Toc438806120)

[**TimeTask定时任务类： 344**](#_Toc438806121)

[**第183集 346**](#_Toc438806122)

[**对线程的总结： 346**](#_Toc438806123)

[**第184集 348**](#_Toc438806124)

[**网络编程： 348**](#_Toc438806125)

[**第185集 349**](#_Toc438806126)

[**网络编程值地址及端口： 349**](#_Toc438806127)

[**第186集 351**](#_Toc438806128)

[**URL： 352**](#_Toc438806129)

[**URI(Uniform resource identifier)统一资源标识符,用来唯一的标识一个资源 352**](#_Toc438806130)

[**URL（Uniform Resource Locator）统一资源定位器,它是一种具体的URI 352**](#_Toc438806131)

[**URL： 352**](#_Toc438806132)

[**一、创建 352**](#_Toc438806133)

[**二、方法 352**](#_Toc438806134)

[**三、流 353**](#_Toc438806135)

[**网络爬虫代码如下： 354**](#_Toc438806136)

[**第187集 356**](#_Toc438806137)

[**UDP通讯： 356**](#_Toc438806138)

[**Server端： 357**](#_Toc438806139)

[**Client端： 358**](#_Toc438806140)

[**第188集 360**](#_Toc438806141)

[**UDP编程进阶： 360**](#_Toc438806142)

[**Server端： 360**](#_Toc438806143)

[**Client端： 362**](#_Toc438806144)

[**第189集 364**](#_Toc438806145)

[**TCP编程： 365**](#_Toc438806146)

[**Socket编程： 365**](#_Toc438806147)

[**Client: 367**](#_Toc438806148)

[**第190集 368**](#_Toc438806149)

[**Tcp编程之多客户端： 368**](#_Toc438806150)

[**聊天室,客户端多线程原理： 368**](#_Toc438806151)

[**Client端： 370**](#_Toc438806152)

[**第191集 371**](#_Toc438806153)

[**聊天室，多客户端原理： 372**](#_Toc438806154)

[**Server端： 372**](#_Toc438806155)

[**Client端： 373**](#_Toc438806156)

[**发送端： 375**](#_Toc438806157)

[**接受端： 377**](#_Toc438806158)

[**关闭流工具类： 379**](#_Toc438806159)

[**第192集 380**](#_Toc438806160)

[**聊天室，在上一个版本上增加对用户的代码如下：别的的代码在上一级直接拷贝即可： 380**](#_Toc438806161)

[**Server: 380**](#_Toc438806162)

[**Client: 382**](#_Toc438806163)

[**第193集 383**](#_Toc438806164)

[**聊天室，增加私聊功能： 384**](#_Toc438806165)

[**Server: 384**](#_Toc438806166)

[**Client: 388**](#_Toc438806167)

[**send: 389**](#_Toc438806168)

[**Receive: 392**](#_Toc438806169)

[**关闭流的基本工具： 393**](#_Toc438806170)

[**第194集 394**](#_Toc438806171)

[**聊天室，私聊的实现代码如下： 395**](#_Toc438806172)

[**Client: 399**](#_Toc438806173)

[**Send: 401**](#_Toc438806174)

[**Receive: 403**](#_Toc438806175)

[**关闭流工具类： 405**](#_Toc438806176)

[**第195集 406**](#_Toc438806177)

[**手写服务器： 406**](#_Toc438806178)

[**第196集 409**](#_Toc438806179)

[**手写服务器之html知识： 409**](#_Toc438806180)

[**Html超文本标记语言： 409**](#_Toc438806181)

[**清爽班Hmtl: 410**](#_Toc438806182)

[**Server端代码的如下： 411**](#_Toc438806183)

[**第197集 413**](#_Toc438806184)

[**手写服务器之Http协议： 413**](#_Toc438806185)

[**第198集 417**](#_Toc438806186)

[**手写Tomcat服务器： 417**](#_Toc438806187)

[**手写tomcat\_准备\_html协议\_抓包工具 417**](#_Toc438806188)

[**抓包工具 417**](#_Toc438806189)

[**firebug使用 417**](#_Toc438806190)

[**总结 419**](#_Toc438806191)

[**第199集 419**](#_Toc438806192)

[**对Request,Response的进一步封装： 419**](#_Toc438806193)

[**Response: 419**](#_Toc438806194)

[**Server代码如下： 423**](#_Toc438806195)

[**第200集 426**](#_Toc438806196)

[**手写服务器之Request类的封装及Method方法的封装如下： 426**](#_Toc438806197)

[**Request类的代码如下： 426**](#_Toc438806198)

[**Server代码： 431**](#_Toc438806199)

[**第201集 434**](#_Toc438806200)

[**对Request的进一步实现存储参数： 434**](#_Toc438806201)

[**手写tomcat\_封装Request\_方式及url 434**](#_Toc438806202)

[**目标 434**](#_Toc438806203)

[**获取Request协议 435**](#_Toc438806204)

[**获取method 437**](#_Toc438806205)

[**获取url 438**](#_Toc438806206)

[**总结 439**](#_Toc438806207)

[**第202集 439**](#_Toc438806208)

[**服务器的分发器的封装： 439**](#_Toc438806209)

[**手写tomcat\_servlet\_分发器 439**](#_Toc438806210)

[**目标 439**](#_Toc438806211)

[**Dispatcher类为: 440**](#_Toc438806212)

[**封装Servlet类 441**](#_Toc438806213)

[**封装Dispatcher类 442**](#_Toc438806214)

[**总结 444**](#_Toc438806215)

[**servlet类的封装： 446**](#_Toc438806216)

[**server7 447**](#_Toc438806217)

[**第203集 451**](#_Toc438806218)

[**httpServet多请求处理多态： 451**](#_Toc438806219)

[**Server类代码： 451**](#_Toc438806220)

[**Servlet类代码： 454**](#_Toc438806221)

[**ServletContext类代码如下： 455**](#_Toc438806222)

[**Request类： 456**](#_Toc438806223)

[**Response类： 462**](#_Toc438806224)

[**Dispatch类代码如下： 465**](#_Toc438806225)

[**RegisterServlet类代码： 467**](#_Toc438806226)

[**loginServlet代码： 468**](#_Toc438806227)

[**WabApp类代码如下： 469**](#_Toc438806228)

[**第204集 470**](#_Toc438806229)

[**手写服务器之反射机制： 470**](#_Toc438806230)

[**反射的代码如下： 470**](#_Toc438806231)

[**Server类的代码如下： 472**](#_Toc438806232)

[**Servlet代码如下： 474**](#_Toc438806233)

[**ServletContext代码： 475**](#_Toc438806234)

[**Request代码如下： 476**](#_Toc438806235)

[**Response类代码如下： 481**](#_Toc438806236)

[**DisPatcher类代码如下： 485**](#_Toc438806237)

[**WebApp代码如下： 487**](#_Toc438806238)

[**第205集 488**](#_Toc438806239)

[**服务器之xml配置文件的解析： 489**](#_Toc438806240)

[**SAX  基于事件流的解析 489**](#_Toc438806241)

[**Person类代码： 489**](#_Toc438806242)

[**Person.xml 490**](#_Toc438806243)

[**ParseDemo01代码： 491**](#_Toc438806244)

[**PersonHandler代码： 492**](#_Toc438806245)

[**第206集 495**](#_Toc438806246)

[**xml配置文件，sax解析的应用： 495**](#_Toc438806247)

[**Entity类代码 495**](#_Toc438806248)

[**Mapping代码如下： 496**](#_Toc438806249)

[**web.xml 497**](#_Toc438806250)

[**WebApp代码： 498**](#_Toc438806251)

[**WebHandler代码： 500**](#_Toc438806252)

[**第207集 505**](#_Toc438806253)

[**手写Tomcat服务器的总结版总结： 505**](#_Toc438806254)

# 第99集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季099集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# **容器当中接口的层次图如下：**http://www.sxt.cn/editor/attached/image/20151120/20151120215940_256.png

# **容器当中接口的特点如下：**

## .Collection接口

        Collection 表示一组对象，他是集中收集的意思，就是把一组数据收集起来，Collection接口的两个子接口Set,List:

Set中的数据没有顺序，不可以重复。

List中的数据有顺序，可以重复。

Collection接口中定义的方法：

boolean  add(Object element);

boolean  remove(Object element);

boolean  contains(Object element);

int    size();

boolean  isEmpty();

void  clear();

Iterator  iterator();

boolean   containsAll(Collection c);

boolean  addAll(Collection c);

boolean  removeAll(Collection c);

boolean  retainAll(Collection c);    //交集

Object[] toArray();

## .List接口：

    有序的Collection，此接口的用户可以成对列表中的每个元素的插入位置进行精确的定位

    ，用户可以根据元素的整数索引（在列表中的位置）访问元素，并搜索列表当中的元素。

    与set不同，列表通常允许重复的元素，更精确的讲列表通常允许满足e1.equals(e2)条件的元素e1，e2并存，并且如果列表本身允许null元素的话，通常他们允许多个null元素。

    多了一些跟顺序有关的方法：

        void  add(Object element);

        void add(int index,Object element);

Object get(int index);

Object set(int index,Object element);//修改某一位置的元素。

Object remove(int index);

        int index(Object o);//返回某一元素的索引，如果没有该元素返回-1；

ArrayList:  底层用数组实现的List,特点是：查询效率高 ，增删效率低，不安全。

    LinkedList:底层是双向链表实现的List,特点是：查询效率低，增删效率高，线程不安全。

Vector:底层用数组实现的List,特点：线程安全.

    如何选用？

    线程安全用Vector.

    线程不安全，查找较多用ArrayList，增加或删除元素较多用LinkedList.

## .SET接口

HashSet:采用哈希算法实现的Set

    HashSet的底层是用HashMap实现的，因此查询效率较高，由于采用hashCode算法直接确定元素的内存地址，增删效率也挺高的。

public class TestSet {

public static void main(String[] args) {

Set s = new HashSet();

s.add("hello");

s.add(1);

s.add(3.14);

s.add(new Integer(4));

s.add(new Double(1.2));

s.add("hello");

System.out.println(s.size() + "个元素。");

System.out.println(s);

}

}

# **第100集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季100集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 容器之List类的特征及常见子类的特点：

package com.bjsxt.list;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

/\*\*

 \*

 \* @author Administrator Collection接口下的list接口有一下常用实现类

 \*

 \*         ArrayList:底层实现是数组，线程不安全，效率高，查询效率高，修改，插入，删除效率低。

 \*         LinkedList:底层实现是链表，线程不安全，效率高，查询效率低，修改，插入，删除效率高。

 \*         Vector:线程安全，效率低，

 \*

 \*/

public class ListTest {

public static void main(String[] args) {

List list = new ArrayList();

list.add("hello");

list.add(2);// 包装类：自动装箱！

list.add(new Integer(4));

list.add(new Date());

list.add(true);

System.out.println(list.size() + "个元素。");

System.out.println(list.isEmpty());

list.remove(new Date() + "<<<<<<<<<<<");// hashcode 与equals关系。

List list2 = new ArrayList();

list2.add("aa");

list2.add("bb");

list.add(list2);

System.out.println(list.size() + "个元素。");

//跟顺序的操作

String str = (String) list.get(0);

System.out.println(str);

list.set(0,"haha");

String str2 = (String) list.get(0);

System.out.println(str2);

System.out.println(list.size()+"个元素.");

}

}

class Dog {

private String name;

private boolean gender;

public Dog() {

}

public Dog(String name, boolean gender) {

this.name = name;

this.gender = gender;

}

}

# **第101与102集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季101与102集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 通过对List接口的实现类ArrayList的常见方法的应用

我们对集合有了个更深刻的认识，为了加深对ArrayList的了解我们利用101与102集来自己封装一个SxtArrayList类来模拟ArrayList类，代码如下：

package com.bjsxt.list;

public class SxtArrayList {

private Object[] elementData;

private int size;

public int size() {

return size;

}

public boolean isEmpty() {

return size == 0;

}

public SxtArrayList() {

this(10);

}

public SxtArrayList(int initialCapacity) {

if (initialCapacity < 0) {

try {

throw new Exception();

} catch (Exception e) {

e.printStackTrace();

}

}

elementData = new Object[initialCapacity];

}

public void add(Object obj) {

// 数组扩容

if (size == elementData.length) {

Object[] newArray = new Object[size \* 2 + 1];

System.arraycopy(elementData, 0, newArray, 0, elementData.length);

/\*

 \* for (int i = 0; i < elementData.length; i++) {

 \* newArray[i]=elementData[i]; }

 \*/

elementData = newArray;

}

elementData[size++] = obj;

// size++;

}

public void add(int index, Object obj) {

rangeCheck(index);

ensureCapacity();//数组扩容

System.arraycopy(elementData, index, elementData, index + 1, size - index);

elementData[index] = obj;

size++;

}

private void ensureCapacity() {

// 数组扩容

if (size == elementData.length) {

Object[] newArray = new Object[size \* 2 + 1];

System.arraycopy(elementData, 0, newArray, 0, elementData.length);

/\*

 \* for (int i = 0; i < elementData.length; i++) {

 \* newArray[i]=elementData[i]; }

 \*/

elementData = newArray;

}

}

public Object get(int index) {

rangeCheck(index);

return elementData[index];

}

public void remove(int index) {

rangeCheck(index);

// 删除指定位置的对象

// a b c d e

if (index < 0 || index >= size) {

try {

throw new Exception();

} catch (Exception e) {

e.printStackTrace();

}

int numMoved = size - index - 1;

if (numMoved > 0) {

System.arraycopy(elementData, index + 1, elementData, index, numMoved);

}

elementData[--size] = null;

}

}

public void remove(Object obj) {

for (int i = 0; i < size; i++) {

if (get(i).equals(obj)) {// 注意：底层调用的equals方法，不是==；

remove(i);

}

}

}

public Object set(int index, Object obj) {

rangeCheck(index);

Object oldValue = elementData[index];

elementData[index] = obj;

return oldValue;

}

private void rangeCheck(int index) {

if (index < 0 || index >= size) {

try {

throw new Exception();

} catch (Exception e) {

e.printStackTrace();

}

}

}

public static void main(String[] args) {

SxtArrayList slist = new SxtArrayList(3);

slist.add(132);

slist.add("444");

slist.add(5);

slist.add("333");

slist.add("333");

slist.add("abc");

slist.add("ccc");

System.out.println(slist.size);

System.out.println(slist.get(6));

}

}

# 第103-104集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季103与104集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 手动封装SxtLinkedList类代码如下：

package cn.bjsxt.collection;

import java.util.LinkedList;

public class SxtLinkedList /\*implements List\*/ {

private Node first;

private Node last;

private int size;

public void add(Object obj){

Node n = new Node();

if(first==null){

n.setPrevious(null);

n.setObj(obj);

n.setNext(null);

first = n;

last = n;

}else{

//直接往last节点后增加新的节点

n.setPrevious(last);

n.setObj(obj);

n.setNext(null);

last.setNext(n);

last = n;

}

size++;

}

public int size(){

return size;

}

private void rangeCheck(int index){

if(index<0||index>=size){

try {

throw new Exception();

} catch (Exception e) {

e.printStackTrace();

}

}

}

public Object get(int index){   //2

rangeCheck(index);

// 0 1 2 3 4

Node temp = node(index);

if(temp!=null){

return temp.obj;

}

return null;

}

public Node node(int index){

Node temp = null;

if(first!=null){

if (index < (size >> 1)) {

temp = first;

for(int i=0;i<index;i++){

temp = temp.next;

}

}else{

temp = last;

            for (int i = size - 1; i > index; i--){

             temp = temp.previous;

            }

}

}

// LinkedList l;

return temp;

}

public void remove(int index){

Node temp = node(index);

if(temp!=null){

Node up = temp.previous;

Node down = temp.next;

up.next = down;

down.previous = up;

size--;

}

}

public void add(int index,Object obj){

Node temp = node(index);

Node newNode = new Node();

newNode.obj = obj;

if(temp!=null){

Node up = temp.previous;

up.next = newNode;

newNode.previous = up;

newNode.next = temp;

temp.previous = newNode;

size++;

}

}

public static void main(String[] args) {

SxtLinkedList list = new SxtLinkedList();

list.add("aaa");

list.add("bbb");

// list.add(1,"BBBB");

list.add("ccc");

list.add("ddd");

list.add("eee");

// list.remove(1);

System.out.println(list.get(3));

}

}

# 第105集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季105集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# **Map的基本用法**

package cn.bjsxt.collection;

import java.util.HashMap;

import java.util.Map;

/\*\*

 \* 测试Map的基本用法

 \* @author Administrator

 \*

 \*/

public class TestMap {

public static void main(String[] args) {

Map map = new HashMap();

map.put("高琪", new Wife("张曼玉"));

map.put("高琪", new Wife("张小玉"));

map.put("张三", new Wife("杨幂"));

Wife w = (Wife) map.get("高琪");

// map.remove("高琪");

// map.containsKey("张三");

System.out.println(w.name);

}

}

class Wife {

String name;

public Wife(String name){

this.name  = name;

}

}

# **第106集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季106集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# **自定义实现Map**

package cn.bjsxt.collection;

/\*\*

 \*自定义实现Map的功能！

 \*暂不完美！

 \*Map:存放键值对，根据键对象找对应的值对象.键不能重复！

 \*

 \*/

public class SxtMap001 {

SxtEntry[]  arr  = new SxtEntry[990];

int size;

public void put(Object key,Object value){

SxtEntry e = new SxtEntry(key,value);

//解决键值重复的处理

for(int i=0;i<size;i++){

if(arr[i].key.equals(key)){

arr[i].value=value;

return ;

}

}

arr[size++] = e;

}

public Object get(Object key){

for(int i=0;i<size;i++){

if(arr[i].key.equals(key)){

return arr[i].value;

}

}

return null;

}

public boolean containsKey(Object key){

for(int i=0;i<size;i++){

if(arr[i].key.equals(key)){

return true;

}

}

return false;

}

public boolean containsValue(Object value){

for(int i=0;i<size;i++){

if(arr[i].value.equals(value)){

return true;

}

}

return false;

}

public static void main(String[] args) {

SxtMap001 m = new SxtMap001();

m.put("高琪", new Wife("杨幂"));

m.put("高琪", new Wife("李四"));

Wife w = (Wife) m.get("高琪");

System.out.println(w.name);

}

}

class  SxtEntry {

Object key;

Object value;

public SxtEntry(Object key, Object value) {

super();

this.key = key;

this.value = value;

}

# **第109集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季109集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# List\_Map

接口的子类我们自己各自封装了一个类，对这些类可以结合jdk的源码，仔细体会一下，

这连个类的代码变化很少，可以直接参考视频理解一下即可。

# **第110集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季110集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# Set接口的特点以及常用方法

Set接口的实现类常用的有HashSet，特点是：没有是顺序，不可以重复，重复的元素添加进行会出现覆盖的现象。

package com.bjsxt.set;

import java.util.HashSet;

import java.util.Set;

public class TestSet {

public static void main(String[] args) {

Set s = new HashSet();

s.add("hello");

s.add(1);

s.add(3.14);

s.add(new Integer(4));

s.add(new Double(1.2));

s.add(new String("hello"));

System.out.println(s.size() + "个元素。");

System.out.println(s);

}

}

# **第111集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季111集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 自定义个SxtHashSet类的代码为：

package cn.bjsxt.collection;

import java.util.HashMap;

/\*\*

 \* 自定义自己的HashSet

 \* @author Administrator

 \* set的不可重复就是利用了map里面键对象的不可重复！

 \*/

public class SxtHashSet {

HashMap map;

private static final Object PRESENT = new Object();

public SxtHashSet(){

map = new HashMap();

}

public int size(){

return map.size();

}

public void add(Object o){

map.put(o, PRESENT);   //set的不可重复就是利用了map里面键对象的不可重复！

}

public static void main(String[] args) {

SxtHashSet s = new SxtHashSet();

s.add("aaa");

s.add(new String("aaa"));

System.out.println(s.size());

}

}

# **第112集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季112集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 容器之数据存储综合练习（JavaBean）：

## .Employee类

package cn.bjsxt.collection2;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

public class Employee {   //Javabean, Enter实体类

private int id;

private String name;

private int salary;

private String department;

private Date hireDate;

public Employee(int id, String name, int salary, String department,

String hireDate) {

super();

this.id = id;

this.name = name;

this.salary = salary;

this.department = department;

DateFormat format = new SimpleDateFormat("yyyy-MM");

try {

this.hireDate = format.parse(hireDate);

} catch (ParseException e) {

e.printStackTrace();

}

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getSalary() {

return salary;

}

public void setSalary(int salary) {

this.salary = salary;

}

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

public Date getHireDate() {

return hireDate;

}

public void setHireDate(Date hireDate) {

this.hireDate = hireDate;

}

}

## .Test01测试类代码如下：

package cn.bjsxt.collection2;

import java.util.ArrayList;

import java.util.List;

public class Test01 {

public static void main(String[] args) throws Exception {

//一个对象对应了一行记录！

Employee e = new Employee(0301,"高琪",3000,"项目部","2007-10");

Employee e2 = new Employee(0302,"马士兵",3500,"教学部","2006-10");

Employee e3 = new Employee(0303,"裴新",3550,"教学部","2006-10");

List<Employee> list = new ArrayList<Employee>();

list.add(e);

list.add(e2);

list.add(e3);

printEmpName(list);

}

public static void printEmpName(List<Employee> list){

for(int i=0;i<list.size();i++){

System.out.println(list.get(i).getName());

}

}

}

# **第113集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季113集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 数据封装之用容器封装数据测试代码如下：

package cn.bjsxt.collection2;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

public class Test02 {

public static void main(String[] args) {

//一个map对象对应一行记录！！！

Map map = new HashMap();

map.put("id", 0301);

map.put("name", "高琪");

map.put("salary", 3050);

map.put("department","项目部");

map.put("hireDate", "2007-10");

Map map2 = new HashMap();

map2.put("id", 0302);

map2.put("name", "马士兵");

map2.put("salary", 3500);

map2.put("department","教学部");

map2.put("hireDate", "2006-10");

Map map3 = new HashMap();

map3.put("id", 0302);

map3.put("name", "裴新");

map3.put("salary", 3500);

map3.put("department","教学部");

map3.put("hireDate", "2006-10");

List<Map> list = new ArrayList<Map>();

list.add(map);

list.add(map2);

list.add(map3);

printEmpName(list);

}

public static void printEmpName(List<Map> list){

for(int i=0;i<list.size();i++){

Map tempMap = list.get(i);

System.out.println(tempMap.get("name")+"--"+tempMap.get("salary"));

}

}

}

# **第114集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季114集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# Iterator接口：

   所有实现了Collection接口的容器类都有一个iterator方法用以返回一个实现Iterator接口的对象

Iterator对象称作为迭代器，用以方便的对容器内元素的遍历操作，Iterator接口定义了如下方法：

    boolean hashNext();//判断是否有元素没有被遍历

Object next();//返回游标当前位置的元素并将游标移动到下一个位置

    void remove();//删除游标左边的元素，在执行完next之后该操作只能执行一次。

package cn.bjsxt.iterator;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.Iterator;

import java.util.List;

import java.util.Set;

public class Test01 {

public static void main(String[] args) {

List list = new ArrayList();

list.add("aaa");

list.add("bbb");

list.add("ccc");

//通过索引遍历List

for(int i=0;i<list.size();i++){

System.out.println(list.get(i));

}

//通过迭代器遍历List

for(Iterator iter2 = list.iterator();iter2.hasNext();){

String str = (String) iter2.next();

System.out.println(str);

iter2.remove();

iter2.remove();

}

System.out.println(list.size()+"\*\*\*\*\*\*");

Set set = new HashSet();

set.add("高1");

set.add("高2");

set.add("高3");

//通过迭代器遍历Set

// Iterator iter = set.iterator();

// while(iter.hasNext()){

for(Iterator iter = set.iterator();iter.hasNext();){

String str = (String) iter.next();

System.out.println(str);

}

}

}

# **第115集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季115集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# Iterator迭代器如何遍历Map:

package cn.bjsxt.iterator;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Map.Entry;

import java.util.Set;

/\*\*

 \* 测试Map的遍历方式

 \* @author Administrator

 \*

 \*/

public class Test02 {

public static void main(String[] args) {

Map map = new HashMap();

map.put("aa", "aaaa");

map.put("bb", "bbbb");

//遍历Map的第一种方式

Set keys = map.keySet();

for(Iterator iter = keys.iterator();iter.hasNext();){

String keyStr = (String) iter.next();

System.out.println(keyStr+"---"+map.get(keyStr));

}

//遍历Map的第二种方式

Set<Entry> set2 = map.entrySet();

for(Iterator iter = set2.iterator();iter.hasNext();){

Entry e = (Entry) iter.next();

System.out.println(e.getKey()+"---"+e.getValue());

}

}

}

# **第117集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季117集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 泛型的继承

package com.bjsxt.gen02;

/\*\*

 \* 泛型父类，子类为富二代

 \* 1、保留父类泛型 -->子类为泛型

 \* 2、不保留父类泛型 -->子类按需实现

 \*

 \* 属性 及方法 类型 -->随位置而定

 \* 1)、子类重写方法的类型 -->随父类而定

 \* 子类新增方法的类型 -->随子类而定

 \* 2)、子类中使用父类的属性 -->随父类而定

 \* 子类中使用自己的属性 -->随子类而定

 \*

 \*

 \* @author bj

 \*

 \*/

public abstract class Father<T1,T2> {

T1 age;

public abstract void test(T2 name);

}

//保留  -->泛型子类

//1)、全部保留

class C1<T1,T2,A,B> extends Father<T1,T2>{

A Sex;

@Override

public void test(T2 name) {

//this.age -->T1

}

public void test2(A a){

}

}

//2)、部分保留

class C2<T2,A,B> extends Father<Integer,T2>{

@Override

public void test(T2 name) {

//this.age -->Integer

}

}

//不保留  -->按需实现

//1)、具体类型

class C3 extends Father<Integer,String>{

@Override

public void test(String name) {

//this.age -->Integer

}

}

//2)、没有类型 擦除 Object

class C4 extends Father{

@Override

public void test(Object name) {

//this.age -->Object

}

}

package com.bjsxt.gen02;

/\*\*

 \* 泛型接口  与 泛型类  同理

 \* @author bj

 \*

 \*/

public interface Comparator<T> {

//全局常量

public static final int MAX\_VALUE=100;

//公共抽象方法

public abstract void test(T t);

}

//实现

class InterC1<A> implements Comparator{

@Override

public void test(Object t) {

}

}

class InterC2<A> implements Comparator<Integer>{

@Override

public void test(Integer t) {

// TODO Auto-generated method stub

}

}

class InterC3<T,A> implements Comparator<T>{

@Override

public void test(T t) {

}

}

# 泛型的擦除。

package com.bjsxt.gen02;

/\*\*

 \* 自定义泛型类

 \* 1、<> -->单个大写字母，尽可能见名知意

 \* 2、

 \* T Type

 \* K V Key Value

 \* E Element

 \*

 \* 3、注意点:

 \* 泛型不能使用在静态属性上

 \* 指定的类型不能为基本类型

 \* @author bj

 \*

 \*/

// T -->type 类型

public class MyStudent<T> {

private T javase;

//private static T test; //泛型不能使用在静态属性上

public MyStudent() {

}

public MyStudent(T javase) {

this.javase = javase;

}

public T getJavase() {

return javase;

}

public void setJavase(T javase) {

this.javase = javase;

}

}

package com.bjsxt.gen02;

/\*\*

 \* 泛型的擦除: 使用时 |实现|继承 没有指定类型 ，

 \* 类似于 Object ,不等同于 Object

 \* @author bj

 \*

 \*/

public class MyStuApp {

public static void main(String[] args) {

//擦除 -->没有指定 泛型的具体类型

MyStudent student =new MyStudent();

student.setJavase(100); //100 -->int -->Integer -->Object

Object obj=student.getJavase();

test(student);//编译不会类型检查

MyStudent<Object> student2 =new MyStudent<Object>();

//不等于 Object

//test(student2);

}

public static void test(MyStudent<Integer> stu){

}

}

# 总结：

    1、继承|实现  
    1)、保留 -->泛型子类  
    2)、不保留 -->按需实现  可以泛型子类  也可以是非泛型子类  
    属性|方法  -->随声明位置而定  
    2、擦除: 使用|继承|实现  没有指定类型 ，类似于Object.

# **第118集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季118集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 自定义泛型的深入2

package com.bjsxt.gen03;

import java.util.ArrayList;

import java.util.List;

/\*\*

 \* ? -->通配符，类型不确定，用于声明变量|形参上

 \* 不能用在

 \* 1、创建对象

 \* 2、创建泛型类 泛型方法 泛型接口上

 \*

 \*

 \* @author bj

 \*

 \*/

public class WildcardsTest {

public static void main(String[] args) {

//声明

List<?> list =new ArrayList<Integer>();

list =new ArrayList<String>();

list =new ArrayList<Object>();

test(list);

//编译错误 不能创建对象

//list =new ArrayList<?>();

}

public static void test(List<?> list){

}

/\*

//不能用在泛型方法上

public static <?> void test2(List<?> list){

}

\*/

class Test<T>{

}

/\*

 //不能用在创建泛型类

class Test2<?>{

}

\*/

}

## 泛型的上限

package com.bjsxt.gen03;

import java.util.ArrayList;

import java.util.List;

/\*\*

 \* extends : 泛型的上限  <= 即子类或自身

 \* 1、一般用于 限制操作

 \* 2、不能使用在 添加数据上面，一般都是读取操作

 \* 3、规则

 \*  List<Fruit> -->List<? extends Fruit>

 \*  List<Apple> -->List<? extends Fruit>

 \*  List<? extends Apple> -->List<? extends Fruit>

 \*  不能存放

 \*  List<?>

 \*  List<? extends Object>

 \*

 \* @author bj

 \*

 \*/

public class ExtendsTest {

public static void main(String[] args) {

//extends 为上限

Test<Fruit> t1 =new Test<Fruit>();

Test<Apple> t2 =new Test<Apple>();

Test<Pear> t3 =new Test<Pear>();

//规则

List<? extends Fruit> list1 =new ArrayList<Fruit>();

test(list1);

List<Fruit> list2 =new ArrayList<Fruit>();

test(list2);

List<Apple> list3 =new ArrayList<Apple>();

test(list3);

// ? extends Apple

List<? extends Apple> list4 =new ArrayList<FujiApple>();

test(list4);

//? -->为什么错误 ,因为 ? 等同于 ? extends Object

List<?> list5=new ArrayList<Object>();

List<? extends Object> list6=new ArrayList<Object>();

//test(list6);

List<FujiApple> app =new ArrayList<FujiApple>();

test(app);

}

//? extends Fruit

public static void test(List<? extends Fruit> list){

/\*

//不能用于添加数据

list.add(new Apple());

list.add(new FujiApple());

list.add(new Pear());

list.add(new Fruit());

\*/

list.add(null);

}

//泛型类

static class Test<T extends Fruit>{

}

}

package com.bjsxt.gen03;

/\*\*

 \*  继承链

 \*   Object

 \*     |

 \*   Fruit

 \*    /  \

 \* Apple Pear

 \*   |

 \*  FujiApple

 \* @author bj

 \*

 \*/

public class Fruit {

}

class Apple extends Fruit{

}

class Pear extends Fruit{

}

class FujiApple extends Apple{

}

## super,泛型的下限

package com.bjsxt.gen03;

import java.util.ArrayList;

import java.util.List;

/\*\*

 \* super : 泛型的下限  >= 即父类或自身

 \* 1、一般用于 下限操作

 \* 2、能够添加数据上面，不能添加父对象

 \* 3、规则

 \*  List<Fruit> -->List<? super Apple>

 \*  List<Apple> -->List<? super Apple>

 \*  List<? super Fruit> -->List<?super Apple>

 \*  不能存放

 \*  List<? super FujiApple> -->List<?super Apple>

 \*

 \*

 \* @author bj

 \*

 \*/

public class SuperTest {

public static void main(String[] args) {

// >= 即父类或自身

List<Apple> list1 =new ArrayList<Apple>();

test(list1);

List<Fruit> list2 =new ArrayList<Fruit>();

test(list2);

List<Object> list3 =new ArrayList<Object>();

test(list3);

//规则

List<? super Apple> list4 =new ArrayList<Apple>();

test(list4);

List<? super Fruit> list5 =new ArrayList<Object>();

test(list5);

List<? super FujiApple> list6 =new ArrayList<Object>();

//test(list6);

List<?> list7 =new ArrayList<Object>();

//test(list7);

}

public static void test(List<? super Apple> list){

//不能添加 父类对象

list.add(new Apple());

list.add(new FujiApple());

//list.add(new Fruit());

}

}

# 泛型的嵌套。

package com.bjsxt.gen03;

public class Student<T> {

private T score;

public T getScore() {

return score;

}

public void setScore(T score) {

this.score = score;

}

}

package com.bjsxt.gen03;

import java.util.Map.Entry;

import java.util.HashMap;

import java.util.Map;

import java.util.Set;

/\*\*

 \* 泛型嵌套 -->由外到内拆分

 \*

 \*

 \* @author bj

 \*

 \*/

public class StudentApp {

public static void main(String[] args) {

Student<String> stu =new Student<String>();

stu.setScore("优秀");

System.out.println(stu.getScore());

//泛型嵌套

Bjsxt<Student<String>> bjsxt =new Bjsxt<Student<String>>();

bjsxt.setStu(stu);

stu=bjsxt.getStu();

String score =stu.getScore();

System.out.println(score);

//HashMap -->使用泛型嵌套

Map<String,String> map =new HashMap<String,String>();

map.put("a", "java300");

map.put("b", "马士兵javase");

Set<Entry<String,String>> entrySet= map.entrySet();

for(Entry<String,String> entry:entrySet){

String key =entry.getKey();

String value =entry.getValue();

System.out.println(key+"-->"+value);

}

}

}

package com.bjsxt.gen03;

public class Bjsxt<T> {

private T stu;

public T getStu() {

return stu;

}

public void setStu(T stu) {

this.stu = stu;

}

}

package com.bjsxt.gen03;

import java.util.ArrayList;

import java.util.List;

/\*\*

 \* 1、泛型没有多态

 \* 2、泛型没有数组

 \* @author bj

 \*

 \*/

public class Others {

public static void main(String[] args) {

//多态

Fruit f =new Apple();

// 泛型没有多态

//List<Fruit> list =new ArrayList<Apple>();

List<? extends Fruit> list =new ArrayList<Apple>();

//泛型没有数组

//Fruit<String>[] arr =new Fruit<String>[10];

//jdk1.7泛型简化

List<Fruit> list2 =new ArrayList<>();

}

}

# **第119集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季119集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 迭代器与分拣

    概念:迭代器（iterator）有时又称游标（cursor），提供一种方法访问一个容器（container）对象中各个元素，而又不需暴露该对象的内部细节。  
作用:是为容器遍历而生，用以方便的实现对容器内元素的遍历操作。类似于”公交车上的售票员” ，售票员关注公交车上的每个乘客。

package com.bjsxt.iter;

/\*\*

 \* 目标:理解迭代器的原理

 \* 迭代器: 遍历使用

 \*

 \* @author bj

 \*

 \*/

public class SxtSimpleList {

//数组 存储值

private String[] elem={"a","b","c"};

//大小

private int size =elem.length;

public int size(){

return this.size;

}

//计数器 -->指针  游标

private int coursor=-1;

//判断是否存在下一个

public boolean hasNext(){

return coursor+1<size;

}

//获取下一个

public String next(){

coursor++;

return elem[coursor];

}

//删除下一个

public void remove(){

//移动数组元素

System.arraycopy(elem,

coursor+1, elem, coursor,

this.size-(coursor+1));

//实际大小-1

this.size--;

//回退

this.coursor--;

}

public static void main(String[] args) {

SxtSimpleList list =new SxtSimpleList();

if(list.hasNext()){

System.out.println(list.next());

list.remove();

}

if(list.hasNext()){

System.out.println(list.next());

list.remove();

}

if(list.hasNext()){

System.out.println(list.next());

list.remove();

}

//先判断后获取

if(list.hasNext()){ //不会获取值

System.out.println(list.next());

list.remove();

}

System.out.println(list.size());

list =new SxtSimpleList();

while(list.hasNext()){

System.out.println(list.next());

}

}

}

## .容器深入

package com.bjsxt.iter;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Iterator;

/\*\*

 \* 目标:深入迭代器的原理  -->一个容器 可以 创建多个迭代对象

 \* 途径: 使用了 内部类  |匿名内部类

 \*

 \* 深入

 \* 1、使用内部类  实现迭代器

 \* 2、使用Iterable 实现foreach迭代

 \* 3、加入末尾添加元素的方法

 \*

 \* @author bj

 \*

 \*/

public class SxtDeepList implements java.lang.Iterable{

//数组 存储值

private String[] elem=new String[5];

//实际大小

private int size =0;

public int size(){

return this.size;

}

/\*\*

 \* 在末尾添加元素

 \* @param ele

 \*/

public void add(String ele){

if(this.size==elem.length){ //容量不够 -->扩容

elem=Arrays.copyOf(elem, elem.length+5);

}

elem[size] =ele; //数组中加入元素 最后

size++; //实际大小+1

}

//内部类

private class MyIter implements Iterator {

//计数器 -->指针  游标

private int coursor=-1;

//判断是否存在下一个

public boolean hasNext(){

return coursor+1<size;

}

//获取下一个

public String next(){

coursor++;

return elem[coursor];

}

//删除下一个

public void remove(){

//移动数组元素

System.arraycopy(elem,

coursor+1, elem, coursor,

/\*SxtDeepList.this.\*/size-(coursor+1));

//实际大小-1

/\*SxtDeepList.this.\*/size--;

//回退

this.coursor--;

}

}

public Iterator iterator1(){

return new MyIter();

}

public Iterator iterator2(){

class MyIter implements Iterator {

//计数器 -->指针  游标

private int coursor=-1;

//判断是否存在下一个

public boolean hasNext(){

return coursor+1<size;

}

//获取下一个

public String next(){

coursor++;

return elem[coursor];

}

//删除下一个

public void remove(){

//移动数组元素

System.arraycopy(elem,

coursor+1, elem, coursor,

/\*SxtDeepList.this.\*/size-(coursor+1));

//实际大小-1

/\*SxtDeepList.this.\*/size--;

//回退

this.coursor--;

}

}

return new MyIter();

}

public Iterator iterator(){

return new Iterator(){ //创建Iterator迭代器接口 实现类(没有名称)的对象

//计数器 -->指针  游标

private int coursor=-1;

//判断是否存在下一个

public boolean hasNext(){

return coursor+1<size;

}

//获取下一个

public String next(){

coursor++;

return elem[coursor];

}

//删除下一个

public void remove(){

//移动数组元素

System.arraycopy(elem,

coursor+1, elem, coursor,

size-(coursor+1));

//实际大小-1

size--;

//回退

this.coursor--;

}

};

}

public static void main(String[] args) {

SxtDeepList list =new SxtDeepList();

list.add("a");

list.add("b");

list.add("c1");

list.add("c2");

list.add("c3");

list.add("c4");

list.add("c5");

Iterator it =list.iterator();

while(it.hasNext()){ //先判断 后获取

System.out.println(it.next());

//it.remove();

}

System.out.println(list.size());

for(Object str:list){

System.out.println(str);

}

ArrayList list2 =new ArrayList();

list2.add("a");

list2.add("a");

list2.add("a");

for(Object obj:list2){ //foreach

System.out.println(obj);

}

}

}

## .自定义泛型的应用。

package com.bjsxt.iter;

import java.util.Arrays;

import java.util.Iterator;

/\*\*

 \* 目标:使用泛型 ，可以操纵多个类型

 \*

 \*

 \* @author bj

 \*

 \*/

public class SxtArrayList<E> implements java.lang.Iterable<E>{

//数组 存储值

private Object[] elem=new Object[5];

//实际大小

private int size =0;

public int size(){

return this.size;

}

/\*\*

 \* 在末尾添加元素

 \* @param ele

 \*/

public void add(E ele){

if(this.size==elem.length){ //容量不够 -->扩容

elem=Arrays.copyOf(elem, elem.length+5);

}

elem[size] =ele; //数组中加入元素 最后

size++; //实际大小+1

}

public Iterator<E> iterator(){

return new Iterator<E>(){ //创建Iterator迭代器接口 实现类(没有名称)的对象

//计数器 -->指针  游标

private int coursor=-1;

//判断是否存在下一个

public boolean hasNext(){

return coursor+1<size;

}

//获取下一个

public E next(){

coursor++;

return (E)elem[coursor];

}

//删除下一个

public void remove(){

//移动数组元素

System.arraycopy(elem,

coursor+1, elem, coursor,

size-(coursor+1));

//实际大小-1

size--;

//回退

this.coursor--;

}

};

}

public static void main(String[] args) {

SxtArrayList<Integer> list =new SxtArrayList<Integer>();

list.add(1);//int -->Integer

list.add(2);

for(Integer e:list){

System.out.println(e);

}

System.out.println("----------------");

SxtArrayList<String> list2 =new SxtArrayList<String>();

list2.add("马云");

list2.add("马化腾");

list2.add("马士兵");

list2.add("弼马温");

Iterator<String> it =list2.iterator();

while(it.hasNext()){

String e =it.next();

System.out.println(e);

}

}

}

# **第120集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季120集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# HashMap深入原理。

}        分拣存储。

    package com.bjsxt.map;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Set;

/\*\*

 \* 分拣存储: 1:N

 \* 统计 单词 出现的次数

 \* this is a cat and that is a mice and where is the food ?

 \*

 \* 思路

 \* 1、分割字符串

 \* 2、分拣存储

 \* 3、按要求查看 单词出现的次数

 \*

 \* @author bj

 \*

 \*/

public class MapDemo01 {

public static void main(String[] args) {

// 1、分割字符串

String[] arr ="this is a cat and that is a mice and where is the food ?".split(" ");

// 2、分拣存储

Map<String,Integer> map =new HashMap<String,Integer>();

for(String key:arr){

//System.out.println(key );//每个单词

/\*if(!map.containsKey(key)){ //查看是否存在该单词,不存在

map.put(key, 1);

}else{ //存在

map.put(key, map.get(key)+1);

}\*/

Integer value =map.get(key);

if(null==value){ //不存在

map.put(key, 1);

}else{

map.put(key, value+1);

}

}

//3、查看每个单词出现的次数

Set<String> keySet =map.keySet();

//获取对象

Iterator<String> it =keySet.iterator();

while(it.hasNext()){//判断

String key =it.next();

Integer value =map.get(key);

System.out.println(key+"-->"+value);

}

}

}

    .javaBea,Po,Vo

package com.bjsxt.map;

/\*\*

 \* javabean 存储数据 po bo vo  包含setter与getter访问器的类

 \* @author bj

 \*

 \*/

public class Letter {

private String name; //单词

private int count; //次数

//alt+/

public Letter() {

}

public Letter(String name) {

this.name = name;

}

//alt+shif+s -->o

public Letter(String name, int count) {

super();

this.name = name;

this.count = count;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

//setter与getter  访问器

//alt+shift+s -->r -->tab -->enter -->shift+tab -->enter

}

package com.bjsxt.map;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Set;

/\*\*

 \* 分拣存储: 1:N

 \* 统计 单词 出现的次数

 \* this is a cat and that is a mice and where is the food ?

 \*

 \* 思路

 \* 1、分割字符串

 \* 2、分拣存储

 \* 3、按要求查看 单词出现的次数

 \* 4、加入面向对象

 \* @author bj

 \*

 \*/

public class MapDemo02 {

public static void main(String[] args) {

// 1、分割字符串

String[] arr ="this is a cat and that is a mice and where is the food ?".split(" ");

// 2、分拣存储

Map<String,Letter> map =new HashMap<String,Letter>();

for(String key:arr){

/\*

//第一次查看是否 存在 袋子

if(!map.containsKey(key)){ //不存在

map.put(key, new Letter(key));  //准备好袋子

}

//获取袋子

Letter value =map.get(key);

value.setCount(value.getCount()+1);//装东西

\*/

Letter value =map.get(key);

if(null==value){

value =new Letter();

map.put(key, value);

}

value.setCount(value.getCount()+1);//装东西

}

//3、查看每个单词出现的次数

for(String key:map.keySet()){

Letter value =map.get(key);

System.out.println(key+"-->"+value.getCount());

}

}

# **第121集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季121集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# HashMap经典存储，分拣思路与面向对象的思路组合结题。

package com.bjsxt.map;

/\*\*

 \* 学生类

 \*

 \*

 \* @author bj

 \*

 \*/

public class Student {

private String name;//姓名

private String no;//班级

private double score;//成绩

public Student() {

}

public Student(String name, String no, double score) {

super();

this.name = name;

this.no = no;

this.score = score;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getNo() {

return no;

}

public void setNo(String no) {

this.no = no;

}

public double getScore() {

return score;

}

public void setScore(double score) {

this.score = score;

}

}

package com.bjsxt.map;

import java.util.ArrayList;

import java.util.List;

/\*\*

 \* 一个班级 多个学生 (学生列表)

 \* @author bj

 \*

 \*/

public class ClassRoom {

private String no;//班级

private List<Student> stuList;//班级列表

private double total; //总分

public ClassRoom() {

stuList=new ArrayList<Student>();

}

public ClassRoom(String no) {

this();

this.no =no;

}

public ClassRoom(String no, List<Student> stuList, double total) {

super();

this.no = no;

this.stuList = stuList;

this.total = total;

}

public String getNo() {

return no;

}

public void setNo(String no) {

this.no = no;

}

public List<Student> getStuList() {

return stuList;

}

public void setStuList(List<Student> stuList) {

this.stuList = stuList;

}

public double getTotal() {

return total;

}

public void setTotal(double total) {

this.total = total;

}

}

## .测试类代码如下：

package com.bjsxt.map;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.Set;

/\*\*

 \*思考: 定义一个Student类,

 属性:name 姓名,no班号,score 成绩   现在将若干Student对象放入List,请统计出每个班级的总分和平均分

方案: 面向对象+分拣存储(1:N)

 \*

 \*

 \*

 \* @author bj

 \*

 \*/

public class MapDemo03 {

public static void main(String[] args) {

//1、考试

List<Student> stuList =exam();

//2、分析成绩

Map<String,ClassRoom> map =count(stuList);

//3、查看成绩(总分 平均分)

view(map);

}

/\*\*

 \* 查看每个班的总分和平均分  -->遍历 map

 \*/

public static void view(Map<String,ClassRoom> map){

Set<String> keys =map.keySet();

//获取迭代器对象

Iterator<String> keysIt =keys.iterator();

//先判断

while(keysIt.hasNext()){

//再获取

String no =keysIt.next();

ClassRoom room =map.get(no);

//查看总分  计算平均分

double total =room.getTotal();

double avg =total/room.getStuList().size();

System.out.println(no+"-->"+total+"-->"+avg);

}

}

/\*\*

 \* 统计分析

 \* 1、面向对象

 \* 2、分拣存储

 \*/

public static Map<String,ClassRoom> count(List<Student> list){

Map<String,ClassRoom> map =new HashMap<String,ClassRoom>();

//1、遍历List

for(Student stu:list){

String no =stu.getNo(); //班级编号

double score =stu.getScore(); //学号

//2、分拣 查看是否存在 该 编号的班级

//如果不存在，创建班级

ClassRoom room =map.get(no);

if(null==room){

room =new ClassRoom(no);

map.put(no, room);

}

//存在，放入学生

room.getStuList().add(stu) ;//放入学生

room.setTotal(room.getTotal()+score); //计算总分

}

return map;

}

/\*\*

 \* 模拟考试  测试数据 到List中

 \*/

public static List<Student> exam(){

List<Student> list =new ArrayList<Student>();

 //存放学生成绩

 list.add(new Student("老裴","a",85));

 list.add(new Student("裴兜兜","a",86));

 list.add(new Student("裴裴","a",89));

 list.add(new Student("高小三","b",80));

 list.add(new Student("高高","b",80));

 return list;

}

}

# **第122与123集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季122与123集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 冒泡排序

    一个数和下面的数一一比较，交换位置，形成排序后的数据，推到代码如下：

package com.bjsxt.sort.bubble;

import java.util.Arrays;

public class BubbleSort1 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

int[] arr ={9,8,7,6,5};

sort(arr);

}

//第一版本，很简单

public static void sort(int[] arr){

int len =arr.length;

for(int j=0;j<len-1;j++){

System.out.println("第"+(j+1)+"趟");

for(int i=0;i<len-1;i++){

System.out.print("第"+(i+1)+"次");

if(arr[i]>arr[i+1]){

int temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

}

System.out.println(Arrays.toString(arr));

}

}

}

public static void sortSecond(int[] arr){

System.out.println("第一趟");

for(int i=0;i<arr.length-1;i++){

System.out.print("第"+(i+1)+"次");

if(arr[i]>arr[i+1]){

int temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

}

System.out.println(Arrays.toString(arr));

}

System.out.println("第二趟");

for(int i=0;i<arr.length-1;i++){

System.out.print("第"+(i+1)+"次");

if(arr[i]>arr[i+1]){

int temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

}

System.out.println(Arrays.toString(arr));

}

System.out.println("第三趟");

for(int i=0;i<arr.length-1;i++){

System.out.print("第"+(i+1)+"次");

if(arr[i]>arr[i+1]){

int temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

}

System.out.println(Arrays.toString(arr));

}

System.out.println("第四趟");

for(int i=0;i<arr.length-1;i++){

System.out.print("第"+(i+1)+"次");

if(arr[i]>arr[i+1]){

int temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

}

System.out.println(Arrays.toString(arr));

}

}

public static void sortFirst(int[] arr){

System.out.println("第一趟");

for(int i=0;i<arr.length-1;i++){

System.out.print("第"+(i+1)+"次");

if(arr[i]>arr[i+1]){

int temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

}

System.out.println(Arrays.toString(arr));

}

/\*

//第一趟 第一次

System.out.println("第一趟 第一次");

int i=0;

if(arr[i]>arr[i+1]){

int temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

}

System.out.println(Arrays.toString(arr));

System.out.println("第一趟 第二次");

i++;

if(arr[i]>arr[i+1]){

int temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

}

System.out.println(Arrays.toString(arr));

System.out.println("第一趟 第三次");

i++;

if(arr[i]>arr[i+1]){

int temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

}

System.out.println(Arrays.toString(arr));

System.out.println("第一趟 第四次");

i++;

if(arr[i]>arr[i+1]){

int temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

}

System.out.println(Arrays.toString(arr));

\*/

}

}

## .冒泡排序晋级版

package com.bjsxt.sort.bubble;

import java.util.Arrays;

public class BubbleSort2 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

int[] arr ={9,8,7,6,5};

sort(arr);

}

//第二版本，减少每一趟的次数

public static void sort(int[] arr){

int len =arr.length;

for(int j=0;j<len-1;j++){ //趟数

System.out.println("第"+(j+1)+"趟");

for(int i=0;i<len-1-j;i++){ //次数

System.out.print("第"+(i+1)+"次");

if(arr[i]>arr[i+1]){

int temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

}

System.out.println(Arrays.toString(arr));

}

}

}

}

## .冒泡排序最终版本

package com.bjsxt.sort.bubble;

import java.util.Arrays;

/\*\*

 \* 最终版本：考虑存在顺序

 \* @author Administrator

 \*

 \*/

public class BubbleSort {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

int[] arr ={1,2,9,3,4};

sort1(arr);

System.out.println("==========final============");

arr =new int[]{9,1,2,3,4};

sortFinal(arr);

}

//第二版本，减少每一趟的次数

public static void sortFinal(int[] arr){

boolean sorted= true;

int len =arr.length;

for(int j=0;j<len-1;j++){ //趟数

sorted =true; //假定有序

for(int i=0;i<len-1-j;i++){ //次数

if(arr[i]>arr[i+1]){

int temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

sorted =false; //假定失败

}

System.out.println(Arrays.toString(arr));

}

if(sorted){ //减少趟数

break;

}

}

}

//第二版本，减少每一趟的次数

public static void sort1(int[] arr){

int len =arr.length;

for(int j=0;j<len-1;j++){ //趟数

System.out.println("第"+(j+1)+"趟");

for(int i=0;i<len-1-j;i++){ //次数

System.out.print("第"+(i+1)+"次");

if(arr[i]>arr[i+1]){

int temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

}

System.out.println(Arrays.toString(arr));

}

}

}

}

# **第124集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季124集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 引用类型的排序：

package com.bjsxt.sort.innerType;

/\*\*

 \* 内置引用数据类型(常用)的比较

 \* @author Administrator

 \*

 \*/

public class Demo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Integer  a ; //根据基本数据类型大小

Character ch; //根据Unicode编码顺序

String str="abc"; //如果其中一个是例外一个起始开始的子串，返回长度之差

String str2 ="abcd123";  //否则返回第一个不相等的unicode码之差

System.out.println(str.compareTo(str2));

str ="abc";

str2 ="aad";

System.out.println(str.compareTo(str2));

java.util.Date d ;  //根据日期的长整形数比较

}

}

package com.bjsxt.sort.innerType;

import java.util.Arrays;

public class Demo02 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

String[] arr ={"a","abcd","abc","def"};

//从到小排序 降序

boolean sorted= true;

int len =arr.length;

for(int j=0;j<len-1;j++){ //趟数

sorted =true; //假定有序

for(int i=0;i<len-1-j;i++){ //次数

if(((Comparable)arr[i]).compareTo(arr[i+1])<0){

String temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

sorted =false; //假定失败

}

}

if(sorted){ //减少趟数

break;

}

}

System.out.println(Arrays.toString(arr));

}

}

package com.bjsxt.sort.innerType;

import java.util.Arrays;

import java.util.Date;

public class Demo03 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Date[] arr =new Date[3];

arr[0] =new Date();

arr[1] =new Date(System.currentTimeMillis()-1000\*60\*60);

arr[2] =new Date(System.currentTimeMillis()+1000\*60\*60);

//降序

//从大到小排序 降序

boolean sorted= true;

int len =arr.length;

for(int j=0;j<len-1;j++){ //趟数

sorted =true; //假定有序

for(int i=0;i<len-1-j;i++){ //次数

if(((Comparable)arr[i]).compareTo(arr[i+1])<0){

Date temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

sorted =false; //假定失败

}

}

if(sorted){ //减少趟数

break;

}

}

System.out.println(Arrays.toString(arr));

}

}

package com.bjsxt.sort.innerType;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Date;

import java.util.List;

public class Demo04 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Date[] arr =new Date[3];

arr[0] =new Date();

arr[1] =new Date(System.currentTimeMillis()-1000\*60\*60);

arr[2] =new Date(System.currentTimeMillis()+1000\*60\*60);

Utils.sort(arr); //降序

System.out.println(Arrays.toString(arr));

//字符串

String[] arr2 ={"a","abcd","abc","def"};

Utils.sort(arr2);

System.out.println(Arrays.toString(arr2));

System.out.println("==========List排序===========");

//存放容器中

List<String> list =new ArrayList<String>();

list.add("a");

list.add("abcd");

list.add("abc");

list.add("def");

Utils.sort(list);

System.out.println(list);

System.out.println("==========使用Comparator 排序数组===============");

arr2 =new String[]{"a","abcd","abc","def"};

Utils.sort(arr2,new StringComp());

System.out.println(Arrays.toString(arr2));

System.out.println("==========List排序+比较器===========");

list =new ArrayList<String>();

list.add("a");

list.add("abcd");

list.add("abc");

list.add("def");

Utils.sort(list,new StringComp());

System.out.println(list);

}

}

## 泛型的应用。

package com.bjsxt.sort.innerType;

import java.util.Comparator;

import java.util.List;

/\*\*

 \* 排序

 \* @author Administrator

 \*

 \*/

public class Utils {

/\*\*

 \* List的排序+比较器

 \* @param list

 \* @param com

 \*/

public static  <T> void sort(List<T> list,Comparator<T> com){

//第一步:转成数组

Object[] arr =list.toArray();

sort(arr,com);

//第二步:改变容器中对应的值

for(int i=0;i<arr.length;i++){

list.set(i, (T)(arr[i]));

}

}

/\*\*

 \* 数组的排序 （降序）+Comparator接口

 \* @param arr

 \*/

public static <T> void sort(Object[] arr,Comparator<T> com){

//从大到小排序 降序

boolean sorted= true;

int len =arr.length;

for(int j=0;j<len-1;j++){ //趟数

sorted =true; //假定有序

for(int i=0;i<len-1-j;i++){ //次数

if(com.compare((T)arr[i], (T)arr[i+1])<0){

Object temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

sorted =false; //假定失败

}

}

if(sorted){ //减少趟数

break;

}

}

}

/\*\*

 \* 容器排序 (使用泛型方法)

 \*/

public static <T extends Comparable<T>> void sort(List<T> list){

//第一步:转成数组

Object[] arr =list.toArray();

sort(arr);

//第二步:改变容器中对应的值

for(int i=0;i<arr.length;i++){

list.set(i, (T)(arr[i]));

}

}

/\*\*

 \* 数组排序 (使用泛型方法)

 \*/

public static <T extends Comparable<T>> void sort(T[] arr){

//从大到小排序 降序

boolean sorted= true;

int len =arr.length;

for(int j=0;j<len-1;j++){ //趟数

sorted =true; //假定有序

for(int i=0;i<len-1-j;i++){ //次数

if(((Comparable)arr[i]).compareTo(arr[i+1])<0){

T temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

sorted =false; //假定失败

}

}

if(sorted){ //减少趟数

break;

}

}

}

/\*\*

 \* 数组的排序 （降序）

 \* @param arr

 \*/

public static void sort(Object[] arr){

//从大到小排序 降序

boolean sorted= true;

int len =arr.length;

for(int j=0;j<len-1;j++){ //趟数

sorted =true; //假定有序

for(int i=0;i<len-1-j;i++){ //次数

if(((Comparable)arr[i]).compareTo(arr[i+1])<0){

Object temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

sorted =false; //假定失败

}

}

if(sorted){ //减少趟数

break;

}

}

}

}

# **第125集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季124集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 排序的进阶

package com.bjsxt.sort.innerType;

/\*\*

\* 排序规则的业务类

\* @author Administrator

\*

\*/

public class StringComp implements java.util.Comparator<String>{

/\*\*

\* 按长度比较大小

\* 正数 >

\* 负数 <

\* 0 ==

\*/

@Override

public int compare(String o1, String o2) {

int len1 =o1.length();

int len2 =o2.length();

return -(len1-len2);

}

}

package com.bjsxt.sort.innerType;

import java.util.Comparator;

import java.util.List;

/\*\*

\* 排序

\* @author Administrator

\*

\*/

public class Utils {

/\*\*

\* List的排序+比较器

\* @param list

\* @param com

\*/

public static <T> void sort(List<T> list,Comparator<T> com){

//第一步:转成数组

Object[] arr =list.toArray();

sort(arr,com);

//第二步:改变容器中对应的值

for(int i=0;i<arr.length;i++){

list.set(i, (T)(arr[i]));

}

}

/\*\*

\* 数组的排序 （降序）+Comparator接口

\* @param arr

\*/

public static <T> void sort(Object[] arr,Comparator<T> com){

//从大到小排序 降序

boolean sorted= true;

int len =arr.length;

for(int j=0;j<len-1;j++){ //趟数

sorted =true; //假定有序

for(int i=0;i<len-1-j;i++){ //次数

if(com.compare((T)arr[i], (T)arr[i+1])<0){

Object temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

sorted =false; //假定失败

}

}

if(sorted){ //减少趟数

break;

}

}

}

/\*\*

\* 容器排序 (使用泛型方法)

\*/

public static <T extends Comparable<T>> void sort(List<T> list){

//第一步:转成数组

Object[] arr =list.toArray();

sort(arr);

//第二步:改变容器中对应的值

for(int i=0;i<arr.length;i++){

list.set(i, (T)(arr[i]));

}

}

/\*\*

\* 数组排序 (使用泛型方法)

\*/

public static <T extends Comparable<T>> void sort(T[] arr){

//从大到小排序 降序

boolean sorted= true;

int len =arr.length;

for(int j=0;j<len-1;j++){ //趟数

sorted =true; //假定有序

for(int i=0;i<len-1-j;i++){ //次数

if(((Comparable)arr[i]).compareTo(arr[i+1])<0){

T temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

sorted =false; //假定失败

}

}

if(sorted){ //减少趟数

break;

}

}

}

/\*\*

\* 数组的排序 （降序）

\* @param arr

\*/

public static void sort(Object[] arr){

//从大到小排序 降序

boolean sorted= true;

int len =arr.length;

for(int j=0;j<len-1;j++){ //趟数

sorted =true; //假定有序

for(int i=0;i<len-1-j;i++){ //次数

if(((Comparable)arr[i]).compareTo(arr[i+1])<0){

Object temp = arr[i];

arr[i] =arr[i+1];

arr[i+1] =temp;

sorted =false; //假定失败

}

}

if(sorted){ //减少趟数

break;

}

}

}

}

package com.bjsxt.sort.innerType;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

/\*\*

 \* 使用Collections对容器的比较

 \* 1、 public static <T> void sort(List<T> list, Comparator<? super T> c)

 \* 2、public static <T extends Comparable<? super T>> void sort(List<T> list)

 \* void sort(List<T> list)

 \* @author Administrator

 \*

 \*/

public class Demo05 {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

List<String> list =new ArrayList<String>();

list.add("a");

list.add("abcd");

list.add("abc");

list.add("def");

Collections.sort(list,new StringComp());

System.out.println(list);

list =new ArrayList<String>();

list.add("a");

list.add("abcd");

list.add("abc");

list.add("def");

Collections.sort(list);

System.out.println(list);

}

}

# 第126集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季126集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 引用类型的排序，新闻排序。

## 排序的策略：

### 1、实体类 java.lang.Comparable+compareTo

### 2、业务排序java.util.Comparator+compare

      1)解耦，与实体分离

      2）方便，应对多遍的排序规则。

package com.bjsxt.sort.refType;

import java.text.SimpleDateFormat;

import java.util.Date;

/\*\*

\* 新闻条目实体类

\* @author Administrator

\*

\*/

public class NewsItem implements java.lang.Comparable<NewsItem>{

//标题

private String title;

//点击量

private int hits;

//时间

private Date pubTime;

public NewsItem() {

}

public NewsItem(String title, int hits, Date pubTime) {

super();

this.title = title;

this.hits = hits;

this.pubTime = pubTime;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public int getHits() {

return hits;

}

public void setHits(int hits) {

this.hits = hits;

}

public Date getPubTime() {

return pubTime;

}

public void setPubTime(Date pubTime) {

this.pubTime = pubTime;

}

//时间降序 +点击量升序+标题降序

@Override

public int compareTo(NewsItem o) {

int result =0;

//比较 时间

result =-this.pubTime.compareTo(o.pubTime); //降序

if(0==result){ //时间相同

//点击量

result =this.hits-o.hits; //升序

if(0==result){ //点击量相同

//标题

result=-this.title.compareTo(o.title);//降序

}

}

return result;

}

@Override

public String toString() {

StringBuilder sb =new StringBuilder();

sb.append("标题:").append(this.title);

sb.append(",时间:").append(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(this.pubTime));

sb.append(",点击量:").append(this.hits).append("\n");

return sb.toString();

}

}

## 测试。

package com.bjsxt.sort.refType;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Date;

import java.util.List;

import com.bjsxt.sort.innerType.Utils;

/\*\*

\* 使用Collections

\* @author Administrator

\*

\*/

public class NewsItemApp {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

List<NewsItem> news=new ArrayList<NewsItem>();

news.add(new NewsItem("美国后怕了，逃跑了悲剧了",50,new Date(System.currentTimeMillis()-1000\*60\*60)));

news.add(new NewsItem("中国登上钓鱼岛了，全国欢呼了",100,new Date()));

news.add(new NewsItem("小日本终于听话了，泪流满面笑了",60,new Date(System.currentTimeMillis()-1000\*60\*60)));

System.out.println("排序前:"+news);

//排序

//Collections.sort(news);

Utils.sort(news);

System.out.println("排序后"+news);

}

}

# 商品的排序：

package com.bjsxt.sort.refType;

/\*\*

\* 实体类

\* @author Administrator

\*

\*/

public class Goods {

//商品名称

private String name;

//价格

private double price;

//收藏量

private int fav;

public Goods() {

}

public Goods(String name, double price, int fav) {

super();

this.name = name;

this.price = price;

this.fav = fav;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public int getFav() {

return fav;

}

public void setFav(int fav) {

this.fav = fav;

}

@Override

public String toString() {

return "商品名:"+name+",收藏量"+this.fav+",价格:"+this.price+"\n";

}

}

## 实现Comparable接口，提高比较的条件。

package com.bjsxt.sort.refType;

/\*\*

\* 按价格排序的业务类 （降序）

\* @author Administrator

\*

\*/

public class GoodsPriceComp implements java.util.Comparator<Goods> {

@Override

public int compare(Goods o1, Goods o2) {

return -(o1.getPrice()-o2.getPrice()>0?1:(o1.getPrice()==o2.getPrice()?0:-1));

}

}

## 测试

package com.bjsxt.sort.refType;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class GoodsApp {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

List<Goods> list =new ArrayList<Goods>();

list.add(new Goods("老马视频",100,2000));

list.add(new Goods("老高视频",50,2000));

list.add(new Goods("老裴视频",1000,1000));

System.out.println("排序前:"+list);

// Collections.sort(list,new GoodsPriceComp());

Collections.sort(list,new GoodsFavComp());

System.out.println("排序后:"+list);

}

}

## 根据收藏量进行排序的标准。

package com.bjsxt.sort.refType;

/\*\*

\* 按收藏量排序的业务类 （升序）

\* @author Administrator

\*

\*/

public class GoodsFavComp implements java.util.Comparator<Goods> {

@Override

public int compare(Goods o1, Goods o2) {

return o1.getFav()-o2.getFav();

}

}

# **第127集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季127集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 引用类型\_排序容器\_TreeSet与TreeMap

    TreeSet：元素不可以重复，元素可以排序，添加数据时自动进行排序，TreeSet存储数据尽量不要修改，实例代码如下：

package com.bjsxt.sort.col;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class Person {

private final String name;//名称

private final int handsome;//帅气指数

public Person() {

name =null;

handsome =0;

}

public Person(String name, int handsome) {

super();

this.name = name;

this.handsome = handsome;

}

public String getName() {

return name;

}

public int getHandsome() {

return handsome;

}

@Override

public String toString() {

return "姓名:"+this.name+",帅气指数:"+this.handsome+"\n";

}

}

## 测试代码：

package com.bjsxt.sort.col;

import java.util.TreeSet;

/\*\*

 \* 提供了 解耦的方式:业务排序类

 \* @author Administrator

 \*

 \*/

public class TreeSetDemo {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Person p1 =new Person("您",100);

Person p2 =new Person("刘德华",1000);

Person p3 =new Person("梁朝伟",1200);

Person p4 =new Person("老裴",50);

//依次存放到TreeSet容器中,使用排序的业务类(匿名内部类)

TreeSet<Person> persons =new TreeSet<Person>(

new java.util.Comparator<Person>(){

@Override

public int compare(Person o1, Person o2) {

return -(o1.getHandsome()-o2.getHandsome());

}

}

);

persons.add(p1);

//TreeSet 在添加数据时排序

persons.add(p2);

persons.add(p3);

persons.add(p4);

System.out.println(persons);

/\*

//改变数据

p4.setHandsome(100);

p4.setName("您");

\*/

//p4 与p1 内容重复

System.out.println(persons);

}

}

## 案例2,Worker代码如下：

package com.bjsxt.sort.col;

public class Worker implements java.lang.Comparable<Worker> {

//工种

private String type;

//工资

private double salary;

public Worker() {

// TODO Auto-generated constructor stub

}

public Worker(String type, double salary) {

super();

this.type = type;

this.salary = salary;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

/\*\*

 \* 按工资升序

 \*/

@Override

public int compareTo(Worker o) {

return this.salary>o.salary?1:( this.salary==o.salary?0:-1);

}

@Override

public String toString() {

return "工种:"+this.type+",工资:"+this.salary+"\n";

}

}

## 测试类代码如下：

package com.bjsxt.sort.col;

import java.util.TreeSet;

/\*\*

 \* 实体类实现Comparable 接口的应用

 \* @author Administrator

 \*

 \*/

public class TreeSetDemo2 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Worker w1 =new Worker("垃圾回收员",12000);

Worker w2 =new Worker("农民工",8000);

Worker w3 =new Worker("程序猿",5000);

TreeSet<Worker> employees =new TreeSet<Worker>();

employees.add(w1);

employees.add(w2);

employees.add(w3);

System.out.println(employees);

}

}

# TreeMap:要求键可以排序，与上TreeSet同理。

package com.bjsxt.sort.col;

import java.util.Set;

import java.util.TreeMap;

public class TreeMapDemo {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Person p1 =new Person("您",100);

Person p2 =new Person("刘德华",1000);

Person p3 =new Person("梁朝伟",1200);

Person p4 =new Person("老裴",50);

TreeMap<Person,String> map =new TreeMap<Person,String>(new java.util.Comparator<Person>(){

@Override

public int compare(Person o1, Person o2) {

return -(o1.getHandsome()-o2.getHandsome());

}

} );

map.put(p1, "bjsxt");

map.put(p2, "bjsxt");

map.put(p3, "bjsxt");

map.put(p4, "bjsxt");

//查看键

Set<Person> persons =map.keySet();

System.out.println(persons);

}

}

package com.bjsxt.sort.col;

import java.util.TreeSet;

/\*\*

 \* 实体类实现Comparable 接口的应用

 \* @author Administrator

 \*

 \*/

public class TreeSetDemo2 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Worker w1 =new Worker("垃圾回收员",12000);

Worker w2 =new Worker("农民工",8000);

Worker w3 =new Worker("程序猿",5000);

TreeSet<Worker> employees =new TreeSet<Worker>();

employees.add(w1);

employees.add(w2);

employees.add(w3);

System.out.println(employees);

}

}

# **第128集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季128集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# Collections工具类常见方法的使用。

package com.bjsxt.sort.util;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

/\*\*

 \*1、 binarySearch(List<? extends Comparable<? super T>> list, T key)   容器有序

 \*2、sort(List<T> list)

     sort(List<T> list, Comparator<? super T> c)

 \*3、reverse(List<?> list)

  4、shuffle(List<?> list) 洗牌

  5、swap(List<?> list, int i, int j)

 \* @author Administrator

 \*

 \*/

public class CollectionsDemo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

List<Integer> cards =new ArrayList<Integer>();

//shuffle 洗牌 模拟斗地主

for(int i=0;i<54;i++){

cards.add(i);

}

//洗牌

Collections.shuffle(cards) ;

//依次发牌

List<Integer> p1 =new ArrayList<Integer>();

List<Integer> p2 =new ArrayList<Integer>();

List<Integer> p3 =new ArrayList<Integer>();

List<Integer> last =new ArrayList<Integer>();

for(int i=0;i<51;i+=3){

p1.add(cards.get(i));

p2.add(cards.get(i+1));

p3.add(cards.get(i+2));

}

//最后三张为底牌

last.add(cards.get(51));

last.add(cards.get(52));

last.add(cards.get(53));

System.out.println("第一个人:"+p1);

System.out.println("第二个人:"+p2);

System.out.println("第三个人:"+p3);

System.out.println("底牌为:"+last);

}

//反转

public static void test1(){

List<Integer> list =new ArrayList<Integer>();

list.add(1);

list.add(2);

list.add(3);

list.add(4);

System.out.println(list);

Collections.reverse(list);

System.out.println("反转之后"+list);

}

}

# **第129集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季129集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 容器的收尾工做：

    队列Queue,与Deque

package com.bjsxt.others.que;

import java.util.ArrayDeque;

import java.util.Queue;

/\*\*

 \* 使用队列模拟银行存款业务

 \* @author Administrator

 \*

 \*/

public class Demo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Queue<Request> que =new ArrayDeque<Request>();

//模拟排队情况

for(int i=0;i<10;i++){

final int num =i;

que.offer(new Request(){

@Override

public void deposit() {

System.out.println("第"+num+"个人，办理存款业务，存款额度为:"+(Math.random()\*10000));

}

});

}

dealWith(que);

}

//处理业务

public static void dealWith(Queue<Request> que){

Request req =null;

while(null!=(req=que.poll())){

req.deposit();

}

}

}

interface Request{

//存款

void deposit();

}

## 使用队列实现自己定义堆栈：（弹，压，获取头）

package com.bjsxt.others.que;

import java.util.ArrayDeque;

import java.util.Deque;

/\*\*

 \* 使用队列实现自定义堆栈

 \* 1、弹

 \* 2、压

 \* 3、获取头

 \* @author Administrator

 \*

 \* @param <E>

 \*/

public class MyStack<E> {

//容器

private Deque<E> container =new ArrayDeque<E>();

//容量

private int cap;

public MyStack(int cap) {

super();

this.cap = cap;

}

//压栈

public boolean push(E e){

if(container.size()+1>cap){

return false;

}

return container.offerLast(e);

}

//弹栈

public E pop(){

return container.pollLast();

}

//获取

public E peek(){

return container.peekLast();

}

public int size(){

return this.container.size();

}

}

package com.bjsxt.others.que;

//测试自定义堆栈

public class Demo02 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

MyStack<String> backHistory =new MyStack<String>(3);

backHistory.push("www.baidu.com");

backHistory.push("www.google.com");

backHistory.push("www.sina.com");

backHistory.push("www.bjsxt.cn");

System.out.println("大小："+backHistory.size());

//遍历

String item=null;

while(null!=(item=backHistory.pop())){

System.out.println(item);

}

}

}

# **第130集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季130集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# Enumeration接口：

package com.bjsxt.others.en;

import java.util.Enumeration;

import java.util.Vector;

/\*\*

 \* Enumeration 的使用

 \* 1、判断  hasMoreElements()

 \* 2、获取 nextElement()

 \*

 \* Vector 的 elements()方法

 \*

 \*

 \* @author Administrator

 \*

 \*/

public class Demo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Vector<String> vector =new Vector<String>();

vector.add("javase");

vector.add("html");

vector.add("oracle");

//遍历该Vector

Enumeration<String> en =vector.elements();

while(en.hasMoreElements()){

System.out.println(en.nextElement());

}

}

}

package com.bjsxt.others.en;

import java.util.StringTokenizer;

/\*\*

 \* Enumeration 子类

 \* StringTokenizer:String split() 字符串分割

 \* 不支持正则表达式

 \*

 \* StringTokenizer(String str, String delim)

 \* @author Administrator

 \*

 \*/

public class Demo02 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

String emailStr="bjsxt@163.com;bjsxt@qq.com;bjsxt@sohu.com";

StringTokenizer token =new StringTokenizer(emailStr,";");

//遍历获取

while(token.hasMoreElements()){

System.out.println(token.nextElement());

}

}

}

# **第131集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季131集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# Hashtable与HashMap区别：

    1)主要：Hashtable线程安全，同步，效率低下

HashMap线程不安全，非同步，效率相对高。

     2）父类：Hashtable 是Dictionary,HashMap是AbstractMap

     3)null:Hashtable键与值不能为null

HashMap键最多一个null,值可以多个null.

# Properties代码的如下：

## .作用：读取资源配置文件

## .键与值只能为字符串。

## .方法：setProperty(String key,String value);

             getProperty(String key);

              getProperty(String key,String defaultValue);

package com.bjsxt.others.pro;

import java.util.Properties;

/\*\*

 \* Properties 资源配置文件的读写

 \* 1、key 与value 只能为字符串

 \* 2、存储与读取

 \* setProperty(String key, String value)

 \* getProperty(String key, String defaultValue)

 \* @author Administrator

 \*

 \*/

public class Demo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//创建对象

Properties pro =new Properties();

//存储

pro.setProperty("driver", "oracle.jdbc.driver.OracleDriver");

//pro.setProperty("url", "jdbc:oracle:thin:@localhost:1521:orcl");

pro.setProperty("user", "scott");

pro.setProperty("pwd", "tiger");

//获取

String url =pro.getProperty("url","test");

System.out.println(url);

}

}

## .输出到文件

package com.bjsxt.others.pro;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.Properties;

/\*\*

 \* 使用Properties 输出到文件

 \* 资源配置文件:

 \*

 \* 1、.properties

 \* store(OutputStream out, String comments)

store(Writer writer, String comments)

   2、.xml

   storeToXML(OutputStream os, String comment)  :UTF-8字符集

   storeToXML(OutputStream os, String comment, String encoding)

 \* @author Administrator

 \*

 \*/

public class Demo02 {

/\*\*

 \* @param args

 \* @throws IOException

 \* @throws FileNotFoundException

 \*/

public static void main(String[] args) throws FileNotFoundException, IOException {

//创建对象

Properties pro =new Properties();

//存储

pro.setProperty("driver", "oracle.jdbc.driver.OracleDriver");

pro.setProperty("url", "jdbc:oracle:thin:@localhost:1521:orcl");

pro.setProperty("user", "scott");

pro.setProperty("pwd", "tiger");

//存储到e:/others  绝对路径  盘符:

//pro.store(new FileOutputStream(new File("e:/others/db.properties")), "db配置");

//pro.storeToXML(new FileOutputStream(new File("e:/others/db.xml")), "db配置");

//使用相对路径 当前的工程

// pro.store(new FileOutputStream(new File("db.properties")), "db配置");

// pro.store(new FileOutputStream(new File("src/db.properties")), "db配置");

pro.store(new FileOutputStream(new File("src/com/bjsxt/others/pro/db.properties")), "db配置");

}

}

package com.bjsxt.others.pro;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.Properties;

/\*\*

 \* 使用Properties读取配置文件

 \* 资源配置文件:

 \* 使用相对与绝对路径读取

 \* load(InputStream inStream)

   load(Reader reader)

   loadFromXML(InputStream in)

 \* @author Administrator

 \*

 \*/

public class Demo03 {

/\*\*

 \* @param args

 \* @throws IOException

 \* @throws FileNotFoundException

 \*/

public static void main(String[] args) throws FileNotFoundException, IOException {

Properties pro=new Properties();

//读取 绝对路径

//pro.load(new FileReader("e:/others/db.properties"));

//读取 相对路径

pro.load(new FileReader("src/com/bjsxt/others/pro/db.properties"));

System.out.println(pro.getProperty("user", "bjsxt"));

}

}

package com.bjsxt.others.pro;

import java.io.IOException;

import java.util.Properties;

/\*\*

 \* 使用类相对路径读取配置文件

 \*  bin

 \* @author Administrator

 \*

 \*/

public class Demo04 {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

Properties pro =new Properties();

//类相对路径的 / bin

//pro.load(Demo04.class.getResourceAsStream("/com/bjsxt/others/pro/db.properties"));

//"" bin

pro.load(Thread.currentThread().getContextClassLoader().getResourceAsStream("com/bjsxt/others/pro/db.properties"));

System.out.println(pro.getProperty("user", "bjsxt"));

}

}

# **第132集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季132集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com](http://www.bjsxt.xn--com-003b/)

# 引用的分类：（强，软，弱，虚）：

package com.bjsxt.others.three;

import java.lang.ref.WeakReference;

/\*\*

 \* 引用分类:强、软、弱、虚

 \* 强与弱引用

 \* @author Administrator

 \*

 \*/

public class RefDemo {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//字符串常量池

String str =new String("bjsxt is very good");

//弱引用 管理 对象

WeakReference<String> wr =new WeakReference<String>(str);

System.out.println("gc运行前:"+wr.get());

//断开引用

str =null;

//通知回收

System.gc();

System.runFinalization();

//对象被回收

System.out.println("gc运行后:"+wr.get());

}

public static void testStrong(){

//字符串常量池  共享(不能回收)

String str ="bjsxt is very good";

//弱引用 管理 对象

WeakReference<String> wr =new WeakReference<String>(str);

System.out.println("gc运行前:"+wr.get());

//断开引用

str =null;

//通知回收

System.gc();

System.runFinalization();

System.out.println("gc运行后:"+wr.get());

}

}

## weakHashMap

package com.bjsxt.others.three;

import java.util.WeakHashMap;

/\*\*

 \* WeakHashMap 键为弱类型，gc运行立即回收

 \* @author Administrator

 \*

 \*/

public class WeakHashMapDemo {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

WeakHashMap<String,String> map =new WeakHashMap<String,String>();

//测试数据

//常量池对象，不会回收

map.put("abc", "a");

map.put("d", "test");

//gc运行 已被回收

map.put(new String("bjsxt"), "c");

map.put(new String("dsf"), "d");

//通知回收

System.gc();

System.runFinalization();

System.out.println(map.size());

}

}

## IdentityHashMapDemo代码如下：

package com.bjsxt.others.three;

import java.util.IdentityHashMap;

/\*\*

 \* IdentityHashMap 键比较地址去重

 \* @author Administrator

 \*

 \*/

public class IdentityHashMapDemo {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

IdentityHashMap<String ,String> map =new IdentityHashMap<String,String>();

//常量池中的"a"

map.put("a", "a1");

map.put("a", "a2");

System.out.println(map.size());

map.put(new String("a"), "a3");

map.put(new String("a"), "a4");

System.out.println(map.size());

}

}

## EnumMap.

package com.bjsxt.others.three;

import java.util.EnumMap;

/\*\*

 \* EnumMap要求键为枚举

 \* @author Administrator

 \*

 \*/

public class EnumMapDemo {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

EnumMap<Season,String> map =new EnumMap<Season,String>(Season.class);

//存放值

map.put(Season.SPRING, "春困");

map.put(Season.SUMMER, "夏无力");

map.put(Season.AUTUMN, "秋乏");

map.put(Season.WINTER, "冬眠");

System.out.println(map.size());

}

}

//季节

enum Season{

SPRING,SUMMER,AUTUMN,WINTER

}

# **第133集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季133集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com](http://www.bjsxt.xn--com-003b/)

# 同步控制：

    package com.bjsxt.others.synread;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

/\*\*

 \* 使用Collections管理同步 容器

 \* synchronizedList()

synchronizedSet()

synchronizedMap()

 \* @author Administrator

 \*

 \*/

public class Demo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

List<String> list =new ArrayList<String>();

list.add("a");

list.add("b");

//list可以同步

List<String> synList =Collections.synchronizedList(list);

System.out.println("线程安全的list制作完毕");

}

}

# 只读设置：

package com.bjsxt.others.synread;

import java.util.Collections;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Set;

/\*\*

 \* 只读设置

 \* emptyXxx()  空的不可变的集合

 \* 1、emptyList()

 \*    emptyMap()

 \*    emptySet()

2、singletonXxx() 一个元素不可变的集合

singleton(T o)

singletonList(T o)

singletonMap(K key, V value)

3、unmodifiableXxx() 不可变容器

unmodifiableList(List<? extends T> list)

unmodifiableSet(Set<? extends T> s)

unmodifiableMap(Map<? extends K,? extends V> m)

 \* @author Administrator

 \*

 \*/

public class Demo02 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Map<String,String> map =new HashMap<String,String>();

map.put("test", "test");

map.put("bjsxt", "bjsxt");

//只读控制

Map<String,String> map2 =Collections.unmodifiableMap(map);

//map2.put("a", "a"); //不能操作

System.out.println(map2.size());

//一个元素的容器测试

List<String> list =Collections.singletonList(new String());

list.add("test");

//list.add("bjsxt"); //只能包含一个元素的容器

}

public static Set<String> oper(Set<String> set){

if(null==set){

return Collections.EMPTY\_SET; //外部获取避免NullPointerException

}

//操作

return set;

}

}

# **第134集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季134与135集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com](http://www.bjsxt.xn--com-003b/)

# 工作当中的常见组件：

    Google Guava.

    134集是Guava环境的搭建，135集是Guava的使用。

package com.bjsxt.others.guava;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import com.google.common.collect.ImmutableList;

/\*\*

 \* 只读设置

 \* @author Administrator

 \*

 \*/

public class Demo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

List<String> list =new ArrayList<String>();

list.add("a");

list.add("b");

list.add("c");

//对原有的list进行包装，相等于原有List的一个视图，快照，不够安全

List<String> readList =Collections.unmodifiableList(list);

//java.lang.UnsupportedOperationException

//readList.add("d");

list.add("d"); //改变原有List 视图也一起改变

//对比 查看 初始化List guava对只读设置 安全可靠，并且相对简单

List<String> immutableList =ImmutableList.of("a", "b", "c");

immutableList.add("d");//java.lang.UnsupportedOperationException

}

}

## 函数式编程。

package com.bjsxt.others.guava;

import java.text.SimpleDateFormat;

import java.util.Collection;

import java.util.List;

import java.util.Set;

import com.google.common.base.Function;

import com.google.common.base.Functions;

import com.google.common.base.Predicate;

import com.google.common.collect.Collections2;

import com.google.common.collect.Lists;

import com.google.common.collect.Sets;

/\*\*

 \* 函数式编程 :解耦

 \* 1、Predicate

 \* 2、Function

 \*

 \* 工具:Collections2.filter() 过滤器

 \* Collections2.transfer() 转换

 \* Functions.compose()组合式函数编程

 \* @author Administrator

 \*

 \*/

public class Demo02 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//组合式函数编程

//确保容器中的字符串长度不超过5，超过进行截取，后全部大写

List<String> list =Lists.newArrayList("bjsxt","good","happiness");

//确保容器中的字符串长度不超过5，超过进行截取

Function<String,String> f1 =new Function<String,String>(){

@Override

public String apply(String input) {

return input.length()>5?input.substring(0,5):input;

}

};

//转成大写

Function<String,String> f2 =new Function<String,String>(){

@Override

public String apply(String input) {

return input.toUpperCase();

}

};

//String =f2(f1(String))

Function<String,String> f =Functions.compose(f1, f2);

Collection<String> resultCol =Collections2.transform(list, f);

for(String temp:resultCol){

System.out.println(temp);

}

}

/\*\*

 \* 转换

 \*/

public static void test2(){

//类型转换

Set<Long> timeSet =Sets.newHashSet();

timeSet.add(10000000L);

timeSet.add(99999999999999999L);

timeSet.add(2000000000L);

Collection<String> timeStrCol =Collections2.transform(timeSet, new Function<Long,String>(){

@Override

public String apply(Long input) {

return new SimpleDateFormat("yyyy-MM-dd").format(input);

}});

for(String temp:timeStrCol){

System.out.println(temp);

}

}

/\*\*

 \* 过滤器

 \*/

public static void test1(){

//创建List 静态初始化

List<String> list =Lists.newArrayList("moom","son","dad","bjsxt","refer");

//找出回文 palindrome  backwords  mirror words

//匿名内部类对象: 匿名内部类，同时创建类对象

Collection<String> palindromeList =Collections2.filter(list, new Predicate<String>(){

@Override

public boolean apply(String input) {

//业务逻辑

return new StringBuilder(input).reverse().toString().equals(input);

}

});

for(String temp:palindromeList){

System.out.println(temp);

}

}

}

## 约束条件的加入。

package com.bjsxt.others.guava;

import java.util.Set;

import com.google.common.base.Preconditions;

import com.google.common.collect.Constraint;

import com.google.common.collect.Constraints;

import com.google.common.collect.Sets;

/\*\*

 \* 加入约束条件:非空、长度验证

 \* Constraint

 \* Preconditions

 \* Constraints

 \* @author Administrator

 \*

 \*/

public class Demo03 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Set<String> sets =Sets.newHashSet();

//创建约束

Constraint<String> constraint =new Constraint<String>(){

@Override

public String checkElement(String element) {

//非空验证

Preconditions.checkNotNull(element);

//长度验证 5-20为字符串

Preconditions.checkArgument(element.length()>=5 && element.length()<=20);

return element;

}

};

Set<String> cs =Constraints.constrainedSet(sets, constraint);

//cs.add(null); //java.lang.NullPointerException

// cs.add("good"); //java.lang.IllegalArgumentException

cs.add("bjsxt");

for(String str:cs){

System.out.println(str);

}

}

}

## 集合的引入：交集，差集，并集

package com.bjsxt.others.guava;

import java.util.Set;

import com.google.common.collect.Sets;

import com.google.common.collect.Sets.SetView;

/\*\*

 \* 集合的操作:交集、差集、并集

 \* Sets.intersection()

 \* Sets.difference()

 \* Sets.union();

 \* @author Administrator

 \*

 \*/

public class Demo04 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Set<Integer> sets =Sets.newHashSet(1,2,3,4,5,6);

Set<Integer> sets2 =Sets.newHashSet(3,4,5,6,7,8,9);

//交集

System.out.println("交集为:");

SetView<Integer> intersection =Sets.intersection(sets, sets2);

for(Integer temp:intersection){

System.out.println(temp);

}

//差集

System.out.println("差集为:");

SetView<Integer> diff =Sets.difference(sets, sets2);

for(Integer temp:diff){

System.out.println(temp);

}

//并集

System.out.println("并集为:");

SetView<Integer> union =Sets.union(sets, sets2);

for(Integer temp:union){

System.out.println(temp);

}

}

}

# **第135集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季135，136与137集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com](http://www.bjsxt.xn--com-003b/)

# Guava的进阶。

package com.bjsxt.others.guava;

import java.util.Set;

import com.google.common.collect.HashMultiset;

import com.google.common.collect.Multiset;

/\*\*

 \* 统计单词出现的次数

 \* 1、HashMap 分拣存储+面向对象思维  -->判断

 \* 2、Multiset :无序+可重复   .count()  增强了可读性 +操作简单

 \* @author Administrator

 \*

 \*/

public class Demo05 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

String str ="this is a cat and that is a mice where is the food";

//分割字符串

String[] strArray =str.split(" ");

//存储到Multiset中

Multiset<String> set =HashMultiset.create();

for(String strTemp:strArray){

set.add(strTemp);

}

//获取所有的单词 Set

Set<String> letters =set.elementSet();

for(String temp:letters){

System.out.println(temp+"-->"+set.count(temp));

}

}

}

## 实例2

package com.bjsxt.others.guava;

import java.util.Collection;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Set;

import com.google.common.collect.ArrayListMultimap;

import com.google.common.collect.Multimap;

/\*\*

 \* 分析查看 教师  教授的每门课程

 \* Multimap :key-value  key可以重复

 \*

 \*

 \* @author Administrator

 \*

 \*/

public class Demo06 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Map<String,String> cours =new HashMap<String,String>();

//加入测试数据

cours.put("改革开放", "邓爷爷");

cours.put("三个代表", "江主席");

cours.put("科学发展观", "胡主席");

cours.put("和谐社会", "胡主席");

cours.put("八荣八耻", "胡主席");

cours.put("....", "习主席");

//Multimap

Multimap<String,String> teachers =ArrayListMultimap.create();

//迭代器

Iterator<Map.Entry<String,String>> it =cours.entrySet().iterator();

while(it.hasNext()){

Map.Entry<String,String> entry =it.next();

String key =entry.getKey(); //课程

String value =entry.getValue(); //教师

//教师 -->课程

teachers.put(value, key);

}

//查看Multimap

Set<String> keyset =teachers.keySet();

for(String key:keyset){

Collection<String> col =teachers.get(key);

System.out.println(key+"-->"+col);

}

}

}

## Guava对HashMap的进一步实现

package com.bjsxt.others.guava;

import com.google.common.collect.BiMap;

import com.google.common.collect.HashBiMap;

/\*\*

 \* HashMap 键唯一，值可以重复

 \* BiMap:双向Map(Bidirectional Map ) 键与值都不能重复(unique -valued map)

 \* @author Administrator

 \*

 \*/

public class Demo07 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

BiMap<String,String> bimap=HashBiMap.create();

bimap.put("bjsxt", "bjsxt@sina.com");

bimap.put("good","good@qq.com");

//通过邮箱找用户

String user =bimap.inverse().get("good@qq.com");

System.out.println(user);

System.out.println(bimap.inverse().inverse()==bimap);

}

}

package com.bjsxt.others.guava;

import java.util.Map;

import java.util.Set;

import com.google.common.collect.HashBasedTable;

import com.google.common.collect.Table;

import com.google.common.collect.Table.Cell;

import com.google.common.collect.Tables;

/\*\*

 \* 双键的Map -->Table -->rowKey+columnKey+value

 \* 1、方法

 \*  所有的行数据:cellSet()

 \*  所有的学生: rowKeySet()

 \*  所有的课程:columnKeySet()

 \*  所有的成绩: values()

 \*  学生对应的课程: rowMap() +get(学生)

 \*   row(学生)

 \*  课程对应的学生: columnMap +get(课程)

 \*   column(课程)

 \* @author Administrator

 \*

 \*/

public class Demo08 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Table<String,String,Integer> tables=HashBasedTable.create();

//测试数据

tables.put("a", "javase", 80);

tables.put("b", "javase", 90);

tables.put("a", "oracle", 100);

tables.put("c", "oracle", 95);

//所有的行数据

Set<Cell<String,String,Integer>> cells =tables.cellSet();

for(Cell<String,String,Integer> temp:cells){

System.out.println(temp.getRowKey()+"-->"+temp.getColumnKey()+"-->"+temp.getValue());

}

System.out.println("==========学生查看成绩==============");

System.out.print("学生\t");

//所有的课程

Set<String> cours =tables.columnKeySet();

for(String t:cours){

System.out.print(t+"\t");

}

System.out.println();

//所有的学生

Set<String> stus =tables.rowKeySet();

for(String stu:stus){

System.out.print(stu+"\t");

Map<String,Integer> scores =tables.row(stu);

for(String c:cours){

System.out.print(scores.get(c)+"\t");

}

System.out.println();

}

System.out.println("==========课程查看成绩==============");

System.out.print("课程\t");

//所有的学生

Set<String> stuSet =tables.rowKeySet();

for(String t:stuSet){

System.out.print(t+"\t");

}

System.out.println();

//所有的课程

Set<String> courSet =tables.columnKeySet();

for(String c:courSet){

System.out.print(c+"\t");

Map<String,Integer> scores =tables.column(c);

for(String s:stuSet){

System.out.print(scores.get(s)+"\t");

}

System.out.println();

}

System.out.println("===========转换===========");

Table<String,String,Integer> tables2 =Tables.transpose(tables);

//所有的行数据

Set<Cell<String,String,Integer>> cells2 =tables2.cellSet();

for(Cell<String,String,Integer> temp:cells2){

System.out.println(temp.getRowKey()+"-->"+temp.getColumnKey()+"-->"+temp.getValue());

}

}

}

# **第138集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季138集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com](http://www.bjsxt.xn--com-003b/)

# Apache Commons Collections

## 函数式编程。

package com.bjsxt.others.commons;

import java.util.ArrayList;

import java.util.List;

import org.apache.commons.collections4.Predicate;

import org.apache.commons.collections4.PredicateUtils;

import org.apache.commons.collections4.functors.EqualPredicate;

import org.apache.commons.collections4.functors.NotNullPredicate;

import org.apache.commons.collections4.functors.UniquePredicate;

import org.apache.commons.collections4.list.PredicatedList;

/\*\*

     函数式编程 之 Predicate 断言

  封装条件或判别式  if..else替代

  1、 new EqualPredicate<类型>(值)

EqualPredicate.equalPredicate(值);

  2、NotNullPredicate.INSTANCE

  3、UniquePredicate.uniquePredicate()

  4、自定义

     new Predicate() +evaluate

PredicateUtils.allPredicate  andPredicate anyPredicate

PredicatedXxx.predicatedXxx(容器,判断)

 \* @author Administrator

 \*

 \*/

public class Demo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

System.out.println("======自定义判断======");

//自定义的判别式

Predicate<String> selfPre =new Predicate<String>(){

@Override

public boolean evaluate(String object) {

return object.length()>=5 && object.length()<=20;

}};

Predicate notNull=NotNullPredicate.notNullPredicate();

Predicate all =PredicateUtils.allPredicate(notNull,selfPre);

List<String> list =PredicatedList.predicatedList(new ArrayList<String>(),all);

list.add("bjsxt");

list.add(null);

list.add("bj");

}

/\*\*

 \* 判断唯一

 \*/

public static void unique(){

System.out.println("====唯一性判断====");

Predicate<Long> uniquePre =UniquePredicate.uniquePredicate();

List<Long> list =PredicatedList.predicatedList(new ArrayList<Long>(), uniquePre);

list.add(100L);

list.add(200L);

list.add(100L); //出现重复值，抛出异常

}

/\*\*

 \* 判断非空

 \*/

public static void notNull(){

System.out.println("====非空判断====");

//Predicate notNull=NotNullPredicate.INSTANCE;

Predicate notNull=NotNullPredicate.notNullPredicate();

//String str ="bjs";

String str =null;

System.out.println(notNull.evaluate(str)); //如果非空为true ,否则为false

//添加容器值的判断

List<Long> list =PredicatedList.predicatedList(new ArrayList<Long>(), notNull);

list.add(1000L);

list.add(null); //验证失败，出现异常

}

/\*\*

 \* 比较相等判断

 \*/

public static void equal(){

System.out.println("======相等判断======");

//Predicate<String> pre =new EqualPredicate<String>("bjsxt");

Predicate<String> pre =EqualPredicate.equalPredicate("bjsxt");

boolean flag =pre.evaluate("bj");

System.out.println(flag);

}

}

## 函数式编程之2

package com.bjsxt.others.commons;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Iterator;

import java.util.List;

import org.apache.commons.collections4.CollectionUtils;

import org.apache.commons.collections4.Predicate;

import org.apache.commons.collections4.Transformer;

import org.apache.commons.collections4.functors.SwitchTransformer;

/\*\*

 解耦，业务处理与判断进行分类

 函数式编程 Transformer 类型转化

 1、new Transformer() +transform

 2、SwitchTransformer

CollectionUtils.collect(容器,转换器)

 \* @author Administrator

 \*

 \*/

public class Demo02 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

System.out.println("===自定义类型转换==");

//判别式

Predicate<Employee> isLow=new Predicate<Employee>(){

@Override

public boolean evaluate(Employee emp) {

return emp.getSalary()<10000;

}

};

Predicate<Employee> isHigh=new Predicate<Employee>(){

@Override

public boolean evaluate(Employee emp) {

return emp.getSalary()>=10000;

}

};

Predicate[] pres ={isLow,isHigh};

//转换

Transformer<Employee,Level> lowTrans =new Transformer<Employee,Level>(){

@Override

public Level transform(Employee input) {

return new Level(input.getName(),"卖身中");

}};

Transformer<Employee,Level> highTrans =new Transformer<Employee,Level>(){

@Override

public Level transform(Employee input) {

return new Level(input.getName(),"养身中");

}};

Transformer[] trans ={lowTrans,highTrans};

//二者进行了关联

Transformer switchTrans =new SwitchTransformer(pres, trans, null);

//容器

List<Employee> list =new ArrayList<Employee>();

list.add(new Employee("老马",1000000));

list.add(new Employee("老裴",999));

Collection<Level> levelList = CollectionUtils.collect(list,switchTrans);

//遍历容器

Iterator<Level> levelIt =levelList.iterator();

while(levelIt.hasNext()){

System.out.println(levelIt.next());

}

}

/\*\*

 \* 内置类型的转换

 \*/

public static void inner(){

System.out.println("===内置类型转换  长整形时间日期，转成指定格式的字符串==");

//类型转换器

Transformer<Long,String> trans =new Transformer<Long,String>(){

@Override

public String transform(Long input) {

return new SimpleDateFormat("yyyy年MM月dd日").format(input);

}};

//容器

List<Long> list =new ArrayList<Long>();

list.add(999999999999L);

list.add(300000000L);

//工具类 程序猿出钱---开发商---农民工出力

Collection<String>  result=CollectionUtils.collect(list, trans);

//遍历查看结果

for(String time:result){

System.out.println(time);

}

}

}

package com.bjsxt.others.commons;

/\*\*

 \* 员工类

 \* @author Administrator

 \*

 \*/

public class Employee {

private String name;

private double salary;

//alt +/

public Employee() {

}

//alt+shift+s  o

public Employee(String name, double salary) {

super();

this.name = name;

this.salary = salary;

}

//alt+shift+s  +r tab 回车 shift+tab 回车

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

@Override

public String toString() {

return "(码农:"+this.name+",敲砖钱:"+this.salary+")";

}

}

package com.bjsxt.others.commons;

/\*\*

 \* 等级类

 \* @author Administrator

 \*

 \*/

public class Level {

private String name;

private String level;

public Level() {

// TODO Auto-generated constructor stub

}

public Level(String name, String level) {

super();

this.name = name;

this.level = level;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getLevel() {

return level;

}

public void setLevel(String level) {

this.level = level;

}

@Override

public String toString() {

return "(码农:"+this.name+",水平:"+this.level+")";

}

}

# **第139集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季139集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com](http://www.bjsxt.xn--com-003b/)

# 函数式编程之闭包

package com.bjsxt.others.commons;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import org.apache.commons.collections4.Closure;

import org.apache.commons.collections4.CollectionUtils;

import org.apache.commons.collections4.Predicate;

import org.apache.commons.collections4.functors.ChainedClosure;

import org.apache.commons.collections4.functors.IfClosure;

import org.apache.commons.collections4.functors.WhileClosure;

/\*\*

 函数式编程 Closure 闭包 封装特定的业务功能

 1、Closure

 2、IfClosure  IfClosure.ifClosure(断言,功能1,功能2)

 3、WhileClosure WhileClosure.whileClosure(断言,功能,标识)

 4、ChainedClosure.chainedClosure(功能列表);

CollectionUtils.forAllDo(容器,功能类对象);

 \* @author Administrator

 \*

 \*/

public class Demo03 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//basic();

ifClosure();

//whileClosure();

chainClosure();

}

/\*\*

 \* 折上减   先打折商品，进行9折，满百再减20

 \*/

public static void chainClosure(){

List<Goods> goodsList =new ArrayList<Goods>();

goodsList.add(new Goods("javase视频",120,true));

goodsList.add(new Goods("javaee视频",100,false));

goodsList.add(new Goods("高新技术视频",80,false));

//满百减20

Closure<Goods> subtract=new Closure<Goods>(){

public void execute(Goods goods) {

if(goods.getPrice()>=100){

goods.setPrice(goods.getPrice()-20);

}

}};

//打折

Closure<Goods> discount=new Closure<Goods>(){

public void execute(Goods goods) {

if(goods.isDiscount()){

goods.setPrice(goods.getPrice()\*0.9);

}

}};

//链式操作

Closure<Goods> chainClo=ChainedClosure.chainedClosure(discount,subtract);

//关联

CollectionUtils.forAllDo(goodsList,chainClo);

//查看操作后的数据

for(Goods temp:goodsList){

System.out.println(temp);

}

}

/\*\*

 \* 确保所有的员工工资都大于10000,如果已经超过的不再上涨

 \*/

public static void whileClosure(){

//数据

List<Employee> empList =new ArrayList<Employee>();

empList.add(new Employee("bjsxt",20000));

empList.add(new Employee("is",10000));

empList.add(new Employee("good",5000));

//业务功能 每次上涨0.2

Closure<Employee> cols=new Closure<Employee>(){

public void execute(Employee emp) {

emp.setSalary(emp.getSalary()\*1.2);

}};

//判断

Predicate<Employee> empPre=new Predicate<Employee>(){

@Override

public boolean evaluate(Employee emp) {

return emp.getSalary()<10000;

}

};

//false 表示 while结构 先判断后执行   true do..while 先执行后判断

Closure<Employee> whileCols =WhileClosure.whileClosure(empPre, cols, false);

//工具类

CollectionUtils.forAllDo(empList, whileCols) ;

//操作后的数据

Iterator<Employee> empIt=empList.iterator();

while(empIt.hasNext()){

System.out.println(empIt.next());

}

}

/\*\*

 \* 二选一  如果是打折商品，进行9折，否则满百减20

 \*/

public static void ifClosure(){

List<Goods> goodsList =new ArrayList<Goods>();

goodsList.add(new Goods("javase视频",120,true));

goodsList.add(new Goods("javaee视频",100,false));

goodsList.add(new Goods("高新技术视频",80,false));

//满百减20

Closure<Goods> subtract=new Closure<Goods>(){

public void execute(Goods goods) {

if(goods.getPrice()>=100){

goods.setPrice(goods.getPrice()-20);

}

}};

//打折

Closure<Goods> discount=new Closure<Goods>(){

public void execute(Goods goods) {

if(goods.isDiscount()){

goods.setPrice(goods.getPrice()\*0.9);

}

}};

//判断

Predicate<Goods> pre=new Predicate<Goods>(){

public boolean evaluate(Goods goods) {

return goods.isDiscount();

}};

//二选一

Closure<Goods> ifClo=IfClosure.ifClosure(pre, discount,subtract);

//关联

CollectionUtils.forAllDo(goodsList,ifClo);

//查看操作后的数据

for(Goods temp:goodsList){

System.out.println(temp);

}

}

/\*\*

 \* 基本操作

 \*/

public static void basic(){

//数据

List<Employee> empList =new ArrayList<Employee>();

empList.add(new Employee("bjsxt",20000));

empList.add(new Employee("is",10000));

empList.add(new Employee("good",5000));

//业务功能

Closure<Employee> cols=new Closure<Employee>(){

public void execute(Employee emp) {

emp.setSalary(emp.getSalary()\*1.2);

}};

//工具类

CollectionUtils.forAllDo(empList, cols) ;

//操作后的数据

Iterator<Employee> empIt=empList.iterator();

while(empIt.hasNext()){

System.out.println(empIt.next());

}

}

}

package com.bjsxt.others.commons;

public class Goods {

private String name;

private double price;

//折扣

private boolean discount;

public Goods() {

// TODO Auto-generated constructor stub

}

public Goods(String name, double price, boolean discount) {

super();

this.name = name;

this.price = price;

this.discount = discount;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public boolean isDiscount() {

return discount;

}

public void setDiscount(boolean discount) {

this.discount = discount;

}

@Override

public String toString() {

return "(商品:"+this.name+",价格:"+this.price+",是否打折:"+(discount?"是":"否")+")";

}

}

# **第140集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季140集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【 专业JAVA培训机构,真正零首付入学[www.bjsxt.com](http://www.bjsxt.xn--com-003b/)

# Commons对集合进行的操作：

## 交集，并集，差集。

package com.bjsxt.others.commons;

import java.util.Collection;

import java.util.HashSet;

import java.util.Set;

import org.apache.commons.collections4.CollectionUtils;

/\*\*

 \* 集合操作

 \* 1、并集

 \* CollectionUtils.union();

 \* 2、交集

 \* CollectionUtils.intersection();

 \* CollectionUtils.retainAll();

 \* 3、差集

 \*  CollectionUtils.subtract();

 \* @author Administrator

 \*

 \*/

public class Demo04 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Set<Integer> set1 =new HashSet<Integer>();

set1.add(1);

set1.add(2);

set1.add(3);

Set<Integer> set2 =new HashSet<Integer>();

set2.add(2);

set2.add(3);

set2.add(4);

//并集

System.out.println("=========并集============");

Collection<Integer> col =CollectionUtils.union(set1,set2);

for(Integer temp:col){

System.out.println(temp);

}

//交集

System.out.println("=========交集============");

//col =CollectionUtils.intersection(set1, set2);

col =CollectionUtils.retainAll(set1, set2);

for(Integer temp:col){

System.out.println(temp);

}

//差集

System.out.println("=========差集============");

col =CollectionUtils.subtract(set1, set2);

for(Integer temp:col){

System.out.println(temp);

}

}

}

## Commons对队列的扩展：

package com.bjsxt.others.commons;

import java.util.Queue;

import org.apache.commons.collections4.Predicate;

import org.apache.commons.collections4.functors.NotNullPredicate;

import org.apache.commons.collections4.queue.CircularFifoQueue;

import org.apache.commons.collections4.queue.PredicatedQueue;

import org.apache.commons.collections4.queue.UnmodifiableQueue;

/\*\*

Queue队列

 1、循环队列:CircularFifoQueue

 2、只读队列:不可改变队列  UnmodifiableQueue

 3、断言队列:PredicatedQueue.predicatedQueue()

 \* @author Administrator

 \*

 \*/

public class Demo05 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//circular();

//readOnly();

predicate();

}

/\*\*

 \* 断言队列

 \*/

public static void predicate(){

//循环队列

CircularFifoQueue<String> que =new CircularFifoQueue<String>(2);

que.add("a");

que.add("b");

que.add("c");

Predicate notNull=NotNullPredicate.INSTANCE;

//包装成对应的队列

Queue<String> que2=PredicatedQueue.predicatedQueue(que, notNull);

que2.add(null);

}

/\*\*

 \* 只读队列

 \*/

public static void readOnly(){

//循环队列

CircularFifoQueue<String> que =new CircularFifoQueue<String>(2);

que.add("a");

que.add("b");

que.add("c");

Queue<String> readOnlyQue =UnmodifiableQueue.unmodifiableQueue(que);

readOnlyQue.add("d");

}

/\*\*

 \* 循环队列

 \*/

public static void circular(){

//循环队列

CircularFifoQueue<String> que =new CircularFifoQueue<String>(2);

que.add("a");

que.add("b");

que.add("c");

//查看

for(int i=0;i<que.size();i++){

System.out.println(que.get(i));

}

}

}

# **第141集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季141集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【 专业JAVA培训机构,真正零首付入学[www.bjsxt.com](http://www.bjsxt.xn--com-003b/)

# Commons对迭代器的扩展：

    package com.bjsxt.others.commons;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import org.apache.commons.collections4.BidiMap;

import org.apache.commons.collections4.IterableMap;

import org.apache.commons.collections4.MapIterator;

import org.apache.commons.collections4.Predicate;

import org.apache.commons.collections4.bidimap.DualHashBidiMap;

import org.apache.commons.collections4.iterators.ArrayListIterator;

import org.apache.commons.collections4.iterators.FilterIterator;

import org.apache.commons.collections4.iterators.LoopingIterator;

import org.apache.commons.collections4.iterators.UniqueFilterIterator;

import org.apache.commons.collections4.map.HashedMap;

/\*\*

 迭代器的扩展

 1、MapIterator 以后不再使用map.keySet.iterator访问

IterableMap  HashedMap

 2、UniqueFilterIterator 去重迭代器

 3、FilterIterator 自定义过滤 +Predicate

 4、LoopingIterator 循环迭代器

 5、ArrayListIterator 数组迭代器

 \* @author Administrator

 \*

 \*/

public class Demo06 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//mapIt();

//uniqueIt();

//filterIt();

//loopIt();

arrayIt();

}

/\*\*

 \* 数组迭代器

 \*/

public static void arrayIt(){

System.out.println("===== 数组迭代器  ====");

int[] arr ={1,2,3,4,5};

//数组迭代器

//Iterator<Integer> it =new ArrayListIterator<Integer>(arr);

//指定起始索引和结束索引

Iterator<Integer> it =new ArrayListIterator<Integer>(arr,1,3);

while(it.hasNext()){

System.out.println(it.next());

}

}

/\*\*

 \* 循环迭代器

 \*/

public static void loopIt(){

System.out.println("===== 循环迭代器  ====");

List<String> list =new ArrayList<String>();

list.add("refer");

list.add("dad");

list.add("bjsxt");

list.add("moom");

Iterator<String> it =new LoopingIterator(list);

for(int i=0;i<15;i++){

System.out.println(it.next());

}

}

/\*\*

 \* 自定义迭代器

 \*/

public static void filterIt(){

System.out.println("=====自定义迭代器  ====");

List<String> list =new ArrayList<String>();

list.add("refer");

list.add("dad");

list.add("bjsxt");

list.add("moom");

//自定义条件判断

Predicate<String> pre =new Predicate<String>(){

public boolean evaluate(String value) {

//回文判断

return new StringBuilder(value).reverse().toString().equals(value);

}};

//去除重复的过滤器

Iterator<String> it =new FilterIterator(list.iterator(),pre);

while(it.hasNext()){

System.out.println(it.next());

}

}

/\*\*

 \* 去重迭代器

 \*/

public static void uniqueIt(){

System.out.println("=====去重迭代器 ====");

List<String> list =new ArrayList<String>();

list.add("a");

list.add("b");

list.add("a");

//去除重复的过滤器

Iterator<String> it =new UniqueFilterIterator(list.iterator());

while(it.hasNext()){

System.out.println(it.next());

}

}

/\*\*

 \* map迭代器

 \*/

public static void mapIt(){

System.out.println("=====map迭代器====");

IterableMap<String,String> map =new HashedMap<String,String>();

map.put("a","bjsxt");

map.put("b", "sxt");

map.put("c", "good");

//使用 MapIterator

MapIterator<String,String> it =map.mapIterator();

while(it.hasNext()){

//一定要it.next()

/\*

it.next();

String key =it.getKey();

\*/

String key =it.next();

String value =it.getValue();

System.out.println(key+"-->"+value);

}

}

}

# Commons双向的Map

package com.bjsxt.others.commons;

import org.apache.commons.collections4.BidiMap;

import org.apache.commons.collections4.MapIterator;

import org.apache.commons.collections4.bidimap.DualHashBidiMap;

import org.apache.commons.collections4.bidimap.DualTreeBidiMap;

/\*\*

 双向Map 要求键与值都不能重复

BidiMap  inverseBidiMap()

 1、DualTreeBidiMap :有序

 2、DualHashBidiMap :无序

 \* @author Administrator

 \*

 \*/

public class Demo07 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//hashMap();

treeMap();

}

/\*\*

 \* 有序的双向Map

 \*/

public static void treeMap(){

System.out.println("=====有序的双向Map====");

BidiMap<String,String> map =new DualTreeBidiMap<String,String>();

map.put("bj", "bj@test.com");

map.put("sxt", "sxt@qq.com");

//遍历查看

MapIterator<String,String> it =map.inverseBidiMap().mapIterator();

while(it.hasNext()){

String key =it.next();

String value =it.getValue();

System.out.println(key+"-->"+value);

}

}

/\*\*

 \* 无序的双向Map

 \*/

public static void hashMap(){

System.out.println("=====无序的双向Map====");

BidiMap<String,String> map =new DualHashBidiMap<String,String>();

map.put("bj", "bj@test.com");

map.put("sxt", "sxt@qq.com");

//反转

System.out.println(map.inverseBidiMap().get("sxt@qq.com"));

//遍历查看

MapIterator<String,String> it =map.inverseBidiMap().mapIterator();

while(it.hasNext()){

String key =it.next();

String value =it.getValue();

System.out.println(key+"-->"+value);

}

}

}

# Bag包允许重复

package com.bjsxt.others.commons;

import java.util.Iterator;

import java.util.Set;

import org.apache.commons.collections4.Bag;

import org.apache.commons.collections4.bag.HashBag;

import org.apache.commons.collections4.bag.TreeBag;

/\*\*

Bag 包 允许重复

 1、HashBag 无序

 2、TreeBag 有序

 统计单词的出现次数

 \* @author Administrator

 \*

 \*/

public class Demo08 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//hashBag();

//treeBag();

String str ="this is a cat and that is a mice where is the food";

//分割字符串

String[] strArray =str.split(" ");

Bag<String> bag =new TreeBag<String>();

for(String temp:strArray){

bag.add(temp);

}

System.out.println("====统计次数===");

Set<String> keys =bag.uniqueSet();

for(String letter:keys){

System.out.println(letter+"-->"+bag.getCount(letter));

}

}

/\*\*

 \* 有序

 \*/

public static void treeBag(){

System.out.println("=====有序的包====");

Bag<String> bag =new TreeBag<String>();

bag.add("a");

bag.add("a",5);

bag.remove("a", 2);

bag.add("b");

bag.add("c");

Iterator<String> it =bag.iterator();

while(it.hasNext()){

System.out.println(it.next());

}

}

/\*\*

 \* 无序

 \*/

public static void hashBag(){

System.out.println("=====无序的包====");

Bag<String> bag =new HashBag<String>();

bag.add("a");

bag.add("a",5);

bag.remove("a", 2);

bag.add("b");

bag.add("c");

Iterator<String> it =bag.iterator();

while(it.hasNext()){

System.out.println(it.next());

}

}

}

# **第142集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

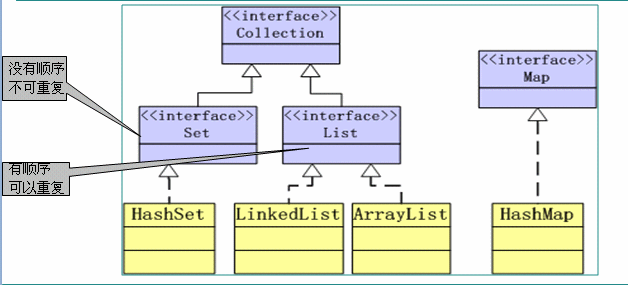
本文档配合：java300集第一季142集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【 专业JAVA培训机构,真正零首付入学[www.bjsxt.com](http://www.bjsxt.xn--com-003b/)

# 对集合这一章的总结：

    这章一共可以概括一下几个知识点：（1，3,6,9）

        1，一张图    

    3，三个知识点：

        1）迭代器

            1）Itarator接口

                boolean hasNext();

                   next();

                boolean remove();

            2)foreach:java.lang.Iterable+iterator();

        2）比较器:

            1)实体类可以排序 java.lang.Comparable+compareTo

            2)排序比较器（解耦，多种排序规则）Comparator+compare

                list+Collection.sort();

TreeSet

TreeMap

        3）泛型：<>

            泛型类，泛型方法，泛型接口，泛型的擦除，通配符？extends,super,泛型的嵌套

    6.六个接口

            Collection ,Set, List,Map,Iterator,Comparable

    9,九常用类：

        1）ArrayList

        2)LinkedList

        3)HashSet

        4)TreeMap

        5)hashMap

        6)Properties

        7)Hashtable

        8)Stack:栈

        9）Collections:工具类。

# **第143集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季143集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# IO入门之File类代码如下：

package com.bjsxt.io.file;

import java.io.File;

/\*\*

 \* 两个常量

 \* 1、路径分隔符  ;

 \* 2、名称分隔符 /(windows)  \(linux 等)

 \*

 \*

 \* @author Administrator

 \*

 \*/

public class Demo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

System.out.println(File.pathSeparator);

System.out.println(File.separator);

//路径表示形式

String path ="E:\\xp\\test\\2.jpg";

path="E:"+File.separator+"xp"+File.separator+"test"+File.separator+"2.jpg";

//推荐方式

path="E:/xp/test/2.jpg";

}

}

## File类进阶：

package com.bjsxt.io.file;

import java.io.File;

/\*\*

 \* 相对路径与绝对路径构造 File对象

 \* 1、相对路径

    File(String parent, String child)  ==>File("E:/xp/test","2.jpg")

    File(File parent, String child)     ==> File(new File("E:/xp/test"),"2.jpg")

    2、绝对路径

File(String name);

 \* @author Administrator

 \*

 \*/

public class Demo02 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

String parentPath ="E:/xp/test";

String name ="2.jpg";

//相对路径

File src =new File(parentPath,name);

src =new File(new File(parentPath),name);

//输出

System.out.println(src.getName());

System.out.println(src.getPath());

//绝对路径

src =new File("E:/xp/test/2.jpg");

System.out.println(src.getName());

System.out.println(src.getPath());

//没有盘符: 以 user.dir构建

src =new File("test.txt");

src =new File(".");

System.out.println(src.getName());

System.out.println(src.getPath());

System.out.println(src.getAbsolutePath());

}

}

# **第144集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季144集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# File类的常见方法代码如下：

package com.bjsxt.io.file;

import java.io.File;

import java.io.IOException;

/\*\*

 \* 常用方法:

1、文件名

getName() 文件名、路径名

getPath()路径名

getAbsoluteFile() 绝对路径所对应的File对象

getAbsolutePath() 绝对路径名

getParent() 父目录 ,相对路径的父目录，可能为null 如. 删除本身后的结果

2、判断信息

exists()

canWrite()

canRead()

isFile()

isDirectory()

isAbsolute()：消除平台差异，ie以盘符开头，其他以/开头

3、长度 字节数  不能读取文件夹的长度

length()

4、创建、删除

createNewFile() 不存在创建新文件,存在返回false

delete() 删除文件

static createTempFile(前缀3个字节长，后缀默认.temp) 默认临时空间

staticcreateTempFile(前缀3个字节长，后缀默认.temp,目录)

deleteOnExit() 退出虚拟机删除,常用于删除临时文件

 \* @author Administrator

 \*

 \*/

public class Demo03 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

test2();

try {

test3();

} catch (IOException e) {

e.printStackTrace();

System.out.println("文件操作失败");

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

//创建删除文件

public static void test3() throws IOException, InterruptedException{

//createNewFile() 不存在创建新文件

//String path="E:/xp/test/con"; //con系统关键字

String path="E:/xp/test/200.jpg";

//String path="E:/xp/test/1.jpg";

File src =new File(path);

if(!src.exists()){

boolean flag =src.createNewFile();

System.out.println(flag?"成功":"失败");

}

//删除文件

boolean flag =src.delete();

System.out.println(flag?"成功":"失败");

//static createTempFile(前缀3个字节长，后缀默认.temp) 默认临时空间

//static createTempFile(前缀3个字节长，后缀默认.temp,目录)

File temp= File.createTempFile("tes", ".temp",new File("e:/xp/test"));

Thread.sleep(10000);

temp.deleteOnExit(); //退出即删除

}

//2、判断信息

//3、长度 length()

public static void test2(){

//String path ="2.txt";

String path="E:/xp/test/200.jpg";

//String path="E:/xp/test/200.jpg";

File src =new File(path);

//是否存在

System.out.println("文件是否存在:"+src.exists());

//是否可读 写 canWrite() canRead()

System.out.println("文件是否可写"+src.canWrite());

System.out.println("============");

//isFile()

//isDirectory()

if(src.isFile()){

System.out.println("文件");

}else if(src.isDirectory()){

System.out.println("文件夹");

}else{

System.out.println("文件不存在");

}

System.out.println("是否为绝对路径"+src.isAbsolute());

System.out.println("长度为:"+src.length());

}

//1、名称

public static void test1(){

//File src =new File("E:/xp/test/2.jpg");

//建立联系

File src =new File("2.txt");

System.out.println(src.getName()); //返回名称

System.out.println(src.getPath()); //如果是绝对路径，返回完整路径，否则相对路径

System.out.println(src.getAbsolutePath());//返回绝对路径

System.out.println(src.getParent());//返回上一级目录，如果是相对，返回null

}

}

# **第145集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季145集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 对目录的操作：

package com.bjsxt.io.file;

import java.io.File;

import java.io.FilenameFilter;

/\*\*

 \* 5、操作目录

mkdir() 创建目录，必须确保 父目录存在，如果不存在，创建失败

mkdirs() 创建目录，如果父目录链不存在一同创建

list() 文件|目录 名字符串形式

listFiles()

static listRoots() 根路径

 \* @author Administrator

 \*

 \*/

public class Demo04 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

String path ="E:/xp/test/";

File src =new File(path); //文件夹

if(src.isDirectory()){ //存在并且为目录

System.out.println("======子目录|文件名===");

String[] subNames =src.list();

for(String temp:subNames){

System.out.println(temp);

}

System.out.println("=====子目录|文件File对象====");

File[] subFiles =src.listFiles();

for(File temp:subFiles){

System.out.println(temp.getAbsolutePath());

}

System.out.println("=====子文件.java对象====");

//命令设计模式

subFiles =src.listFiles(new FilenameFilter(){

@Override

/\*\*

 \* dir 代表src

 \*/

public boolean accept(File dir, String name) {

//System.out.println(dir.getAbsolutePath());

return  new File(dir,name).isFile()&&name.endsWith(".java");

}

});

for(File temp:subFiles){

System.out.println(temp.getAbsolutePath());

}

}

}

public static void test1(){

String path ="E:/xp/test/parent/p/test";

File src =new File(path);

//src.mkdir();

src.mkdirs();

}

}

## 文件夹子目录的遍历：

package com.bjsxt.io.file;

import java.io.File;

import java.util.Arrays;

/\*\*

 \* 输出子孙级目录|文件的名称(绝对路径)

 \* 1、listFiles()

 \* 2、递归

 \*

 \* static listRoots() 根路径

 \* @author Administrator

 \*

 \*/

public class Demo05 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

String path ="E:/xp/test";

File parent =new File(path);

//printName(parent);

File[] roots =File.listRoots();

System.out.println(Arrays.toString(roots));

for(File temp:roots){

//printName(temp);

}

}

/\*\*

 \* 输出路径

 \*/

public static void printName(File src){

if(null==src || !src.exists()){

return ;

}

System.out.println(src.getAbsolutePath());

if(src.isDirectory()){ //文件夹

for(File sub:src.listFiles()){

printName(sub);

}

}

}

}

# **第146集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季146集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# IO流的基本概念和原理：

一、概念  
流:流动 、流向 从一端移动到另一端  源头与目的地  
程序 与  文件|数组|网络连接|数据库  ，以程序为中心   
二、IO流分类  
1、流向: 输入流与输出流  
2、数据:字节流:二进制，可以一切文件 包括  纯文本 doc 音频、视频等等  
字符流:文本文件，只能处理纯文本  
3、功能:节点:包裹源头  
处理:增强功能，提供性能  
三、字符流与字节流 （重点） 与文件  
1、字节流  
    输入流:InputStream  read(byte[] b) 、read(byte[] b, int off, int len)  +close()   
FileInputStream()  
    输出流:OutputStream write(byte[] b)  write(byte[] b, int off, int len)  +flush() +close()  
FileOutputStream  
2、字符流  
   输入流:Reader read(char[] cbuf) read(char[] cbuf, int off, int len)  +close()   
FileReader()  
   输出流:Writer  write(char[] cbuf) write(char[] cbuf, int off, int len) +flush() +close()  
 write(String str, int off, int len)   
FileWriter()  
四、操作  
1、举例:搬家                 -->读取文件  
   1)、关联房子              --->建立与文件联系  
   2)、选择搬家           -->选择对应流  
   3)、搬家 -->读取|写出  
a)、卡车大小 --->数组大小  
b)、运输 -->读取、写出  
   4)、打发over -->释放资源  
2、操作  
1)建立联系  
2)选择流  
3)操作  数组大小+read 、write  
4)释放资源

# **第147集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季147集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 读取文件：

package com.bjsxt.io.byteIO;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStream;

/\*\*

 \* 文件的读取

 \* 1、建立联系   File对象

2、选择流     文件输入流  InputStream FileInputStream

3、操作  : byte[] car =new byte[1024];  +read+读取大小

              输出

4、释放资源 :关闭

 \* @author Administrator

 \*

 \*/

public class Demo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//1、建立联系   File对象

File src =new File("e:/xp/test/a.txt");

//2、选择流

InputStream is =null; //提升作用域

try {

is =new FileInputStream(src);

//3、操作 不断读取 缓冲数组

byte[] car =new byte[1024];

int len =0; //接收 实际读取大小

//循环读取

StringBuilder sb =new StringBuilder();

while(-1!=(len=is.read(car))){

//输出  字节数组转成字符串

String info =new String(car,0,len);

sb.append(info);

}

System.out.println(sb.toString());

} catch (FileNotFoundException e) {

e.printStackTrace();

System.out.println("文件不存在");

} catch (IOException e) {

e.printStackTrace();

System.out.println("读取文件失败");

}finally{

try {

//4、释放资源

if (null != is) {

is.close();

}

} catch (Exception e2) {

System.out.println("关闭文件输入流失败");

}

}

}

}

# 写出文件：

package com.bjsxt.io.byteIO;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.OutputStream;

/\*\*

 \* 写出文件

1、建立联系   File对象  目的地

2、选择流     文件输出流  OutputStream FileOutputStream

3、操作  :  write() +flush

4、释放资源 :关闭

 \* @author Administrator

 \*

 \*/

public class Demo02 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//1、建立联系   File对象  目的地

File dest =new File("e:/xp/test/test.txt");

//2、选择流   文件输出流  OutputStream FileOutputStream

OutputStream os =null;

//以追加形式 写出文件 必须为true 否则为覆盖

try {

os =new FileOutputStream(dest,true);

//3、操作

String str="bjsxt is very good \r\n";

//字符串转字节数组

byte[] data =str.getBytes();

os.write(data,0,data.length);

os.flush(); //强制刷新出去

} catch (FileNotFoundException e) {

e.printStackTrace();

System.out.println("文件未找到");

} catch (IOException e) {

e.printStackTrace();

System.out.println("文件写出失败");

}finally{

//4、释放资源 :关闭

try {

if (null != os) {

os.close();

}

} catch (Exception e2) {

System.out.println("关闭输出流失败");

}

}

}

}

# **第148集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季148集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 文件的拷贝：

package com.bjsxt.io.byteIO;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

/\*\*

 1、建立联系   File对象   源头 目的地

2、选择流

 文件输入流  InputStream FileInputStream

  文件输出流  OutputStream FileOutputStream

3、操作  :  拷贝

 byte[] flush =new byte[1024];

 int len =0;

  while(-1!=(len=输入流.read(flush))){

 输出流.write(flush,0,len)

  }

             输出流.flush

4、释放资源 :关闭 两个流

 \* @author Administrator

 \*

 \*/

public class CopyFileDemo {

/\*\*

 \* @param args

 \* @throws FileNotFoundException

 \*/

public static void main(String[] args) {

String src ="E:/xp/test";

String dest="e:/xp/test/4.jpg";

try {

copyFile(src,dest);

} catch (FileNotFoundException e) {

e.printStackTrace();

System.out.println("文件不存在");

} catch (IOException e) {

e.printStackTrace();

System.out.println("拷贝文件失败|关闭流失败");

}

}

/\*\*

 \* 文件的拷贝

 \* @param  源文件路径

 \* @param  目录文件路径

 \* @throws FileNotFoundException,IOException

 \* @return

 \*/

public static void copyFile(String srcPath,String destPath) throws FileNotFoundException,IOException {

//1、建立联系 源(存在且为文件) +目的地(文件可以不存在)

File src =new File(srcPath);

File dest =new File(destPath);

if(! src.isFile()){ //不是文件或者为null

System.out.println("只能拷贝文件");

throw new IOException("只能拷贝文件");

}

//2、选择流

InputStream is =new FileInputStream(src);

OutputStream os =new FileOutputStream(dest);

//3、文件拷贝   循环+读取+写出

byte[] flush =new byte[1024];

int len =0;

//读取

while(-1!=(len=is.read(flush))){

//写出

os.write(flush, 0, len);

}

os.flush(); //强制刷出

//关闭流

os.close();

is.close();

}

}

# 工具类的封装：

package com.bjsxt.io.byteIO;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

/\*\*

 \* 文件操作

 \* 1、文件拷贝

 \* 2、文件夹拷贝  拒绝自己拷贝给自己

 \* @author Administrator

 \*

 \*/

public class FileUtil {

/\*\*

 \* 拷贝文件夹

 \* @param src 源路径

 \* @param dest 目标路径

 \* @throws IOException

 \* @throws FileNotFoundException

 \*/

public static void copyDir(String  srcPath,String destPath) throws FileNotFoundException, IOException{

//拒绝自己拷贝给自己

if(srcPath.equals(destPath)){

return ;

}

File src=new File(srcPath);

File dest =new File(destPath);

copyDir(src,dest);

}

/\*\*

 \* 拷贝文件夹

 \* @param src 源File对象

 \* @param dest 目标File对象

 \* @throws IOException

 \* @throws FileNotFoundException

 \*/

public static void copyDir(File src,File dest) throws FileNotFoundException, IOException{

if(src.isDirectory()){ //文件夹

dest =new File(dest,src.getName());

if(dest.getAbsolutePath().contains(src.getAbsolutePath())){

System.out.println("父目录不能拷贝到子目录中");

return;

}

}

copyDirDetail(src,dest);

}

/\*\*

 \* 拷贝文件夹细节

 \* @param src

 \* @param dest

 \*/

public static void copyDirDetail(File src,File dest) throws FileNotFoundException,IOException{

if(src.isFile()){ //文件

try {

FileUtil.copyFile(src, dest);

} catch (FileNotFoundException e) {

//e.printStackTrace();

throw e;

} catch (IOException e) {

//e.printStackTrace();

throw e;

}

}else if(src.isDirectory()){ //文件夹

//确保目标文件夹存在

dest.mkdirs();

//获取下一级目录|文件

for(File sub:src.listFiles()){

copyDirDetail(sub,new File(dest,sub.getName()));

}

}

}

/\*\*

 \* 文件的拷贝

 \* @param  源文件路径

 \* @param  目录文件路径

 \* @throws FileNotFoundException,IOException

 \* @return

 \*/

public static void copyFile(String srcPath,String destPath) throws FileNotFoundException,IOException {

//1、建立联系 源(存在且为文件) +目的地(文件可以不存在)

copyFile(new File(srcPath),new File(destPath));

}

/\*\*

 \* 文件的拷贝

 \* @param  源文件File对象

 \* @param  目录文件File对象

 \* @throws FileNotFoundException,IOException

 \* @return

 \*/

public static void copyFile(File src,File dest) throws FileNotFoundException,IOException {

if(! src.isFile()){ //不是文件或者为null

System.out.println("只能拷贝文件");

throw new IOException("只能拷贝文件");

}

//dest为已经存在的文件夹，不能建立于文件夹同名的文件

if(dest.isDirectory()){

System.out.println(dest.getAbsolutePath()+"不能建立于文件夹同名的文件");

throw new IOException(dest.getAbsolutePath()+"不能建立于文件夹同名的文件");

}

//2、选择流

InputStream is =new BufferedInputStream(new FileInputStream(src));

OutputStream os =new BufferedOutputStream(new FileOutputStream(dest));

//3、文件拷贝   循环+读取+写出

byte[] flush =new byte[1024];

int len =0;

//读取

while(-1!=(len=is.read(flush))){

//写出

os.write(flush, 0, len);

}

os.flush(); //强制刷出

//关闭流

os.close();

is.close();

}

}

# **第149集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季149集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 文件的拷贝：

package com.bjsxt.io.byteIO;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

/\*\*

 \* 文件夹的拷贝

 \* 1、文件 赋值  copyFile

 \* 2、文件 创建 mkdirs()

 \* 3、递归查找子孙级

 \*

 \* @author Administrator

 \*

 \*/

public class CopyDir {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//源目录

String srcPath="E:/xp/test/a";

//目标目录

String destPath="E:/xp/test/a/b";

try {

FileUtil.copyDir(srcPath,destPath);

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

 \* 拷贝文件夹

 \* @param src 源路径

 \* @param dest 目标路径

 \*/

public static void copyDir(String  srcPath,String destPath){

File src=new File(srcPath);

File dest =new File(destPath);

copyDir(src,dest);

}

/\*\*

 \* 拷贝文件夹

 \* @param src 源File对象

 \* @param dest 目标File对象

 \*/

public static void copyDir(File src,File dest){

if(src.isDirectory()){ //文件夹

dest =new File(dest,src.getName());

}

copyDirDetail(src,dest);

}

/\*\*

 \* 拷贝文件夹细节

 \* @param src

 \* @param dest

 \*/

public static void copyDirDetail(File src,File dest){

if(src.isFile()){ //文件

try {

FileUtil.copyFile(src, dest);

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}else if(src.isDirectory()){ //文件夹

//确保目标文件夹存在

dest.mkdirs();

//获取下一级目录|文件

for(File sub:src.listFiles()){

copyDirDetail(sub,new File(dest,sub.getName()));

}

}

}

}

# **第150集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季150集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 参考150集视频，对以前的代码的健壮性进行了扩展即可了：

# **第151集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季151集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 字符流,读取纯文本。

package com.bjsxt.io.charIO;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.io.Reader;

/\*\*

 \* 纯文本读取

 \* @author Administrator

 \*

 \*/

public class Demo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//创建源

File src =new File("E:/xp/test/a.txt");

//选择流

Reader reader =null;

try {

reader =new FileReader(src);

//读取操作

char[] flush =new char[1024];

int len =0;

while(-1!=(len=reader.read(flush))){

//字符数组转成 字符串

String str =new String(flush,0,len);

System.out.println(str);

}

} catch (FileNotFoundException e) {

e.printStackTrace();

System.out.println("源文件不存在");

} catch (IOException e) {

e.printStackTrace();

System.out.println("文件读取失败");

}finally{

try {

if (null != reader) {

reader.close();

}

} catch (Exception e2) {

}

}

}

}

# 文件的写出：

package com.bjsxt.io.charIO;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileWriter;

import java.io.IOException;

import java.io.Writer;

/\*\*

 \* 写出文件

 \* @author Administrator

 \*

 \*/

public class Demo02 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//创建源

File dest =new File("e:/xp/test/char.txt");

//选择流

Writer wr =null;

try {

//追加文件，而不是覆盖文件

wr =new FileWriter(dest);

//写出

String msg ="追加.....锄禾日当午\r\n码农真辛苦\r\n一本小破书\r\n一读一上午";

wr.write(msg);

wr.append("倒萨发了看电视剧 ");

wr.flush();

} catch (FileNotFoundException e) {

e.printStackTrace();

}catch (IOException e) {

e.printStackTrace();

}finally{

try {

if (null != wr) {

wr.close();

}

} catch (Exception e2) {

}

}

}

}

# **第152集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季152集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# IO之处理流：

处理流:增强功能、提供性能，节点流之上  
一、缓冲流  
1)、字节缓冲流  
BufferedInputStream  
BufferedOutputStream  
2)、字符缓冲流  
BufferedReader   readLine()   
BufferedWriter    newLine()   
转换流: 字节流 转为字符流   处理乱码(编码集、解码集)  
1、编码与解码概念  
 编码：  字符   ---编码字符集>二进制  
 解码 : 二进制   --解码字符集->      字符  
2、乱码:  
1)编码与解码的字符集不统一  
2)字节缺少，长度丢失  
3、文件乱码   
InputStreamReader(字节输入流,"解码集")  
OutputStreamWriter(字符输出流,"编码集")

package com.bjsxt.io.buffered;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

/\*\*

 \* 字节流文件拷贝+缓冲流 ，提高性能

 \* 缓冲流(节点流)

 \* @author Administrator

 \*

 \*/

public class BufferedByteDemo {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

}

/\*\*

 \* 文件的拷贝

 \* @param  源文件路径

 \* @param  目录文件路径

 \* @throws FileNotFoundException,IOException

 \* @return

 \*/

public static void copyFile(String srcPath,String destPath) throws FileNotFoundException,IOException {

//1、建立联系 源(存在且为文件) +目的地(文件可以不存在)

File src =new File(srcPath);

File dest =new File(destPath);

if(! src.isFile()){ //不是文件或者为null

System.out.println("只能拷贝文件");

throw new IOException("只能拷贝文件");

}

//2、选择流

InputStream is =new BufferedInputStream(new FileInputStream(src));

OutputStream os =new BufferedOutputStream( new FileOutputStream(dest));

//3、文件拷贝   循环+读取+写出

byte[] flush =new byte[1024];

int len =0;

//读取

while(-1!=(len=is.read(flush))){

//写出

os.write(flush, 0, len);

}

os.flush(); //强制刷出

//关闭流

os.close();

is.close();

}

}

# 处理流字符流：

package com.bjsxt.io.buffered;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

/\*\*

 \* 字符缓冲流 +新增方法(不能发生多态)

 \* @author Administrator

 \*

 \*/

public class BufferedCharDemo {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//创建源 仅限于 字符的纯文本

File src =new File("E:/xp/test/Demo03.java");

File dest =new File("e:/xp/test/char.txt");

//选择流

BufferedReader reader =null;

BufferedWriter wr =null;

try {

reader =new BufferedReader(new FileReader(src));

wr =new BufferedWriter(new FileWriter(dest));

//读取操作

/\*

char[] flush =new char[1024];

int len =0;

while(-1!=(len=reader.read(flush))){

wr.write(flush, 0, len);

}\*/

//新增方法的操作

String line =null;

while(null!=(line=reader.readLine())){

wr.write(line);

//wr.append("\r\n");

wr.newLine(); //换行符号

}

wr.flush();//强制刷出

} catch (FileNotFoundException e) {

e.printStackTrace();

System.out.println("源文件不存在");

} catch (IOException e) {

e.printStackTrace();

System.out.println("文件读取失败");

}finally{

try {

if (null != wr) {

wr.close();

}

} catch (Exception e2) {

}

try {

if (null != reader) {

reader.close();

}

} catch (Exception e2) {

}

}

}

}

# **第153与154集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季153-154集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 处理流之转换流：

## 一、编码《-----》解码

二、转换流: 字节流 转为字符流   处理乱码(编码集、解码集)  
1、编码与解码概念  
 编码：  字符   ---编码字符集>二进制  
 解码 : 二进制   --解码字符集->      字符  
2、乱码:  
1)编码与解码的字符集不统一  
2)字节缺少，长度丢失  
3、文件乱码   
InputStreamReader(字节输入流,"解码集")  
OutputStreamWriter(字符输出流,"编码集")

package com.bjsxt.io.convert;

import java.io.UnsupportedEncodingException;

public class ConverDemo01 {

/\*\*

 \* @param args

 \* @throws UnsupportedEncodingException

 \*/

public static void main(String[] args) throws UnsupportedEncodingException {

String str ="中国";

byte[] data =str.getBytes();

//字节数不完整

System.out.println(new String(data,0,3));

}

/\*\*

 \* 编码与解码字符集必须相同，否则乱码

 \* @throws UnsupportedEncodingException

 \*/

public static void test1() throws UnsupportedEncodingException{

//解码 byte -->char

String str ="中国"; //gbk

//编码 char -->byte

byte[] data =str.getBytes();

//编码与解码字符集同一

System.out.println(new String(data));

data =str.getBytes("utf-8"); //设定编码字符集

//不同一出现乱码

System.out.println(new String(data));

//编码

byte[] data2 = "中国".getBytes("utf-8");

//解码

str=new String(data2,"utf-8");

System.out.println(str);

}

}

## 编码-》解码

package com.bjsxt.io.convert;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

/\*\*

 \* 转换流: 字节转为字符

 \* 1、输出流 OutputStreamWriter 编码

 \* 2、输入流 InputStreamReader  解码

 \*

 \* 确保源不能为乱码

 \* @author Administrator

 \*

 \*/

public class ConverDemo02 {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

//指定解码字符集

BufferedReader br =new BufferedReader(

new InputStreamReader(

new BufferedInputStream(

new FileInputStream(

new File("E:/xp/test/Demo03.java"))),"UTF-8")

);

//写出文件 编码

BufferedWriter bw =new BufferedWriter(

new OutputStreamWriter(

new BufferedOutputStream(

new FileOutputStream(new File("E:/xp/test/encode.java")))));

String info =null;

while(null!=(info=br.readLine())){

//System.out.println(info);

bw.write(info);

bw.newLine();

}

bw.flush();

bw.close();

br.close();

}

}

# **第155集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季155集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# IO图形化小结：

一、步骤： 创建源 选择流 操作(读取|写出) 释放  
二、流  
   
节点流:离数据源|程序最近的流 处理流:装饰模式 提高性能增强功能     
1、 字节流:可以处理一切(纯文本、音频、视频等)  
1)、输入流 InputStream FileInputStream  
ByteArrayInputStream  
操作:read(字节数组)  
 a)、中间容器  byte[] flush=new byte[长度]  
 b)、接收长度  int len =0;  
c)、循环读取 while(-1!=(len=流.read(flush))){}  
 d)、操作:输出、拷贝  
2)、输出流 OutputStream FileOutputStream  
ByteArrayOutputStream  
操作:write(字节数组,0,长度)  输出  
2、 字符流:只能处理纯文本  
1)、输入流:Reader FileReader  
操作:read(字符数组)  
a)、中间容器  char[] flush=new char[长度]  
b)、接收长度  int len =0;  
c)、循环读取 while(-1!=(len=流.read(flush))){}  
d)、操作:输出、拷贝  
2）、输出流:Writer FileWriter  
操作:write(字符数组,0,长度)  输出 1、 转换流:解码与编码字符集问题  
1)、输入流:InputStreamReader à解码  
2)、输出流:OutputStreamWriter—>编码  
2、缓冲流:提高性能  
   1)、输入流:BufferedInputStream BufferedReader  
   2)、输出流:BufferedOutputStream BufferedWriter  
3、处理数据+类型  
1)、基本+字符串：必须存在才能读取 读取与写出顺序一致  
   a)、输入流:DataInputStream readXxx  
   b)、输出流:DataOutputStream writeXxx  
2）、引用类型:Serializable transient  
   a)、反序列化:ObjectInputStream readObject  
   b)、序列化:ObjectOutputStream writeObject  
4、打印流: PrintStream   
5、System.in out err  setIn setOut  
以下流使用新增方法不能发生多态  
1、 ByteArrayOutputStream:  toByteArray()  
2、 BufferedReader: readLine()  
3、 BufferedWriter:newLine()  
4、 DataInputStream DataOutputStream  
5、 ObjectInputStream ObjectOutputStream  
6、 PrintStream

# **第156-157集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季156-157集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 其他流的补充：

## 字节流：

package com.bjsxt.io.others;

import java.io.BufferedInputStream;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.io.InputStream;

/\*\*

 \* 字节数组 节点流

 \* 数组的长度有限 ，数据量不会很大

 \*

 \* 文件内容不用太大

 \* 1、文件  --程序->字节数组

 \* 2、字节数组  --程序->文件

 \*

 \*

 \*

 \* @author Administrator

 \*

 \*/

public class ByteArrayDemo01 {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

read(write());

}

/\*\*

 \* 输出流  操作与文件输出流 有些不同， 有新增方法，不能使用多态

 \* @throws IOException

 \*/

public static byte[] write() throws IOException{

//目的地

byte[] dest;

//选择流   不同点

ByteArrayOutputStream bos =new ByteArrayOutputStream();

//操作 写出

String msg ="操作与 文件输入流操作一致";

byte[] info =msg.getBytes();

bos.write(info, 0, info.length);

//获取数据

dest =bos.toByteArray();

//释放资源

bos.close();

return dest;

}

/\*\*

 \* 输入流  操作与 文件输入流操作一致

 \* 读取字节数组

 \* @throws IOException

 \*/

public static void read(byte[] src) throws IOException{

//数据源传入

//选择流

InputStream is =new BufferedInputStream(

new ByteArrayInputStream(

src

)

);

//操作

byte[] flush =new byte[1024];

int len =0;

while(-1!=(len=is.read(flush))){

System.out.println(new String(flush,0,len));

}

//释放资源

is.close();

}

}

## 案例2;

package com.bjsxt.io.others;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

/\*\*

 \*1、文件  --程序->字节数组

 \*1)、文件输入流

 \*     字节数组输出流

 \*

 \*

 \* 2、字节数组  --程序->文件

 \* 1)、字节数组输入流

 \*      文件输出流

 \* @author Administrator

 \*

 \*/

public class ByteArrayDemo02 {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

byte[] data =getBytesFromFile("e:/xp/test/1.jpg");

toFileFromByteArray(data,"e:/xp/test/arr.jpg");

}

/\*\*

 \* 2、字节数组  --程序->文件

 \*/

public static void toFileFromByteArray(byte[] src,String destPath) throws IOException{

//创建源

//目的地

File dest=new File(destPath);

//选择流

//字节数组输入流

InputStream is =new BufferedInputStream(new ByteArrayInputStream(src));

//文件输出流

OutputStream os =new BufferedOutputStream(new FileOutputStream(dest));

//操作 不断读取字节数组

byte[] flush =new byte[1];

int len =0;

while(-1!=(len =is.read(flush))){

//写出到文件中

os.write(flush, 0, len);

}

os.flush();

//释放资源

os.close();

is.close();

}

/\*\*

 \* 1、文件  --程序->字节数组

 \* @return

 \* @throws IOException

 \*/

public static byte[] getBytesFromFile(String srcPath) throws IOException{

//创建文件源

File src =new File(srcPath);

//创建字节数组目的地

byte[] dest =null;

//选择流

//文件输入流

InputStream is =new BufferedInputStream(new FileInputStream(src));

//字节数组输出流 不能使用多态

ByteArrayOutputStream bos =new ByteArrayOutputStream();

//操作   不断读取文件 写出到字节数组流中

byte[] flush =new byte[1024];

int len =0;

while(-1!=(len =is.read(flush))){

//写出到字节数组流中

bos.write(flush, 0, len);

}

bos.flush();

//获取数据

dest =bos.toByteArray();

bos.close();

is.close();

return dest;

}

}

# **第158集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季158集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 其他流之处理流：

DataInputStream:

DataOutputStream:

package com.bjsxt.io.others;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

/\*\*

 \* 数据类型(基本+String)处理流

 \* 1、输入流 DataInputStream  readXxx()

 \* 2、输出流 DataOutputStream writeXxx()

 \* 新增方法不能使用多态

 \*

 \* java.io.EOFException :没有读取到相关的内容

 \* @author Administrator

 \*

 \*/

public class DataDemo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

try {

//write("e:/xp/test/data.txt");

//read("e:/xp/test/arr.txt"); //非法内容

read("e:/xp/test/data.txt");

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

/\*\*

 \* 从文件读取数据+类型

 \* @throws IOException

 \*/

public static void read(String destPath) throws IOException{

//创建源

File src =new File(destPath);

//选择流

DataInputStream dis =new DataInputStream(

new BufferedInputStream(

new FileInputStream(src)

)

);

//操作 读取的顺序与写出一致   必须存在才能读取

//不一致，数据存在问题

long num2 =dis.readLong();

double num1 =dis.readDouble();

String str =dis.readUTF();

dis.close();

System.out.println(num2+"-->"+str);

}

/\*\*

 \* 数据+类型输出到文件

 \* @throws IOException

 \*/

public static void write(String destPath) throws IOException{

double point =2.5;

long num=100L;

String str ="数据类型";

//创建源

File dest =new File(destPath);

//选择流  DataOutputStream

DataOutputStream dos =new DataOutputStream(

new BufferedOutputStream(

new FileOutputStream(dest)

)

);

//操作 写出的顺序 为读取准备

dos.writeDouble(point);

dos.writeLong(num);

dos.writeUTF(str);

dos.flush();

//释放资源

dos.close();

}

}

## 案例2：

package com.bjsxt.io.others;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

/\*\*

 \* 数据类型(基本+String)处理流

 \* 1、输入流 DataInputStream  readXxx()

 \* 2、输出流 DataOutputStream writeXxx()

 \* 新增方法不能使用多态

 \*

 \* java.io.EOFException :没有读取到相关的内容

 \* @author Administrator

 \*

 \*/

public class DataDemo02 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

try {

byte[] data=write();

read(data);

System.out.println(data.length);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

/\*\*

 \* 从字节数组读取数据+类型

 \* @throws IOException

 \*/

public static void read(byte[] src) throws IOException{

//选择流

DataInputStream dis =new DataInputStream(

new BufferedInputStream(

new ByteArrayInputStream(src)

)

);

//操作 读取的顺序与写出一致   必须存在才能读取

double num1 =dis.readDouble();

long num2 =dis.readLong();

String str =dis.readUTF();

dis.close();

System.out.println(num1+"-->"+num2+"-->"+str);

}

/\*\*

 \* 数据+类型输出到字节数组中

 \* @throws IOException

 \*/

public static byte[] write() throws IOException{

//目标数组

byte[] dest =null;

double point =2.5;

long num=100L;

String str ="数据类型";

//选择流 ByteArrayOutputStream  DataOutputStream

ByteArrayOutputStream bos =new ByteArrayOutputStream();

DataOutputStream dos =new DataOutputStream(

new BufferedOutputStream(

bos

)

);

//操作 写出的顺序 为读取准备

dos.writeDouble(point);

dos.writeLong(num);

dos.writeUTF(str);

dos.flush();

dest =bos.toByteArray();

//释放资源

dos.close();

return dest;

}

}

# **第159集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季159集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 处理流

处理流  
1、基本类型+String  保留数据+类型  
  输入流:DataInputStream    readXxx  
  输出流:DataOutputStream  writeXxx  
2、引用类型 (对象) 保留数据+类型  
   反序列化 输入流:ObjectInputStream  readObject()   
   序列化  输出流:ObjectOutputStream  writeObject()  
  注意:  
1)、先序列化后反序列化；反序列化顺序必须与序列化一致  
2)、不是所有的对象都可以序列化， java.io.Serializable  
      不是所有的属性都需要序列化，transient

package com.bjsxt.io.others;

/\*\*

 \* 空接口只是标识

 \* @author Administrator

 \*

 \*/

public class Employee implements java.io.Serializable {

//不需要序列化

private transient String name;

private double salary;

public Employee() {

}

public Employee(String name, double salary) {

super();

this.name = name;

this.salary = salary;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

}

案例：

package com.bjsxt.io.others;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.Arrays;

/\*\*

 \* 不是所有的對象都可以序列化  java.io.NotSerializableException

 \* 不是所有的屬性都需要序列化  transient

 \* @author Administrator

 \*

 \*/

public class ObjectDemo01 {

/\*\*

 \* @param args

 \* @throws ClassNotFoundException

 \*/

public static void main(String[] args) throws ClassNotFoundException {

try {

seri("e:/xp/test/ser.txt");

read("e:/xp/test/ser.txt");

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

//反序列化

public static void read(String destPath) throws IOException, ClassNotFoundException{

//创建源

File src =new File(destPath);

//选择流

ObjectInputStream dis =new ObjectInputStream(

new BufferedInputStream(

new FileInputStream(src)

)

);

//操作 读取的顺序与写出一致   必须存在才能读取

//不一致，数据存在问题

Object obj =dis.readObject();

if(obj instanceof Employee){

Employee emp=(Employee)obj;

System.out.println(emp.getName());

System.out.println(emp.getSalary());

}

obj =dis.readObject();

int[] arr=(int[])obj;

System.out.println(Arrays.toString(arr));

dis.close();

}

//序列化

public static void seri(String destPath) throws IOException{

Employee emp =new Employee("bjsxt",1000000);

int[] arr ={1,2,3,45};

//创建源

File dest =new File(destPath);

//选择流  ObjectOutputStream

ObjectOutputStream dos =new ObjectOutputStream(

new BufferedOutputStream(

new FileOutputStream(dest)

)

);

//操作 写出的顺序 为读取准备

dos.writeObject(emp);

dos.writeObject(arr);

//释放资源

dos.close();

}

}

# **第160集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季160集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 编写工具类实现关闭的方法：

package com.bjsxt.io.util;

import java.io.Closeable;

public class FileUtil {

/\*\*

 \* 工具类关闭流

 \* 可变参数: ...  只能形参最后一个位置,处理方式与数组一致

 \*/

public static void close(Closeable ... io){

for(Closeable temp:io){

try {

if (null != temp) {

temp.close();

}

} catch (Exception e) {

}

}

}

/\*\*

 \* 使用泛型方法

 \*/

public static <T extends Closeable> void closeAll(T ... io){

for(Closeable temp:io){

try {

if (null != temp) {

temp.close();

}

} catch (Exception e) {

}

}

}

}

## 测试;

package com.bjsxt.io.util;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.util.Properties;

/\*\*

 1、建立联系   File对象   源头 目的地

2、选择流

 文件输入流  InputStream FileInputStream

  文件输出流  OutputStream FileOutputStream

3、操作  :  拷贝

 byte[] flush =new byte[1024];

 int len =0;

  while(-1!=(len=输入流.read(flush))){

 输出流.write(flush,0,len)

  }

             输出流.flush

4、释放资源 :关闭 两个流

 \* @author Administrator

 \*

 \*/

public class CopyFileDemo {

/\*\*

 \* @param args

 \* @throws IOException

 \* @throws FileNotFoundException

 \*/

public static void main(String[] args) throws FileNotFoundException, IOException {

String src ="E:/xp/test";

String dest="e:/xp/test/4.jpg";

try {

copyFile(src,dest);

} catch (FileNotFoundException e) {

e.printStackTrace();

System.out.println("文件不存在");

} catch (IOException e) {

e.printStackTrace();

System.out.println("拷贝文件失败|关闭流失败");

}

}

/\*\*

 \* 文件的拷贝

 \* @param  源文件路径

 \* @param  目录文件路径

 \* @throws FileNotFoundException,IOException

 \* @return

 \*/

public static void copyFile(String srcPath,String destPath) throws FileNotFoundException,IOException {

//1、建立联系 源(存在且为文件) +目的地(文件可以不存在)

File src =new File(srcPath);

File dest =new File(destPath);

if(! src.isFile()){ //不是文件或者为null

System.out.println("只能拷贝文件");

throw new IOException("只能拷贝文件");

}

//2、选择流

InputStream is =new FileInputStream(src);

OutputStream os =new FileOutputStream(dest);

//3、文件拷贝   循环+读取+写出

byte[] flush =new byte[1024];

int len =0;

//读取

while(-1!=(len=is.read(flush))){

//写出

os.write(flush, 0, len);

}

os.flush(); //强制刷出

//关闭流

FileUtil.closeAll(os,is);

}

/\*\*

 \* 1.7新特性 try --with --resource

 \* @param srcPath

 \* @param destPath

 \* @throws FileNotFoundException

 \* @throws IOException

 \*/

public static void copyFile2(String srcPath,String destPath) throws FileNotFoundException,IOException {

//1、建立联系 源(存在且为文件) +目的地(文件可以不存在)

File src =new File(srcPath);

File dest =new File(destPath);

if(! src.isFile()){ //不是文件或者为null

System.out.println("只能拷贝文件");

throw new IOException("只能拷贝文件");

}

//2、选择流

try(

InputStream is =new FileInputStream(src);

OutputStream os =new FileOutputStream(dest);

){

//3、文件拷贝   循环+读取+写出

byte[] flush =new byte[1024];

int len =0;

//读取

while(-1!=(len=is.read(flush))){

//写出

os.write(flush, 0, len);

}

os.flush(); //强制刷出

}

}

}

# **第161集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季161集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 打印流：

package com.bjsxt.io.others;

import java.io.BufferedOutputStream;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.PrintStream;

/\*\*

 \* PrintStream 打印流 -->处理流

 \* @author Administrator

 \*

 \*/

public class PrintStreamDemo01 {

/\*\*

 \* @param args

 \* @throws FileNotFoundException

 \*/

public static void main(String[] args) throws FileNotFoundException {

System.out.println("test");

PrintStream ps =System.out;

ps.println(false);

//输出到文件

File src = new File("e:/xp/test/print.txt");

ps = new PrintStream(new BufferedOutputStream(new FileOutputStream(src)));

ps.println("io is so easy....");

ps.close();

}

}

System.in

System.out

System.err

package com.bjsxt.io.others;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.FileDescriptor;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.io.PrintStream;

import java.util.Scanner;

/\*\*

 \* 三个常量

 \* 1、System.in  输入流  键盘输入

 \* 2、System.out 输出流   控制台输出

 \*    System.err

 \*

 \* ==>重定向

 \* setIn()

 \* setOut()

 \* setErr()

 \* FileDescriptor.in

 \* FileDescriptor.out

 \* @author Administrator

 \*

 \*/

public class SystemDemo01 {

/\*\*

 \* @param args

 \* @throws FileNotFoundException

 \*/

public static void main(String[] args) throws FileNotFoundException {

//test1();

//test2();

//重定向

System.setOut(new PrintStream(new BufferedOutputStream(new FileOutputStream("e:/xp/test/print.txt")),true));

System.out.println("a");  //控制台  -->文件

System.out.println("test");

//回控制台

System.setOut(new PrintStream(new BufferedOutputStream(new FileOutputStream(FileDescriptor.out)),true));

System.out.println("back....");

}

public static void test2() throws FileNotFoundException{

InputStream is =System.in;  //键盘输入

is = new BufferedInputStream(new FileInputStream("e:/xp/test/print.txt"));

Scanner sc = new Scanner(is);

//System.out.println("请输入:");

System.out.println(sc.nextLine());

}

public static void test1(){

System.out.println("test");

System.err.println("err");

}

}

## buffer:

package com.bjsxt.io.others;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

/\*\*

 \* 封装输入

 \* @author Administrator

 \*

 \*/

public class BuffereIn {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

InputStream is =System.in;

BufferedReader br = new BufferedReader(new InputStreamReader(is));

System.out.println("请输入。。。。");

String msg =br.readLine();

System.out.println(msg);

}

}

# **第162集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季162集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# IO流之设计模式（装饰模式）：

    装饰模式：增强功能。

package com.bjsxt.io.pattern;

public class Voice {

private int voice =10;

public Voice() {

// TODO Auto-generated constructor stub

}

public int getVoice() {

return voice;

}

public void setVoice(int voice) {

this.voice = voice;

}

public void say(){

System.out.println(voice);

}

}

## ·类与类之间的关系：

     \* 类与类之间的关系

 \* 1、依赖:形参|局部变量  
 \* 2、关联:属性  
 \* 聚合:属性 整体与部分 不一致的生命周期 人与手  
 \*      组合:属性 整体与部分 一致的生命周期  人与大脑  
 \* 3、继承:父子类关系  
 \* 4、实现: 接口与实现类关系

package com.bjsxt.io.pattern;

/\*\*

 \* 扩音器

 \* 类与类之间的关系

 \* 1、依赖:形参|局部变量

 \* 2、关联:属性

 \*  聚合:属性 整体与部分 不一致的生命周期 人与手

 \*      组合:属性 整体与部分 一致的生命周期  人与大脑

 \* 3、继承:父子类关系

 \* 4、实现: 接口与实现类关系

 \*

 \*

 \*

 \*

 \* @author Administrator

 \*

 \*/

public class Amplifier {

private Voice voice;

public Amplifier() {

}

public Amplifier(Voice voice) {

super();

this.voice = voice;

}

public void say(){

System.out.println(voice.getVoice()\*1000);

}

}

## ·测试：

package com.bjsxt.io.pattern;

public class App {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Voice v =new Voice();

v.say();

Amplifier am =new Amplifier(v);

am.say();

}

}

# **第163集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季163集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# ruboCopy的应用：

## 1.环境的配置跟java的环境变量配置一样：

## 2.不要把父目录拷贝到子目录当中:

## 3.删除超长目录;

package com.bjsxt.io.byteIO;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

/\*\*

 \* 文件夹的拷贝

 \* 1、文件 赋值  copyFile

 \* 2、文件 创建 mkdirs()

 \* 3、递归查找子孙级

 \*

 \* @author Administrator

 \*

 \*/

public class CopyDir {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//源目录

String srcPath="E:/xp/test/a";

//目标目录

String destPath="E:/xp/test/a/b";

try {

FileUtil.copyDir(srcPath,destPath);

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

 \* 拷贝文件夹

 \* @param src 源路径

 \* @param dest 目标路径

 \*/

public static void copyDir(String  srcPath,String destPath){

File src=new File(srcPath);

File dest =new File(destPath);

copyDir(src,dest);

}

/\*\*

 \* 拷贝文件夹

 \* @param src 源File对象

 \* @param dest 目标File对象

 \*/

public static void copyDir(File src,File dest){

if(src.isDirectory()){ //文件夹

dest =new File(dest,src.getName());

}

copyDirDetail(src,dest);

}

/\*\*

 \* 拷贝文件夹细节

 \* @param src

 \* @param dest

 \*/

public static void copyDirDetail(File src,File dest){

if(src.isFile()){ //文件

try {

FileUtil.copyFile(src, dest);

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}else if(src.isDirectory()){ //文件夹

//确保目标文件夹存在

dest.mkdirs();

//获取下一级目录|文件

for(File sub:src.listFiles()){

copyDirDetail(sub,new File(dest,sub.getName()));

}

}

}

}

# **第164集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季164集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# Io之，RandomAccessFilel类：

package com.bjsxt.io.others;

import java.io.File;

import java.io.IOException;

import java.io.RandomAccessFile;

import com.bjsxt.io.util.FileUtil;

/\*\*

 \* 文件的分割思路

 \* 1、分割的块数 size   n块

 \* 2、每一块的大小 blockSize

 \*   最后:总的文件大小 -(n-1)\*blockSize

 \*

 \* @author Administrator

 \*

 \*/

public class RndDemo01 {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

RandomAccessFile rnd =new RandomAccessFile(new File("E:/xp/20130502/test/test.java"),"r");

rnd.seek(40);

//定义缓冲大小

byte[] flush =new byte[1024];

//接收长度

int len =0;

while(-1!=(len=rnd.read(flush))){

if(len>=20){

System.out.println(new String(flush,0,20));

break;

}else{

System.out.println(new String(flush,0,len));

}

}

FileUtil.close(rnd);

}

}

# **第165->167集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季165->167集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 文件的分割与合并：

package com.bjsxt.io.others;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.RandomAccessFile;

import java.io.SequenceInputStream;

import java.util.ArrayList;

import java.util.List;

import java.util.Vector;

import com.bjsxt.io.util.FileUtil;

public class SplitFile {

//文件的路径

private String filePath;

//文件名

private String fileName;

//文件大小

private long length;

//块数

private int size;

//每块的大小

private long blockSize;

//分割后的存放目录

private String destBlockPath;

//每块的名称

private List<String> blockPath;

public SplitFile(){

blockPath = new ArrayList<String>();

}

public SplitFile(String filePath,String destBlockPath){

this(filePath,destBlockPath,1024);

}

public SplitFile(String filePath,String destBlockPath,long blockSize){

this();

this.filePath= filePath;

this.destBlockPath =destBlockPath;

this.blockSize=blockSize;

init();

}

/\*\*

 \* 初始化操作 计算 块数、确定文件名

 \*/

public void init(){

File src =null;

//健壮性

if(null==filePath ||!(((src=new File(filePath)).exists()))){

return;

}

if(src.isDirectory()){

return ;

}

//文件名

this.fileName =src.getName();

//计算块数 实际大小 与每块大小

this.length = src.length();

//修正 每块大小

if(this.blockSize>length){

this.blockSize =length;

}

//确定块数

size= (int)(Math.ceil(length\*1.0/this.blockSize));

//确定文件的路径

initPathName();

}

private void initPathName(){

for(int i=0;i<size;i++){

this.blockPath.add(destBlockPath+"/"+this.fileName+".part"+i);

}

}

/\*\*

 \* 文件的分割

 \* 0)、第几块

 \* 1、起始位置

 \* 2、实际大小

 \* @param destPath 分割文件存放目录

 \*/

public void split(){

long beginPos =0;  //起始点

long actualBlockSize =blockSize; //实际大小

//计算所有块的大小、位置、索引

for(int i=0;i<size;i++){

if(i==size-1){ //最后一块

actualBlockSize =this.length-beginPos;

}

spiltDetail(i,beginPos,actualBlockSize);

beginPos+=actualBlockSize; //本次的终点，下一次的起点

}

}

/\*\*

 \* 文件的分割 输入 输出

 \* 文件拷贝

 \* @param idx 第几块

 \* @param beginPos 起始点

 \* @param actualBlockSize 实际大小

 \*/

private void spiltDetail(int idx,long beginPos,long actualBlockSize){

//1、创建源

File src = new File(this.filePath);  //源文件

File dest = new File(this.blockPath.get(idx)); //目标文件

//2、选择流

RandomAccessFile raf = null;  //输入流

BufferedOutputStream bos=null; //输出流

try {

raf=new RandomAccessFile(src,"r");

bos =new BufferedOutputStream(new FileOutputStream(dest));

//读取文件

raf.seek(beginPos);

//缓冲区

byte[] flush = new byte[1024];

//接收长度

int len =0;

while(-1!=(len=raf.read(flush))){

if(actualBlockSize-len>=0){ //查看是否足够

//写出

bos.write(flush, 0, len);

actualBlockSize-=len; //剩余量

}else{ //写出最后一次的剩余量

bos.write(flush, 0, (int)actualBlockSize);

break;

}

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}finally{

FileUtil.close(bos,raf);

}

}

/\*\*

 \* 文件的合并

 \*/

public void merge(String destPath){

//创建源

File dest =new File(destPath);

//选择流

BufferedOutputStream bos=null; //输出流

SequenceInputStream sis =null ;//输入流

//创建一个容器

Vector<InputStream> vi = new Vector<InputStream>();

try {

for (int i = 0; i < this.blockPath.size(); i++) {

vi.add(new BufferedInputStream(new FileInputStream(new File(this.blockPath.get(i)))));

}

bos =new BufferedOutputStream(new FileOutputStream(dest,true)); //追加

sis=new SequenceInputStream(vi.elements());

//缓冲区

byte[] flush = new byte[1024];

//接收长度

int len =0;

while(-1!=(len=sis.read(flush))){

bos.write(flush, 0, len);

}

bos.flush();

FileUtil.close(sis);

} catch (Exception e) {

}finally{

FileUtil.close(bos);

}

}

/\*\*

 \* 文件的合并

 \*/

public void merge1(String destPath){

//创建源

File dest =new File(destPath);

//选择流

BufferedOutputStream bos=null; //输出流

try {

bos =new BufferedOutputStream(new FileOutputStream(dest,true)); //追加

BufferedInputStream bis = null;

for (int i = 0; i < this.blockPath.size(); i++) {

bis = new BufferedInputStream(new FileInputStream(new File(this.blockPath.get(i))));

//缓冲区

byte[] flush = new byte[1024];

//接收长度

int len =0;

while(-1!=(len=bis.read(flush))){

bos.write(flush, 0, len);

}

bos.flush();

FileUtil.close(bis);

}

} catch (Exception e) {

}finally{

FileUtil.close(bos);

}

}

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

SplitFile split = new SplitFile("E:/xp/20130502/test/学员设置(20130502).xls","E:/xp/20130502",51);

//System.out.println(split.size);

//split.split();

split.merge("E:/xp/20130502/test1.xls");

}

}

# **第168集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季168集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# .IO总结：

 一、步骤： 创建源 选择流 操作(读取|写出) 释放

二、流  
   
节点流:离数据源|程序最近的流 处理流:装饰模式 提高性能增强功能     
1、 字节流:可以处理一切(纯文本、音频、视频等)  
1)、输入流 InputStream FileInputStream  
ByteArrayInputStream  
操作:read(字节数组)  
 a)、中间容器  byte[] flush=new byte[长度]  
 b)、接收长度  int len =0;  
c)、循环读取 while(-1!=(len=流.read(flush))){}  
 d)、操作:输出、拷贝  
2)、输出流 OutputStream FileOutputStream  
ByteArrayOutputStream  
操作:write(字节数组,0,长度)  输出  
2、 字符流:只能处理纯文本  
1)、输入流:Reader FileReader  
操作:read(字符数组)  
a)、中间容器  char[] flush=new char[长度]  
b)、接收长度  int len =0;  
c)、循环读取 while(-1!=(len=流.read(flush))){}  
d)、操作:输出、拷贝  
2）、输出流:Writer FileWriter  
操作:write(字符数组,0,长度)  输出 1、 转换流:解码与编码字符集问题  
1)、输入流:InputStreamReader à解码  
2)、输出流:OutputStreamWriter—>编码  
2、缓冲流:提高性能  
   1)、输入流:BufferedInputStream BufferedReader  
   2)、输出流:BufferedOutputStream BufferedWriter  
3、处理数据+类型  
1)、基本+字符串：必须存在才能读取 读取与写出顺序一致  
   a)、输入流:DataInputStream readXxx  
   b)、输出流:DataOutputStream writeXxx  
2）、引用类型:Serializable transient  
   a)、反序列化:ObjectInputStream readObject  
   b)、序列化:ObjectOutputStream writeObject  
4、打印流: PrintStream   
5、System.in out err  setIn setOut  
以下流使用新增方法不能发生多态  
1、 ByteArrayOutputStream:  toByteArray()  
2、 BufferedReader: readLine()  
3、 BufferedWriter:newLine()  
4、 DataInputStream DataOutputStream  
5、 ObjectInputStream ObjectOutputStream  
6、 PrintStream  
   
  
三、重点  
一、步骤： 创建源 选择流 操作(读取|写出) 释放  
二、流  
   
节点流:离数据源|程序最近的流 处理流:装饰模式 提高性能增强功能     
1、 字节流:可以处理一切(纯文本、音频、视频等)  
1)、输入流 InputStream FileInputStream  
ByteArrayInputStream  
操作:read(字节数组)  
 a)、中间容器  byte[] flush=new byte[长度]  
 b)、接收长度  int len =0;  
c)、循环读取 while(-1!=(len=流.read(flush))){}  
 d)、操作:输出、拷贝  
2)、输出流 OutputStream FileOutputStream  
ByteArrayOutputStream  
操作:write(字节数组,0,长度)  输出  
2、 字符流:只能处理纯文本  
1)、输入流:Reader FileReader  
操作:read(字符数组)  
a)、中间容器  char[] flush=new char[长度]  
b)、接收长度  int len =0;  
c)、循环读取 while(-1!=(len=流.read(flush))){}  
d)、操作:输出、拷贝  
2）、输出流:Writer FileWriter  
操作:write(字符数组,0,长度)  输出 1、 转换流:解码与编码字符集问题  
1)、输入流:InputStreamReader à解码  
2)、输出流:OutputStreamWriter—>编码  
2、缓冲流:提高性能  
   1)、输入流:BufferedInputStream BufferedReader  
   2)、输出流:BufferedOutputStream BufferedWriter  
3、处理数据+类型  
1)、基本+字符串：必须存在才能读取 读取与写出顺序一致  
   a)、输入流:DataInputStream readXxx  
   b)、输出流:DataOutputStream writeXxx  
2）、引用类型:Serializable transient  
   a)、反序列化:ObjectInputStream readObject  
   b)、序列化:ObjectOutputStream writeObject  
4、打印流: PrintStream   
5、System.in out err  setIn setOut  
以下流使用新增方法不能发生多态  
1、 ByteArrayOutputStream:  toByteArray()  
2、 BufferedReader: readLine()  
3、 BufferedWriter:newLine()  
4、 DataInputStream DataOutputStream  
5、 ObjectInputStream ObjectOutputStream  
6、 PrintStream  
   
  
三、重点  
   
  
四、操作  
0、打印文件|目录  
1、文件拷贝  
2、关闭流方法  
3、文件分割与合并(自学)

# **第169集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季169集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# Thread线程的入门：

    几个常见的概念：

        程序：指令集，静态的概念：

        进程：操作系统 调度程序， 动态的概念。

        线程：在进程内多条执行路径。

# **第170集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季170集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# java事件如何实现多线程：

    多线程的实现方式一。

package com.bjsxt.thread.create;

/\*\*

 \* 模拟龟兔赛跑

 1、创建多线程  继承  Thread  +重写run(线程体)

 2、使用线程: 创建子类对象 + 对象.start()  线程启动

 \*

 \* @author Administrator

 \*

 \*/

public class Rabbit extends Thread {

@Override

public void run() {

//线程体

for(int i=0;i<100;i++){

System.out.println("兔子跑了"+i+"步");

}

}

}

class Tortoise extends Thread {

@Override

public void run() {

//线程体

for(int i=0;i<100;i++){

System.out.println("乌龟跑了"+i+"步");

}

}

}

package com.bjsxt.thread.create;

public class RabbitApp {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//创建子类对象

Rabbit rab = new Rabbit();

Tortoise tor =new Tortoise();

//调用start 方法

//rab.start(); //不要调用run方法

rab.run();

//tor.start();

tor.run();

for(int i=0;i<1000;i++){

System.out.println("main==>"+i);

}

}

}

# **第171集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季171集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 静态代理模式：

package com.bjsxt.thread.create;

/\*\*

 \* 静态代理 设计模式

 \* 1、真实角色

 \* 2、代理角色： 持有真实角色的引用

 \* 3、二者 实现相同的接口

 \*

 \* @author Administrator

 \*

 \*/

public class StaticProxy {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//创建真实角色

Marry you =new You();

//创建代理角色 +真实角色的引用

WeddingCompany company =new WeddingCompany(you);

//执行任务

company.marry();

}

}

//接口

interface Marry{

public abstract void marry();

}

//真实角色

class You implements Marry{

@Override

public void marry() {

System.out.println("you and  嫦娥结婚了....");

}

}

//代理角色

class WeddingCompany implements Marry{

private Marry you;

public WeddingCompany() {

}

public WeddingCompany(Marry you) {

this.you = you;

}

private void before(){

System.out.println("布置猪窝....");

}

private void after(){

System.out.println("闹玉兔....");

}

@Override

public void marry() {

before();

you.marry();

after();

}

}

# **第172集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季172集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 使用Runnable创建线程：

一、继承Thread + run()  
启动: 创建子类对象 +对象.start()  
二、实现Runnable +run()  
启动:使用静态代理  
  1、创建真实角色  
  2、创建代理角色 Thread+引用  
  3、代理角色.start()  
  
推荐使用接口:  
1、避免单继承局限性  
2、便于共享资源

package com.bjsxt.thread.create;

/\*\*

 推荐  Runnable 创建线程

 1)、避免单继承的局限性

 2)、便于共享资源

  使用 Runnable 创建线程

  1、类 实现 Runnable接口 +重写 run()   -->真实角色类

  2、启动多线程  使用静态代理

    1)、创建真实角色

    2)、创建代理角色 +真实角色引用

    3)、调用 .start() 启动线程

 \* @author Administrator

 \*

 \*/

public class Programmer implements Runnable {

@Override

public void run() {

for(int i=0;i<1000;i++){

System.out.println("一边敲helloworld....");

}

}

}

## 测试：

package com.bjsxt.thread.create;

public class ProgrammerApp {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

 //1)、创建真实角色

Programmer pro =new Programmer();

  //2)、创建代理角色 +真实角色引用

Thread proxy =new Thread(pro);

  //3)、调用 .start() 启动线程

proxy.start();

for(int i=0;i<1000;i++){

System.out.println("一边聊qq....");

}

}

}

## web12306代码：

package com.bjsxt.thread.create;

/\*\*

 \* 方便共享资源

 \* @author Administrator

 \*

 \*/

public class Web12306 implements Runnable {

private int num =50;

@Override

public void run() {

while(true){

if(num<=0){

break; //跳出循环

}

System.out.println(Thread.currentThread().getName()+"抢到了"+num--);

}

}

public static void main(String[] args) {

//真实角色

Web12306 web = new Web12306();

//代理

Thread t1 =new Thread(web,"路人甲");

Thread t2 =new Thread(web,"黄牛已");

Thread t3 =new Thread(web,"攻城师");

//启动线程

t1.start();

t2.start();

t3.start();

}

}

# **第173集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季173集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 多线程的第三种实现方式（Callable）:

package com.bjsxt.thread.create;

import java.util.concurrent.Callable;

import java.util.concurrent.ExecutionException;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.Future;

/\*\*

 \* 使用Callable创建线程

 \* @author Administrator

 \*

 \*/

public class Call {

public static void main(String[] args) throws InterruptedException, ExecutionException {

//创建线程

ExecutorService  ser=Executors.newFixedThreadPool(2);

Race tortoise = new Race("老不死",1000);

Race rabbit = new Race("小兔子",500);

//获取值

Future<Integer> result1 =ser.submit(tortoise) ;

Future<Integer> result2 =ser.submit(rabbit) ;

Thread.sleep(2000); //2秒

tortoise.setFlag(false); //停止线程体循环

rabbit.setFlag(false);

int num1 =result1.get();

int num2 =result2.get();

System.out.println("乌龟跑了-->"+num1+"步");

System.out.println("小兔子跑了-->"+num2+"步");

//停止服务

ser.shutdownNow();

}

}

class Race implements Callable<Integer>{

private String name ; //名称

private long time; //延时时间

private boolean flag =true;

private int step =0; //步

public Race() {

}

public Race(String name) {

super();

this.name = name;

}

public Race(String name,long time) {

super();

this.name = name;

this.time =time;

}

@Override

public Integer call() throws Exception {

while(flag){

Thread.sleep(time); //延时

step++;

}

return step;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public long getTime() {

return time;

}

public void setTime(long time) {

this.time = time;

}

public boolean isFlag() {

return flag;

}

public void setFlag(boolean flag) {

this.flag = flag;

}

public int getStep() {

return step;

}

public void setStep(int step) {

this.step = step;

}

}

# **第174集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季174集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 线程的状态：

## 如何干涉线程的状态以及停止方法：

package com.bjsxt.thread.status;

public class StopDemo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Study s =new Study();

new Thread(s).start();

//外部干涉

for(int i=0;i<100;i++){

if(50==i){ //外部干涉

s.stop();

}

System.out.println("main.....-->"+i);

}

}

}

class Study implements Runnable{

 //1)、线程类中 定义 线程体使用的标识

private boolean flag =true;

@Override

public void run() {

//2)、线程体使用该标识

while(flag){

System.out.println("study thread....");

}

}

//3)、对外提供方法改变标识

public void stop(){

this.flag =false;

}

}

第175集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季175集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 线程的合并，阻塞：

    1、join :合并线程  
    2、yield:暂停自己的线程   static

package com.bjsxt.thread.status;

/\*\*

\* join:合并线程

\* @author Administrator

\*

\*/

public class JoinDemo01 extends Thread {

/\*\*

\* @param args

\* @throws InterruptedException

\*/

public static void main(String[] args) throws InterruptedException {

JoinDemo01 demo = new JoinDemo01();

Thread t = new Thread(demo); //新生

t.start();//就绪

//cpu调度 运行

for(int i=0;i<1000;i++){

if(50==i){

t.join(); //main阻塞...

}

System.out.println("main...."+i);

}

}

@Override

public void run() {

for(int i=0;i<1000;i++){

System.out.println("join...."+i);

}

}

}

## Yield方法（）：

package com.bjsxt.thread.status;

public class YieldDemo01 extends Thread {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

YieldDemo01 demo = new YieldDemo01();

Thread t = new Thread(demo); //新生

t.start();//就绪

//cpu调度 运行

for(int i=0;i<1000;i++){

if(i%20==0){

//暂停本线程 main

Thread.yield();

}

System.out.println("main...."+i);

}

}

@Override

public void run() {

for(int i=0;i<1000;i++){

System.out.println("yield...."+i);

}

}

}

# 第176集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季176集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 线程的Sleep（）；

package com.bjsxt.thread.status;

import java.text.SimpleDateFormat;

import java.util.Date;

/\*\*

 \* 倒计时

 \* 1、倒数10个数，一秒内打印一个

 \* 2、倒计时

 \* @author Administrator

 \*

 \*/

public class SleepDemo01 {

/\*\*

 \* @param args

 \* @throws InterruptedException

 \*/

public static void main(String[] args) throws InterruptedException {

Date endTime =new Date(System.currentTimeMillis()+10\*1000);

long end =endTime.getTime();

while(true){

//输出

System.out.println(new SimpleDateFormat("mm:ss").format(endTime));

//等待一秒

Thread.sleep(1000);

//构建下一秒时间

endTime =new Date(endTime.getTime()-1000);

//10秒以内 继续 否则 退出

if(end-10000>endTime.getTime()){

break;

}

}

}

public static void test1() throws InterruptedException{

int num =10;

while(true){

System.out.println(num--);

Thread.sleep(1000); //暂停

if(num<=0){

break;

}

}

}

}

## 案例2

package com.bjsxt.thread.status;

/\*\*

 \* Sleep模拟 网络延时  线程不安全的类

 \* @author Administrator

 \*

 \*/

public class SleepDemo02 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//真实角色

Web12306 web= new Web12306();

Web12306 web2 = new Web12306();

//代理

Thread t1 =new Thread(web,"路人甲");

Thread t2 =new Thread(web,"黄牛已");

Thread t3 =new Thread(web,"攻城师");

//启动线程

t1.start();

t2.start();

t3.start();

}

}

class Web12306 implements Runnable {

private int num =50;

@Override

public void run() {

while(true){

if(num<=0){

break; //跳出循环

}

try {

Thread.sleep(500);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(Thread.currentThread().getName()+"抢到了"+num--);

}

}

}

# 第177集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季177集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 线程的基本信息：

   1、sleep:休眠，不释放锁   
    1)、与时间相关:倒计时  
    2)、模拟网络延时

package com.bjsxt.thread.info;

public class MyThread implements Runnable {

private boolean flag =true;

private int num =0;

@Override

public void run() {

while(flag){

System.out.println(Thread.currentThread().getName()+"-->"+num++);

}

}

public void stop(){

this.flag=!this.flag;

}

}

## 测试：

package com.bjsxt.thread.info;

/\*\*

 \*

  Thread.currentThread()  :当前线程

  setName():设置名称

  getName():获取名称

  isAlive():判断状态

 \* @author Administrator

 \*

 \*/

public class InfoDemo01 {

/\*\*

 \* @param args

 \* @throws InterruptedException

 \*/

public static void main(String[] args) throws InterruptedException {

MyThread it =new MyThread();

Thread proxy =new Thread(it,"挨踢");

proxy.setName("test");

System.out.println(proxy.getName());

System.out.println(Thread.currentThread().getName()); //main

proxy.start();

System.out.println("启动后的状态:"+proxy.isAlive());

Thread.sleep(200);

it.stop();

Thread.sleep(100);

System.out.println("停止后的状态:"+proxy.isAlive());

}

}

## 线程的优级别：

    优先级代表的概率，不是绝对的先后顺序：

package com.bjsxt.thread.info;

/\*\*

 \* 优先级:概率，不是绝对的先后顺序

MAX\_PRIORITY  10

NORM\_PRIORITY 5 (默认)

MIN\_PRIORITY  1

   setPriority()

   getPriority()

 \* @author Administrator

 \*

 \*/

public class InfoDemo02 {

/\*\*

 \* @param args

 \* @throws InterruptedException

 \*/

public static void main(String[] args) throws InterruptedException {

MyThread it =new MyThread();

Thread p1 =new Thread(it,"挨踢1");

MyThread it2 =new MyThread();

Thread p2 =new Thread(it2,"挨踢2");

p1.setPriority(Thread.MIN\_PRIORITY); //设置优先级

p2.setPriority(Thread.MAX\_PRIORITY);//设置优先级

p1.start();

p2.start();

Thread.sleep(100);

it.stop();

it2.stop();

}

}

# **第178集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季178集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 线程的同步：

package com.bjsxt.thread.syn;

public class SynDemo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

//真实角色

Web12306 web= new Web12306();

//代理

Thread t1 =new Thread(web,"路人甲");

Thread t2 =new Thread(web,"黄牛已");

Thread t3 =new Thread(web,"攻城师");

//启动线程

t1.start();

t2.start();

t3.start();

}

}

/\*\*

 \* 线程安全的类

 \* @author Administrator

 \*

 \*/

class Web12306 implements Runnable {

private int num =10;

private boolean flag =true;

@Override

public void run() {

while(flag){

test5();

}

}

public void test6(){

if(num<=0){

flag=false; //跳出循环

return ;

}

 //a  b  c

synchronized(this){

try {

Thread.sleep(500); //模拟 延时

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(Thread.currentThread().getName()+"抢到了"+num--);

}

}

//线程不安全  锁定资源不正确

public void test5(){

//a  b  c

synchronized((Integer)num){

if(num<=0){

flag=false; //跳出循环

return ;

}

try {

Thread.sleep(500); //模拟 延时

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(Thread.currentThread().getName()+"抢到了"+num--);

}

}

//锁定范围不正确 线程不安全

public void test4(){

//   c  1

synchronized(this){

//b

if(num<=0){

flag=false; //跳出循环

return ;

}

}

// b

try {

Thread.sleep(500); //模拟 延时

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(Thread.currentThread().getName()+"抢到了"+num--);

}//a -->1

//线程安全  锁定正确

public void test3(){

//a  b  c

synchronized(this){

if(num<=0){

flag=false; //跳出循环

return ;

}

try {

Thread.sleep(500); //模拟 延时

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(Thread.currentThread().getName()+"抢到了"+num--);

}

}

//线程安全  锁定正确

public synchronized void test2(){

if(num<=0){

flag=false; //跳出循环

return ;

}

try {

Thread.sleep(500); //模拟 延时

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(Thread.currentThread().getName()+"抢到了"+num--);

}

//线程不安全

public void test1(){

if(num<=0){

flag=false; //跳出循环

return ;

}

try {

Thread.sleep(500);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(Thread.currentThread().getName()+"抢到了"+num--);

}

}

# **第179集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季179集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 单例的设计模式：

在外部使用的时候，确保一个类的只有一个对象：

## 线程的同步：

同步:并发 多个线程访问同一份资源  确保资源安全  -->线程安全  
synchronized -->同步  
  
一、同步块  
synchronized(引用类型|this|类.class){  
  
}  
二、同步方法  
synchronized   
  
三、死锁: 过多的同步容易造成死锁

package com.bjsxt.thread.syn;

/\*\*

 \* 单例设计模式:确保一个类只有一个对象

 \* @author Administrator

 \*

 \*/

public class SynDemo02 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

JvmThread thread1 = new JvmThread(100);

JvmThread thread2 = new JvmThread(500);

thread1.start();

thread2.start();

}

}

class JvmThread extends Thread{

private long time;

public JvmThread() {

}

public JvmThread(long time) {

this.time =time;

}

@Override

public void run() {

System.out.println(Thread.currentThread().getName()+"-->创建:"+Jvm.getInstance(time));

}

}

/\*\*

 \* 单例设计模式

 \* 确保一个类只有一个对象

 \* 懒汉式  double checking

 \* 1、构造器私有化,避免外部直接创建对象

 \* 2、声明一个私有的静态变量

 \* 3、创建一个对外的公共的静态方法 访问该变量，如果变量没有对象，创建该对象

 \*/

class Jvm {

//声明一个私有的静态变量

private static Jvm instance =null;

//构造器私有化,避免外部直接创建对象

private Jvm(){

}

//创建一个对外的公共的静态方法 访问该变量，如果变量没有对象，创建该对象

public static Jvm getInstance(long time){

// c d e  -->效率  提供 已经存在对象的访问效率

if(null==instance){

// a b

synchronized(Jvm.class){

if(null==instance ){

try {

Thread.sleep(time); //延时 ，放大错误

} catch (InterruptedException e) {

e.printStackTrace();

}

instance =new Jvm();

}

}

  }//a

  return instance;

}

public static Jvm getInstance3(long time){

//a b c d e  -->效率不高 c  存在对象也需要等待

synchronized(Jvm.class){

if(null==instance ){

try {

Thread.sleep(time); //延时 ，放大错误

} catch (InterruptedException e) {

e.printStackTrace();

}

instance =new Jvm();

}

return instance;

}

}

public static synchronized Jvm getInstance2(long time){

if(null==instance ){

try {

Thread.sleep(time); //延时 ，放大错误

} catch (InterruptedException e) {

e.printStackTrace();

}

instance =new Jvm();

}

return instance;

}

public static Jvm getInstance1(long time){

if(null==instance ){

try {

Thread.sleep(time); //延时 ，放大错误

} catch (InterruptedException e) {

e.printStackTrace();

}

instance =new Jvm();

}

return instance;

}

}

## 单例模式的两种情况：

package com.bjsxt.thread.syn;

/\*\*

 \* 单例创建的方式

 \* 1、懒汉式

 \* 1)、构造器私有化

 \* 2)、声明私有的静态属性

 \* 3)、对外提供访问属性的静态方法，确保该对象存在

 \*

 \* @author Administrator

 \*

 \*/

public class MyJvm {

private static MyJvm instance;

private MyJvm(){

}

public static MyJvm getInstance (){

if(null==instance){ //提供效率

synchronized(MyJvm.class){

if(null==instance){ //安全

instance =new MyJvm();

}

}

}

return instance;

}

}

/\*\*

 \* 饿汉式

   1)、构造器私有化

 \* 2)、声明私有的静态属性,同时创建该对象

 \* 3)、对外提供访问属性的静态方法

 \* @author Administrator

 \*

 \*/

class MyJvm2 {

private static MyJvm2 instance =new MyJvm2();

private MyJvm2(){

}

public static MyJvm2 getInstance (){

return instance;

}

}

/\*\*

 \* 类在使用的时候加载 ，延缓加载时间

 \* @author Administrator

 \*

 \*/

class MyJvm3 {

private static class JVMholder{

private static MyJvm3 instance =new MyJvm3();

}

private MyJvm3(){

}

public static MyJvm3 getInstance (){

return JVMholder.instance;

}

}

# **第180集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季180集

【 最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 一.线程死锁问题：

# 二、同步方法synchronized

# 三、死锁: 过多的同步容易造成死锁

package com.bjsxt.thread.syn;

/\*\*

 \* 过多的同步方法可能造成死锁

 \* @author Administrator

 \*

 \*/

public class SynDemo03 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Object g =new Object();

Object m = new Object();

Test t1 =new Test(g,m);

Test2 t2 = new Test2(g,m);

Thread proxy = new Thread(t1);

Thread proxy2 = new Thread(t2);

proxy.start();

proxy2.start();

}

}

class Test implements Runnable{

Object goods ;

Object money ;

public Test(Object goods, Object money) {

super();

this.goods = goods;

this.money = money;

}

@Override

public void run() {

while(true){

test();

}

}

public void test(){

synchronized(goods){

try {

Thread.sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

synchronized(money){

}

}

System.out.println("一手给钱");

}

}

class Test2  implements Runnable{

Object goods ;

Object money ;

public Test2(Object goods, Object money) {

super();

this.goods = goods;

this.money = money;

}

@Override

public void run() {

while(true){

test();

}

}

public void test(){

synchronized(money){

try {

Thread.sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

synchronized(goods){

}

}

System.out.println("一手给货");

}

}

# **第181集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季181集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 生产者与消费者模式：

    当多个线程共享一份资源的时候，会发生死锁的现象，我们一般是通过生产者与消费者模式进行解决。也有称为信号灯法。代码展示如下：

共享资料电影类：

package com.bjsxt.thread.pro;

/\*\*

 一个场景,共同的资源

  生产者消费者模式 信号灯法

 wait() :等待，释放锁   sleep 不释放锁

 notify()/notifyAll():唤醒

  与 synchronized

 \* @author Administrator

 \*

 \*/

public class Movie {

private String pic ;

//信号灯

//flag -->T 生产生产，消费者等待 ，生产完成后通知消费

//flag -->F 消费者消费 生产者等待, 消费完成后通知生产

private boolean flag =true;

/\*\*

 \* 播放

 \* @param pic

 \*/

public synchronized void play(String pic){

if(!flag){ //生产者等待

try {

this.wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

//开始生产

try {

Thread.sleep(500);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("生产了:"+pic);

//生产完毕

this.pic =pic;

//通知消费

this.notify();

//生产者停下

this.flag =false;

}

public synchronized void watch(){

if(flag){ //消费者等待

try {

this.wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

//开始消费

try {

Thread.sleep(200);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("消费了"+pic);

//消费完毕

//通知生产

this.notifyAll();

//消费停止

this.flag=true;

}

}

## 生产者：

package com.bjsxt.thread.pro;

/\*\*

 \* 生产者

 \* @author Administrator

 \*

 \*/

public class Player implements Runnable {

private Movie m ;

public Player(Movie m) {

super();

this.m = m;

}

@Override

public void run() {

for(int i=0;i<20;i++){

if(0==i%2){

m.play("左青龙");

}else{

m.play("右白虎");

}

}

}

}

## 消费者：

package com.bjsxt.thread.pro;

public class Watcher implements Runnable {

private Movie m ;

public Watcher(Movie m) {

super();

this.m = m;

}

@Override

public void run() {

for(int i=0;i<20;i++){

m.watch();

}

}

}

## 测试;

package com.bjsxt.thread.pro;

public class App {

public static void main(String[] args) {

//共同的资源

Movie m = new Movie();

//多线程

Player p = new Player(m);

Watcher w = new Watcher(m);

new Thread(p).start();

new Thread(w).start();

}

}

## 馒头案例的代码如下：

package com.bjsxt.thread.pro;

public class TestProduce {

public static void main(String[] args) {

SyncStack sStack = new SyncStack();

Shengchan sc = new Shengchan(sStack);

Xiaofei xf = new Xiaofei(sStack);

sc.start();

xf.start();

}

}

class Mantou {

int id;

Mantou(int id){

this.id=id;

}

}

class SyncStack{

int index=0;

Mantou[] ms = new Mantou[10];

public synchronized void push(Mantou m){

while(index==ms.length){

try {

this.wait();

//wait后，线程会将持有的锁释放。sleep是即使睡着也持有互斥锁。

} catch (InterruptedException e) {

e.printStackTrace();

}

}

this.notify(); //唤醒在当前对象等待池中等待的第一个线程。notifyAll叫醒所有在当前对象等待池中等待的所有线程。

//如果不唤醒的话。以后这两个线程都会进入等待线程，没有人唤醒。

ms[index]=m;

index++;

}

public synchronized Mantou pop(){

while(index==0){

try {

this.wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

this.notify();

index--;

return ms[index];

}

}

class Shengchan extends Thread{

SyncStack ss = null;

public Shengchan(SyncStack ss) {

this.ss=ss;

}

@Override

public void run() {

for (int i = 0; i < 20; i++) {

System.out.println("造馒头："+i);

Mantou m = new Mantou(i);

ss.push(m);

}

}

}

class Xiaofei extends Thread{

SyncStack ss = null;

public Xiaofei(SyncStack ss) {

this.ss=ss;

}

@Override

public void run() {

for (int i = 0; i < 20; i++) {

Mantou m = ss.pop();

System.out.println("吃馒头："+i);

}

}

}

# **第182集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季182集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 任务调度：

## Time定时器类

## TimeTask定时任务类：

package com.bjsxt.thread.schedule;

import java.util.Date;

import java.util.Timer;

import java.util.TimerTask;

/\*\*

    了解

Timer()

  schedule(TimerTask task, Date time)

  schedule(TimerTask task, Date firstTime, long period)

  自学 quartz

 \* @author Administrator

 \*

 \*/

public class TimeDemo01 {

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Timer timer =new Timer();

timer.schedule(new TimerTask(){

@Override

public void run() {

System.out.println("so easy....");

}}, new Date(System.currentTimeMillis()+1000), 200);

}

}

# **第183集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

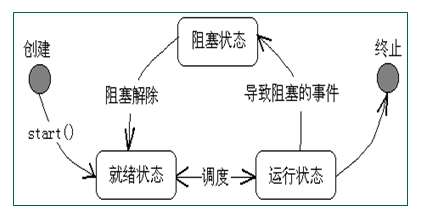
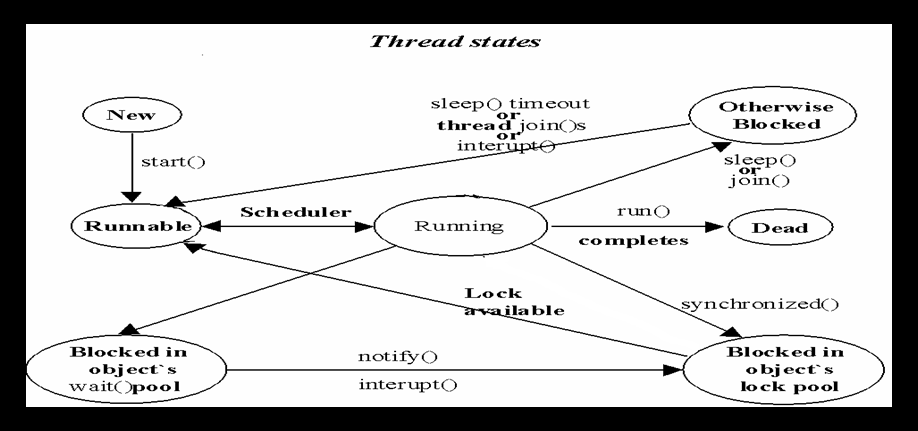
本文档配合：java300集第一季183集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 对线程的总结：

一、创建线程 重点  
1、继承 Thread  
2、实现 Runnable   
3、实现 Callable (了解)  
二、线程的状态

  
1、   
新生 -->start -->就绪 -->运行-->阻塞 -->终止  
2、终止线程 (重点)  
3、阻塞: join yield sleep  
   
三、线程的信息  
1、Thread.currentThread  
2、获取名称 设置名称  设置优先级  判断状态  
四、同步:对同一份资源  
synchronized(引用类型变量|this|类.class){  
}  
修饰符 synchronized 方法的签名{  
方法体  
}  
过多的同步可能造成死锁  
五、生产者消费者模式  
六、任务调度  
  
后期 : juc  quartz 自学。。。。

# **第184集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季184集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 网络编程：

    首先要明确一下基本的网络编程的概念：

一、网络:将不同区域的计算机连接到一起   局域网 城域网  互联网  
二、地址:IP地址 确定网络上 一个绝对地址 |位置  --->房子的地址  
三、端口号: 区分计算机软件的   -->房子的房门 2个字节    0-65535  共65536个  
1、在同一个协议下 端口号不能重复  不同协议下可以重复  
2、1024以下的不要使用  80-->http 21 -->ftp  
四、资源定位: URL 统一资源定位符   URI :统一资源  
五、数据的传输  
1、协议: Tcp 和UDP协议  
  1)、TCP(transfer control protocol): 电话 类似于三次握手   面向连接  安全可靠 效率低下  
  2)、UDP(UserDatagramProtocol ): 短信 非面向连接  效率高  
2、传输:  
  1)、先封装  
  2)、后拆封  
  
  
==>类  
1、InetAddress  InetSocketAddress  
2、URL  
3、TCP: ServerSocket  Socket  
4、UDP: DatagramSocket DatagramPacket

# **第185集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季185集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 网络编程值地址及端口：

1、InetAddress :封装 IP 及DNS  
 方法:  
getHostAddress() 返回ip地址  
getHostName() 返回域名|本机为计算机名  
InetAddress.getLocalHost();  
InetAddress.getByName("ip地址|域名");  
2、InetSocketAddress:封装端口  
1)、创建对象:   
 InetSocketAddress(String hostname, int port)   
InetSocketAddress(InetAddress addr, int port)   
2)、方法：  
getAddress()   
getHostName()   
getPort()

package com.bjsxt.net.ip;

import java.net.InetAddress;

import java.net.MalformedURLException;

import java.net.URL;

import java.net.UnknownHostException;

/\*\*

 \* 没有封装端口

 \* @author Administrator

 \*

 \*/

public class InetDemo01 {

/\*\*

 \* @param args

 \* @throws UnknownHostException

 \* @throws MalformedURLException

 \*/

public static void main(String[] args) throws UnknownHostException{

//使用getLocalHost方法创建InetAddress对象

InetAddress addr = InetAddress.getLocalHost();

System.out.println(addr.getHostAddress());  //返回：192.168.1.100

System.out.println(addr.getHostName());  //输出计算机名

//根据域名得到InetAddress对象

addr = InetAddress.getByName("www.163.com");

System.out.println(addr.getHostAddress());  //返回 163服务器的ip:61.135.253.15

System.out.println(addr.getHostName());  //输出：www.163.com

//根据ip得到InetAddress对象

addr = InetAddress.getByName("61.135.253.15");

System.out.println(addr.getHostAddress());  //返回 163服务器的ip:61.135.253.15

System.out.println(addr.getHostName());  //输出ip而不是域名。如果这个IP地 址不存在或DNS服务器不允许进行IP地址和域名的映射，getHostName方法就直接返回这个IP地址。

}

}

package com.bjsxt.net.ip;

import java.net.InetAddress;

import java.net.InetSocketAddress;

import java.net.UnknownHostException;

/\*\*

 \* 封装端口:在InetAddress基础上+端口

 \* @author Administrator

 \*

 \*/

public class InetSockeDemo01 {

/\*\*

 \* @param args

 \* @throws UnknownHostException

 \*/

public static void main(String[] args) throws UnknownHostException {

InetSocketAddress  address = new InetSocketAddress("127.0.0.1",9999);

address = new InetSocketAddress(InetAddress.getByName("127.0.0.1"),9999);

System.out.println(address.getHostName());

System.out.println(address.getPort());

InetAddress addr =address.getAddress();

System.out.println(addr.getHostAddress());  //返回：地址

System.out.println(addr.getHostName());  //输出计算机名

}

}

# **第186集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季186集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# URL：

URI(Uniform resource identifier)统一资源标识符,用来唯一的标识一个资源。

URL（Uniform Resource Locator）统一资源定位器,它是一种具体的URI

四部分组成: 协议  存放资源的主机域名  端口 资源文件名(/)

## URL：

### 一、创建

URL(String spec)  :绝对路径构建

URL(URL context, String spec)  :相对路径构建

### 二、方法

System.out.println("协议:"+url.getProtocol());

System.out.println("域名:"+url.getHost());

System.out.println("端口:"+url.getPort());

System.out.println("资源:"+url.getFile());

System.out.println("相对路径:"+url.getPath());

System.out.println("锚点:"+url.getRef()); //锚点

System.out.println("参数:"+url.getQuery());//?参数 :存在锚点  返回null ,不存在，返回正确

### 三、流

openStream()

package com.bjsxt.net.url;

import java.net.MalformedURLException;

import java.net.URL;

public class URLDemo01 {

/\*\*

 \* @param args

 \* @throws MalformedURLException

 \*/

public static void main(String[] args) throws MalformedURLException {

//绝对路径构建

URL url = new URL("http://www.baidu.com:80/index.html?uname=bjsxt");

System.out.println("协议:"+url.getProtocol());

System.out.println("域名:"+url.getHost());

System.out.println("端口:"+url.getPort());

System.out.println("资源:"+url.getFile());

System.out.println("相对路径:"+url.getPath());

System.out.println("锚点:"+url.getRef()); //锚点

System.out.println("参数:"+url.getQuery());//?参数 :存在锚点  返回null ,不存在，返回正确

url = new URL("http://www.baidu.com:80/a/");

url = new URL(url,"b.txt"); //相对路径

System.out.println(url.toString());

}

}

## 网络爬虫代码如下：

package com.bjsxt.net.url;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.net.URL;

/\*\*

 \* 获取资源：源代码

 \* @author Administrator

 \*

 \*/

public class URLDemo02 {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

URL url = new URL("http://www.baidu.com"); //主页 默认资源

//获取资源 网络流

/\*

InputStream is =url.openStream();

byte[] flush = new byte[1024];

int len =0;

while(-1!=(len=is.read(flush))){

System.out.println(new String(flush,0,len));

}

is.close();

\*/

BufferedReader  br =

new BufferedReader(new InputStreamReader(url.openStream(),"utf-8"));

BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("baidu2.html"),"utf-8"));

String msg =null;

while((msg=br.readLine())!=null){

//System.out.println(msg);

bw.append(msg);

bw.newLine();

}

bw.flush();

bw.close();

br.close();

}

}

# **第187集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季187集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# UDP通讯：

UDP：以数据为中心   非面向连接  不安全 数据可能丢失  效率高  
一、类 DatagramSocket DatagramPacket  
1、客户端：  
   1)、创建客户端   DatagramSocket  类  +指定端口  
   2)、准备数据  字节数组  
   3)、打包  DatagramPacket +服务器地址 及端口  
   4)、发送  
   5)、释放资源  
  
2、服务器端：  
   1)、创建 服务端  DatagramSocket  类  +指定端口  
   2)、准备接受容器 字节数组  封装 DatagramPacket   
   3)、包 接受数据  
   4)、分析  
   5)、释放资源

## Server端;

package com.bjsxt.net.udp;

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

/\*\*

 \* 服务端

 \* 1、创建服务端 +端口

 \* 2、准备接受容器

 \* 3、封装成 包

 \* 4、接受数据

 \* 5、分析数据

 \* 6、释放

 \* @author Administrator

 \*

 \*/

public class MyServer {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

//1、创建服务端 +端口

DatagramSocket server = new DatagramSocket(8888);

//2、准备接受容器

byte[] container = new byte[1024];

//3、封装成 包 DatagramPacket(byte[] buf, int length)

DatagramPacket packet =new DatagramPacket(container, container.length) ;

//4、接受数据

server.receive(packet);

//5、分析数据

byte[] data =packet.getData();

int len =packet.getLength();

System.out.println(new String(data,0,len));

//6、释放

server.close();

}

}

## Client端：

package com.bjsxt.net.udp;

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetSocketAddress;

/\*\*

 \* 客户端

 \* 1、创建客户端 +端口

 \* 2、准备数据

 \* 3、打包（发送的地点 及端口）

 \* 4、发送

 \* 5、释放

 \*

 \* 思考: 89.12  数据+类型

 \* @author Administrator

 \*

 \*/

public class MyClient {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

//1、创建客户端 +端口

DatagramSocket client = new DatagramSocket(6666);

//2、准备数据

String msg ="udp编程";

byte[] data =msg.getBytes();

//3、打包（发送的地点 及端口） DatagramPacket(byte[] buf, int length, InetAddress address, int port)

DatagramPacket packet = new DatagramPacket(data,data.length,new InetSocketAddress("localhost",8888));

//4、发送

client.send(packet);

//5、释放

client.close();

}

}

# **第188集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季188集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# UDP编程进阶：

## Server端：

package com.bjsxt.net.udp;

import java.io.ByteArrayInputStream;

import java.io.DataInputStream;

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

/\*\*

 \* 服务端

 \* 1、创建服务端 +端口

 \* 2、准备接受容器

 \* 3、封装成 包

 \* 4、接受数据

 \* 5、分析数据 字节数组-->double

 \* 6、释放

 \* @author Administrator

 \*

 \*/

public class Server {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

//1、创建服务端 +端口

DatagramSocket server = new DatagramSocket(8888);

//2、准备接受容器

byte[] container = new byte[1024];

//3、封装成 包 DatagramPacket(byte[] buf, int length)

DatagramPacket packet =new DatagramPacket(container, container.length) ;

//4、接受数据

server.receive(packet);

//5、分析数据

double data =convert(packet.getData());

System.out.println(data);

//6、释放

server.close();

}

/\*\*

 \* 字节数组 +Data 输入流

 \* @param data

 \* @return

 \* @throws IOException

 \*/

public static double convert(byte[] data) throws IOException{

DataInputStream dis =new DataInputStream(new ByteArrayInputStream(data));

double num =dis.readDouble();

dis.close();

return num;

}

}

## Client端：

package com.bjsxt.net.udp;

import java.io.ByteArrayOutputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetSocketAddress;

/\*\*

 \* 客户端

 \* 1、创建客户端 +端口

 \* 2、准备数据   double -->字节数组   字节数组输出流

 \* 3、打包（发送的地点 及端口）

 \* 4、发送

 \* 5、释放

 \*

 \*

 \* @author Administrator

 \*

 \*/

public class Client {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

//1、创建客户端 +端口

DatagramSocket client = new DatagramSocket(6666);

//2、准备数据

double num =89.12;

byte[] data =convert(num);

//3、打包（发送的地点 及端口） DatagramPacket(byte[] buf, int length, InetAddress address, int port)

DatagramPacket packet = new DatagramPacket(data,data.length,new InetSocketAddress("localhost",8888));

//4、发送

client.send(packet);

//5、释放

client.close();

}

/\*\*

 \* 字节数组 数据源  +Data 输出流

 \* @param num

 \* @return

 \* @throws IOException

 \*/

public static byte[] convert(double num) throws IOException{

byte[] data =null;

ByteArrayOutputStream bos = new ByteArrayOutputStream();

DataOutputStream dos =new DataOutputStream(bos);

dos.writeDouble(num);

dos.flush();

//获取数据

data = bos.toByteArray();

dos.close();

return data;

}

}

# **第189集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季189集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# TCP编程;

一, 基于tcp: 面向连接 安全 可靠 效率低 ，类似于打电话

一、面向连接:  请求-相应 Request --Response  
二、Socket编程  
1、服务器: ServerSocket  
2、客户端: Socket

## Socket编程：

package com.bjsxt.net.tcp基于tcp: 面向连接 安全 可靠 效率低 ，类似于打电话

一、面向连接:  请求-相应 Request --Response

二、Socket编程

1、服务器: ServerSocket

2、客户端: Socket

.socket;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

/\*\*

 必须先启动服务器 后连接

1、创建服务器  指定端口   ServerSocket(int port)

2、接收客户端连接

3、发送数据+接收数据

\*

\*/

public class Server {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

//1、创建服务器  指定端口   ServerSocket(int port)

ServerSocket server = new ServerSocket(8888);

//2、接收客户端连接   阻塞式

Socket socket =server.accept();

System.out.println("一个客户端建立连接");

//3、发送数据

String msg ="欢迎使用";

//输出流

/\*

BufferedWriter bw = new BufferedWriter(

new OutputStreamWriter(

socket.getOutputStream()));

bw.write(msg);

bw.newLine();

bw.flush();

\*/

DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

dos.writeUTF(msg);

dos.flush();

}

}

## Client:

package com.bjsxt.net.tcp.socket;

import java.io.DataInputStream;

import java.io.IOException;

import java.net.Socket;

import java.net.UnknownHostException;

/\*\*

 1、创建客户端   必须指定服务器+端口    此时就在连接

Socket(String host, int port)

 2、接收数据 +发送数据

 \* @author Administrator

 \*

 \*/

public class Client {

/\*\*

 \* @param args

 \* @throws IOException

 \* @throws UnknownHostException

 \*/

public static void main(String[] args) throws UnknownHostException, IOException {

//1、创建客户端   必须指定服务器+端口    此时就在连接

Socket client = new Socket("localhost",8888);

//2、接收数据

/\*

BufferedReader br = new BufferedReader(new InputStreamReader(client.getInputStream()));

String echo =br.readLine(); //阻塞式方法

System.out.println(echo);

\*/

DataInputStream dis = new DataInputStream(client.getInputStream());

String echo = dis.readUTF();

System.out.println(echo);

}

}

# **第190集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季190集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# Tcp编程之多客户端：

## 聊天室,客户端多线程原理：

package com.bjsxt.net.tcp.chat.demo01;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

/\*\*

 \* 创建服务器

 \* 写出数据：输出流

 \* 读取数据：输入流

 \* @author Administrator

 \*

 \*/

public class Server {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

ServerSocket server =new ServerSocket(9999);

Socket client =server.accept();

//写出数据

//输入流

DataInputStream dis = new DataInputStream(client.getInputStream());

String msg =dis.readUTF();

//输出流

DataOutputStream dos = new DataOutputStream(client.getOutputStream());

dos.writeUTF("服务器-->"+msg);

dos.flush();

}

}

## Client端：

package com.bjsxt.net.tcp.chat.demo01;

import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

import java.net.UnknownHostException;

/\*\*

 \* 创建客户端: 发送数据+接收数据

 \* 写出数据：输出流

 \* 读取数据：输入流

 \*

    输入流 与输出流 在同一个线程内 应该 独立处理，彼此独立

 \*

 \*/

public class Client {

/\*\*

 \* @param args

 \* @throws IOException

 \* @throws UnknownHostException

 \*/

public static void main(String[] args) throws UnknownHostException, IOException {

Socket client = new Socket("localhost",9999);

//控制台输入流

BufferedReader console = new BufferedReader(new InputStreamReader(System.in));

DataOutputStream dos = new DataOutputStream(client.getOutputStream());

DataInputStream dis = new DataInputStream(client.getInputStream());

while(true){

String info =console.readLine();

//输出流

dos.writeUTF(info);

dos.flush();

//输入流

String msg =dis.readUTF();

System.out.println(msg);

}

}

}

# **第191集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季191集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 聊天室，多客户端原理：

## Server端：

package com.bjsxt.net.tcp.chat.demo02;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

/\*\*

 \* 创建服务器

 \* 写出数据：输出流

 \* 读取数据：输入流

 \* @author Administrator

 \*

 \*/

public class Server {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

ServerSocket server =new ServerSocket(9999);

while(true){

Socket client =server.accept();

//写出数据

//输入流

DataInputStream dis = new DataInputStream(client.getInputStream());

DataOutputStream dos = new DataOutputStream(client.getOutputStream());

while(true){

String msg =dis.readUTF();

System.out.println(msg);

//输出流

dos.writeUTF("服务器-->"+msg);

dos.flush();

}

}

}

}

## Client端：

package com.bjsxt.net.tcp.chat.demo02;

import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

import java.net.UnknownHostException;

/\*\*

 \* 创建客户端: 发送数据+接收数据

 \* 写出数据：输出流

 \* 读取数据：输入流

 \*

    输入流 与输出流 在同一个线程内 应该 独立处理，彼此独立

 \*

 \*

 \*

 \* @author Administrator

 \*

 \*/

public class Client {

/\*\*

 \* @param args

 \* @throws IOException

 \* @throws UnknownHostException

 \*/

public static void main(String[] args) throws UnknownHostException, IOException {

Socket client = new Socket("localhost",9999);

new Thread(new Send(client)).start(); //一条路径

new Thread(new Receive(client)).start(); //一条路径

}

}

## 发送端：

package com.bjsxt.net.tcp.chat.demo02;

import java.io.BufferedReader;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

/\*\*

 \* 发送数据 线程

 \* @author Administrator

 \*

 \*/

public class Send implements Runnable{

//控制台输入流

private BufferedReader console;

//管道输出流

private DataOutputStream dos;

//控制线程

private boolean isRunning =true;

public Send() {

console =new BufferedReader(new InputStreamReader(System.in));

}

public Send(Socket client){

this();

try {

dos =new DataOutputStream(client.getOutputStream());

} catch (IOException e) {

//e.printStackTrace();

isRunning =false;

CloseUtil.closeAll(dos,console);

}

}

//1、从控制台接收数据

private String getMsgFromConsole(){

try {

return console.readLine();

} catch (IOException e) {

//e.printStackTrace();

}

return "";

}

/\*\*

 \* 1、从控制台接收数据

 \* 2、发送数据

 \*/

public void send(){

String msg = getMsgFromConsole();

try {

if(null!=msg&& !msg.equals("")){

dos.writeUTF(msg);

dos.flush(); //强制刷新

}

} catch (IOException e) {

//e.printStackTrace();

isRunning =false;

CloseUtil.closeAll(dos,console);

}

}

@Override

public void run() {

//线程体

while(isRunning){

send();

}

}

}

## 接受端：

package com.bjsxt.net.tcp.chat.demo02;

import java.io.DataInputStream;

import java.io.IOException;

import java.net.Socket;

/\*\*

 \* 接收线程

 \* @author Administrator

 \*

 \*/

public class Receive implements Runnable {

//输入流

private  DataInputStream dis ;

//线程标识

private boolean isRunning = true;

public Receive() {

}

public Receive(Socket client){

try {

dis = new DataInputStream(client.getInputStream());

} catch (IOException e) {

e.printStackTrace();

isRunning =false;

CloseUtil.closeAll(dis);

}

}

/\*\*

 \* 接收数据

 \* @return

 \*/

public String  receive(){

String msg ="";

try {

msg=dis.readUTF();

} catch (IOException e) {

e.printStackTrace();

isRunning =false;

CloseUtil.closeAll(dis);

}

return msg;

}

@Override

public void run() {

//线程体

while(isRunning){

System.out.println(receive());

}

}

}

## 关闭流工具类：

package com.bjsxt.net.tcp.chat.demo02;

import java.io.Closeable;

/\*\*

 \* 关闭流的方法

 \* @author Administrator

 \*

 \*/

public class CloseUtil {

public static void closeAll(Closeable... io){

for(Closeable temp:io){

try {

if (null != temp) {

temp.close();

}

} catch (Exception e) {

// TODO: handle exception

}

}

}

}

# **第192集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季192集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 聊天室，在上一个版本上增加对用户的代码如下：别的的代码在上一级直接拷贝即可：

## Server:

package com.bjsxt.net.tcp.chat.demo02;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

/\*\*

 \* 创建服务器

 \* 写出数据：输出流

 \* 读取数据：输入流

 \* @author Administrator

 \*

 \*/

public class Server {

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

ServerSocket server =new ServerSocket(9999);

while(true){

Socket client =server.accept();

//写出数据

//输入流

DataInputStream dis = new DataInputStream(client.getInputStream());

DataOutputStream dos = new DataOutputStream(client.getOutputStream());

while(true){

String msg =dis.readUTF();

System.out.println(msg);

//输出流

dos.writeUTF("服务器-->"+msg);

dos.flush();

}

}

}

}

## Client:

package com.bjsxt.net.tcp.chat.demo02;

import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

import java.net.UnknownHostException;

/\*\*

 \* 创建客户端: 发送数据+接收数据

 \* 写出数据：输出流

 \* 读取数据：输入流

 \*

    输入流 与输出流 在同一个线程内 应该 独立处理，彼此独立

 \*

 \*/

public class Client {

/\*\*

 \* @param args

 \* @throws IOException

 \* @throws UnknownHostException

 \*/

public static void main(String[] args) throws UnknownHostException, IOException {

Socket client = new Socket("localhost",9999);

new Thread(new Send(client)).start(); //一条路径

new Thread(new Receive(client)).start(); //一条路径

}

}

# **第193集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季193集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 聊天室，增加私聊功能：

## Server:

package com.bjsxt.net.tcp.chat.demo03;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

import java.util.List;

/\*\*

 \* 创建服务器

 \* 写出数据：输出流

 \* 读取数据：输入流

 \* @author Administrator

 \*

 \*/

public class Server {

private List<MyChannel> all = new ArrayList<MyChannel>();

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

new Server().start();

}

public void start() throws IOException{

ServerSocket server =new ServerSocket(9999);

while(true){

Socket client =server.accept();

MyChannel channel = new MyChannel(client);

all.add(channel);//统一管理

new Thread(channel).start(); //一条道路

}

}

/\*\*

 \* 一个客户端 一条道路

 \* 1、输入流

 \* 2、输出流

 \* 3、接收数据

 \* 4、发送数据

 \* @author Administrator

 \*

 \*/

private class MyChannel implements Runnable{

private DataInputStream dis ;

private DataOutputStream dos ;

private boolean isRunning =true;

public MyChannel(Socket client ) {

try {

dis = new DataInputStream(client.getInputStream());

dos = new DataOutputStream(client.getOutputStream());

} catch (IOException e) {

//e.printStackTrace();

CloseUtil.closeAll(dis,dos);

isRunning =false;

}

}

/\*\*

 \* 读取数据

 \* @return

 \*/

private String receive(){

String msg ="";

try {

msg=dis.readUTF();

} catch (IOException e) {

//e.printStackTrace();

CloseUtil.closeAll(dis);

isRunning =false;

all.remove(this); //移除自身

}

return msg;

}

/\*\*

 \* 发送数据

 \*/

private void send(String msg){

if(null==msg ||msg.equals("")){

return ;

}

try {

dos.writeUTF(msg);

dos.flush();

} catch (IOException e) {

//e.printStackTrace();

CloseUtil.closeAll(dos);

isRunning =false;

all.remove(this); //移除自身

}

}

/\*\*

 \* 发送给其他客户端

 \*/

private void sendOthers(){

String msg = this.receive();

//遍历容器

for(MyChannel other:all){

if(other ==this){

continue;

}

//发送其他客户端

other.send(msg);

}

}

@Override

public void run() {

while(isRunning){

sendOthers();

}

}

}

}

## Client:

package com.bjsxt.net.tcp.chat.demo03;

import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

import java.net.UnknownHostException;

/\*\*

 \* 创建客户端: 发送数据+接收数据

 \* 写出数据：输出流

 \* 读取数据：输入流

 \*

    输入流 与输出流 在同一个线程内 应该 独立处理，彼此独立

 \*/

public class Client {

/\*\*

 \* @param args

 \* @throws IOException

 \* @throws UnknownHostException

 \*/

public static void main(String[] args) throws UnknownHostException, IOException {

Socket client = new Socket("localhost",9999);

new Thread(new Send(client)).start(); //一条路径

new Thread(new Receive(client)).start(); //一条路径

}

}

## send:

package com.bjsxt.net.tcp.chat.demo03;

import java.io.BufferedReader;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

/\*\*

 \* 发送数据 线程

 \* @author Administrator

 \*

 \*/

public class Send implements Runnable{

//控制台输入流

private BufferedReader console;

//管道输出流

private DataOutputStream dos;

//控制线程

private boolean isRunning =true;

public Send() {

console =new BufferedReader(new InputStreamReader(System.in));

}

public Send(Socket client){

this();

try {

dos =new DataOutputStream(client.getOutputStream());

} catch (IOException e) {

//e.printStackTrace();

isRunning =false;

CloseUtil.closeAll(dos,console);

}

}

//1、从控制台接收数据

private String getMsgFromConsole(){

try {

return console.readLine();

} catch (IOException e) {

//e.printStackTrace();

}

return "";

}

/\*\*

 \* 1、从控制台接收数据

 \* 2、发送数据

 \*/

public void send(){

String msg = getMsgFromConsole();

try {

if(null!=msg&& !msg.equals("")){

dos.writeUTF(msg);

dos.flush(); //强制刷新

}

} catch (IOException e) {

//e.printStackTrace();

isRunning =false;

CloseUtil.closeAll(dos,console);

}

}

@Override

public void run() {

//线程体

while(isRunning){

send();

}

}

}

## Receive:

package com.bjsxt.net.tcp.chat.demo03;

import java.io.DataInputStream;

import java.io.IOException;

import java.net.Socket;

/\*\*

 \* 接收线程

 \* @author Administrator

 \*

 \*/

public class Receive implements Runnable {

//输入流

private  DataInputStream dis ;

//线程标识

private boolean isRunning = true;

public Receive() {

}

public Receive(Socket client){

try {

dis = new DataInputStream(client.getInputStream());

} catch (IOException e) {

e.printStackTrace();

isRunning =false;

CloseUtil.closeAll(dis);

}

}

/\*\*

 \* 接收数据

 \* @return

 \*/

public String  receive(){

String msg ="";

try {

msg=dis.readUTF();

} catch (IOException e) {

e.printStackTrace();

isRunning =false;

CloseUtil.closeAll(dis);

}

return msg;

}

@Override

public void run() {

//线程体

while(isRunning){

System.out.println(receive());

}

}

}

## 关闭流的基本工具：

package com.bjsxt.net.tcp.chat.demo03;

import java.io.Closeable;

/\*\*

 \* 关闭流的方法

 \* @author Administrator

 \*

 \*/

public class CloseUtil {

public static void closeAll(Closeable... io){

for(Closeable temp:io){

try {

if (null != temp) {

temp.close();

}

} catch (Exception e) {

// TODO: handle exception

}

}

}

}

# **第194集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季194集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 聊天室，私聊的实现代码如下：

基于tcp: 面向连接 安全 可靠 效率低 ，类似于打电话  
一、面向连接:  请求-相应 Request --Response  
二、Socket编程  
1、服务器: ServerSocket  
2、客户端: Socket

package com.bjsxt.net.tcp.chat.demo04;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

import java.util.List;

/\*\*

 \* 创建服务器

 \* 写出数据：输出流

 \* 读取数据：输入流

 \*/

public class Server {

private List<MyChannel> all = new ArrayList<MyChannel>();

/\*\*

 \* @param args

 \* @throws IOException

 \*/

public static void main(String[] args) throws IOException {

new Server().start();

}

public void start() throws IOException{

ServerSocket server =new ServerSocket(9999);

while(true){

Socket client =server.accept();

MyChannel channel = new MyChannel(client);

all.add(channel);//统一管理

new Thread(channel).start(); //一条道路

}

}

/\*\*

 \* 一个客户端 一条道路

 \* 1、输入流

 \* 2、输出流

 \* 3、接收数据

 \* 4、发送数据

 \*/

private class MyChannel implements Runnable{

private DataInputStream dis ;

private DataOutputStream dos ;

private boolean isRunning =true;

private String name;

public MyChannel(Socket client ) {

try {

dis = new DataInputStream(client.getInputStream());

dos = new DataOutputStream(client.getOutputStream());

this.name =dis.readUTF();

this.send("欢迎您进入聊天室");

sendOthers(this.name+"进入了聊天室",true);

} catch (IOException e) {

//e.printStackTrace();

CloseUtil.closeAll(dis,dos);

isRunning =false;

}

}

/\*\*

 \* 读取数据

 \* @return

 \*/

private String receive(){

String msg ="";

try {

msg=dis.readUTF();

} catch (IOException e) {

//e.printStackTrace();

CloseUtil.closeAll(dis);

isRunning =false;

all.remove(this); //移除自身

}

return msg;

}

/\*\*

 \* 发送数据

 \*/

private void send(String msg){

if(null==msg ||msg.equals("")){

return ;

}

try {

dos.writeUTF(msg);

dos.flush();

} catch (IOException e) {

//e.printStackTrace();

CloseUtil.closeAll(dos);

isRunning =false;

all.remove(this); //移除自身

}

}

/\*\*

 \* 发送给其他客户端

 \*/

private void sendOthers(String msg,boolean sys){

//是否为私聊 约定

if(msg.startsWith("@")&& msg.indexOf(":")>-1 ){ //私聊

//获取name

String name =msg.substring(1,msg.indexOf(":"));

String content = msg.substring(msg.indexOf(":")+1);

for(MyChannel other:all){

if(other.name.equals(name)){

other.send(this.name+"对您悄悄地说:"+content);

}

}

}else{

//遍历容器

for(MyChannel other:all){

if(other ==this){

continue;

}

if(sys){ //系统信息

other.send("系统信息:"+msg);

}else{

//发送其他客户端

other.send(this.name+"对所有人说:"+msg);

}

}

}

}

@Override

public void run() {

while(isRunning){

sendOthers(receive(),false);

}

}

}

}

## Client:

package com.bjsxt.net.tcp.chat.demo04;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

import java.net.UnknownHostException;

/\*\*

 \* 创建客户端: 发送数据+接收数据

 \* 写出数据：输出流

 \* 读取数据：输入流

 \*

    输入流 与输出流 在同一个线程内 应该 独立处理，彼此独立

    加入名称

 \*/

public class Client {

/\*\*

 \* @param args

 \* @throws IOException

 \* @throws UnknownHostException

 \*/

public static void main(String[] args) throws UnknownHostException, IOException {

System.out.println("请输入名称:");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String name = br.readLine();

if(name.equals("")){

return;

}

Socket client = new Socket("localhost",9999);

new Thread(new Send(client,name)).start(); //一条路径

new Thread(new Receive(client)).start(); //一条路径

}

}

## Send:

package com.bjsxt.net.tcp.chat.demo04;

import java.io.BufferedReader;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

/\*\*

 \* 发送数据 线程

 \* @author Administrator

 \*

 \*/

public class Send implements Runnable{

//控制台输入流

private BufferedReader console;

//管道输出流

private DataOutputStream dos;

//控制线程

private boolean isRunning =true;

//名称

private String name;

public Send() {

console =new BufferedReader(new InputStreamReader(System.in));

}

public Send(Socket client,String name){

this();

try {

dos =new DataOutputStream(client.getOutputStream());

this.name =name;

send(this.name);

} catch (IOException e) {

//e.printStackTrace();

isRunning =false;

CloseUtil.closeAll(dos,console);

}

}

//1、从控制台接收数据

private String getMsgFromConsole(){

try {

return console.readLine();

} catch (IOException e) {

//e.printStackTrace();

}

return "";

}

/\*\*

 \* 1、从控制台接收数据

 \* 2、发送数据

 \*/

public void send(String msg){

try {

if(null!=msg&& !msg.equals("")){

dos.writeUTF(msg);

dos.flush(); //强制刷新

}

} catch (IOException e) {

//e.printStackTrace();

isRunning =false;

CloseUtil.closeAll(dos,console);

}

}

@Override

public void run() {

//线程体

while(isRunning){

send(getMsgFromConsole());

}

}

}

## Receive:

package com.bjsxt.net.tcp.chat.demo04;

import java.io.DataInputStream;

import java.io.IOException;

import java.net.Socket;

/\*\*

 \* 接收线程

 \* @author Administrator

 \*

 \*/

public class Receive implements Runnable {

//输入流

private  DataInputStream dis ;

//线程标识

private boolean isRunning = true;

public Receive() {

}

public Receive(Socket client){

try {

dis = new DataInputStream(client.getInputStream());

} catch (IOException e) {

e.printStackTrace();

isRunning =false;

CloseUtil.closeAll(dis);

}

}

/\*\*

 \* 接收数据

 \* @return

 \*/

public String  receive(){

String msg ="";

try {

msg=dis.readUTF();

} catch (IOException e) {

e.printStackTrace();

isRunning =false;

CloseUtil.closeAll(dis);

}

return msg;

}

@Override

public void run() {

//线程体

while(isRunning){

System.out.println(receive());

}

}

}

## 关闭流工具类：

package com.bjsxt.net.tcp.chat.demo04;

import java.io.Closeable;

/\*\*

 \* 关闭流的方法

 \* @author Administrator

 \*

 \*/

public class CloseUtil {

public static void closeAll(Closeable... io){

for(Closeable temp:io){

try {

if (null != temp) {

temp.close();

}

} catch (Exception e) {

// TODO: handle exception

}

}

}

}

# **第195集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季195集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 手写服务器：

    用到的知识如下：

Socket编程

Html技术

XML解析

         反射

Server代码如下：

package com.bjsxt.server.demo1;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.ServerSocket;

import java.net.Socket;

/\*\*

 \* 创建服务器，并启动

 \* @author Administrator

 \*

 \*/

public class Server {

private ServerSocket server;

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Server server = new Server();

server.start();

}

/\*\*

 \* 启动方法

 \*/

public void start(){

try {

server = new ServerSocket(8888);

this.receive();

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

 \* 接收客户端

 \*/

private void receive(){

try {

Socket client =server.accept();

StringBuilder sb =new StringBuilder();

String msg =null;

BufferedReader br = new BufferedReader(new InputStreamReader(client.getInputStream()));

while((msg=br.readLine()).length()>0){

sb.append(msg);

sb.append("\r\n");

}

//接收客户端的请求信息

String requestInfo =sb.toString().trim();

System.out.println(requestInfo);

} catch (IOException e) {

//e.printStackTrace();

}

}

/\*\*

 \* 听着服务器

 \*/

public void stop(){

}

}

# **第196集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季196集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 手写服务器之html知识：

## Html超文本标记语言：

    基本的模板如下：

<html>

<head>

<title>第一个表单</title>

</head>

<body>

<pre>

method:请求方式 get/post

get: 默认方式,数据量小，安全性不高

post：量大，安全性相对高

action:请求的服务器路径

id:编号,前端区分唯一性 ,js中使用

name:名称，后端(服务器)区分唯一性 ，获取值

只要提交数据给后台，必须存在name

</pre>

<form method="post" action="http://localhost:8888/index.html">

用户名:<input type="text" name="uname" id="uname"/>

密码:<input type="password" name="pwd" id="pwd"/>

<input type="submit" value="登录"/>

</form>

</body>

</html>

[尝试一下](http://www.sxt.cn/searchsxt/tr?oper=l&&i=235)

### **清爽班Hmtl:**

<html>

<head>

<title>登录</title>

</head>

<body>

<form method="post" action="http://localhost:8888/log">

用户名:<input type="text" name="uname" id="uname"/>

密码:<input type="password" name="pwd" id="pwd"/>

<input type="submit" value="登录"/>

</form>

</body>

</html>

[尝试一下](http://www.sxt.cn/searchsxt/tr?oper=l&&i=236)

### Server端代码的如下;

package com.bjsxt.server.demo1;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

/\*\*

 \* 创建服务器，并启动

 \* @author Administrator

 \*

 \*/

public class Server2 {

private ServerSocket server;

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Server2 server = new Server2();

server.start();

}

/\*\*

 \* 启动方法

 \*/

public void start(){

try {

server = new ServerSocket(8888);

this.receive();

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

 \* 接收客户端

 \*/

private void receive(){

try {

Socket client =server.accept();

byte[] data=new byte[20480];

int len =client.getInputStream().read(data);

//接收客户端的请求信息

String requestInfo=new String(data,0,len).trim();

System.out.println(requestInfo);

} catch (IOException e) {

//e.printStackTrace();

}

}

/\*\*

 \* 听着服务器

 \*/

public void stop(){

}

}

# **第197集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季197集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 手写服务器之Http协议：

package com.bjsxt.server.demo1;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.Date;

/\*\*

 \* 创建服务器，并启动

 \*

 \* 1、请求

 \* 2、响应

 \* @author Administrator

 \*

 \*/

public class Server3 {

private ServerSocket server;

public static final String CRLF="\r\n";

public static final String BLANK=" ";

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Server3 server = new Server3();

server.start();

}

/\*\*

 \* 启动方法

 \*/

public void start(){

try {

server = new ServerSocket(8888);

this.receive();

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

 \* 接收客户端

 \*/

private void receive(){

try {

Socket client =server.accept();

byte[] data=new byte[20480];

int len =client.getInputStream().read(data);

//接收客户端的请求信息

String requestInfo=new String(data,0,len).trim();

System.out.println(requestInfo);

//响应

StringBuilder responseContext =new StringBuilder();

responseContext.append("<html><head><title>HTTP响应示例</title>" +

"</head><body>Hello bjsxt!</body></html>");

StringBuilder response =new StringBuilder();

//1)  HTTP协议版本、状态代码、描述

response.append("HTTP/1.1").append(BLANK).append("200").append(BLANK).append("OK").append(CRLF);

//2)  响应头(Response Head)

response.append("Server:bjsxt Server/0.0.1").append(CRLF);

response.append("Date:").append(new Date()).append(CRLF);

response.append("Content-type:text/html;charset=GBK").append(CRLF);

//正文长度 ：字节长度

response.append("Content-Length:").append(responseContext.toString().getBytes().length).append(CRLF);

//3)正文之前

response.append(CRLF);

//4)正文

response.append(responseContext);

System.out.println(responseContext);

//输出流

BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(client.getOutputStream()));

bw.write(response.toString());

bw.flush();

bw.close();

} catch (IOException e) {

}

}

/\*\*

 \* 听着服务器

 \*/

public void stop(){

}

}

# **第198集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季198集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 手写Tomcat服务器：

    Tttp协议的知识：

# **手写tomcat\_准备\_html协议\_抓包工具**

本节课主要讲解firebug工具。

# 抓包工具

抓包工具主要是用于分析http协议， 主要使用以下两种:

1、    firefox 为firebug

2、    ie 为httpwatch

 请自行下载相应工具，安装非常简单，自行安装。

当然随着浏览器升级，自带了一些开发者工具，也存在抓包功能也是非常好用的。

# firebug使用

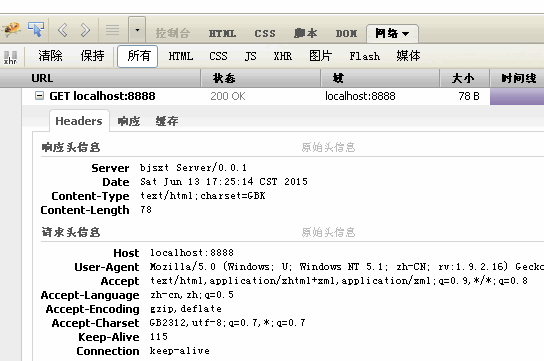
1、    启用fifebug

2、 访问服务器

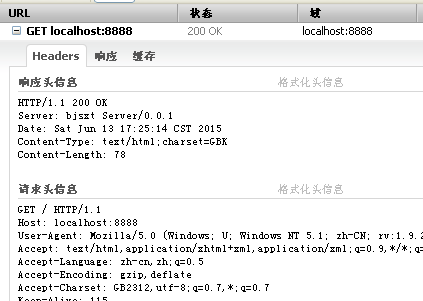
确保服务器启动时访问 ，http://localhost:8888

3、 分析http协议

点击GET localhost:8888可以看到协议信息



可以点击  原始头信息 看到原始内容



# 总结

·    使用抓包工具分析http协议

# **第199集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季199集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 对Request,Response的进一步封装：

## Response:

package com.bjsxt.server.demo1;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.OutputStream;

import java.io.OutputStreamWriter;

import java.net.Socket;

import java.util.Date;

import com.bjsxt.util.CloseUtil;

/\*\*

 \* 封装响应信息

 \* @author Administrator

 \*

 \*/

public class Response {

//两个常量

public static final String CRLF="\r\n";

public static final String BLANK=" ";

//流

private BufferedWriter bw ;

//正文

private StringBuilder content;

//存储头信息

private StringBuilder headInfo;

//存储正文长度

private int len =0;

public Response(){

headInfo =new StringBuilder();

content =new StringBuilder();

len =0;

}

public Response(Socket client){

this();

try {

bw= new BufferedWriter(new OutputStreamWriter(client.getOutputStream()));

} catch (IOException e) {

headInfo=null;

}

}

public Response(OutputStream os){

this();

bw= new BufferedWriter(new OutputStreamWriter(os));

}

/\*\*

 \* 构建正文

 \*/

public Response print(String info){

content.append(info);

len+=info.getBytes().length;

return this;

}

/\*\*

 \* 构建正文+回车

 \*/

public Response println(String info){

content.append(info).append(CRLF);

len+=(info+CRLF).getBytes().length;

return this;

}

/\*\*

 \* 构建响应头

 \*/

private void createHeadInfo(int code){

//1)  HTTP协议版本、状态代码、描述

headInfo.append("HTTP/1.1").append(BLANK).append(code).append(BLANK);

switch(code){

case 200:

headInfo.append("OK");

break;

case 404:

headInfo.append("NOT FOUND");

break;

case 505:

headInfo.append("SEVER ERROR");

break;

}

headInfo.append(CRLF);

//2)  响应头(Response Head)

headInfo.append("Server:bjsxt Server/0.0.1").append(CRLF);

headInfo.append("Date:").append(new Date()).append(CRLF);

headInfo.append("Content-type:text/html;charset=GBK").append(CRLF);

//正文长度 ：字节长度

headInfo.append("Content-Length:").append(len).append(CRLF);

headInfo.append(CRLF); //分隔符

}

//推送到客户端

void pushToClient(int code) throws IOException{

if(null==headInfo){

code =500;

}

createHeadInfo(code);

//头信息+分割符

bw.append(headInfo.toString());

//正文

bw.append(content.toString());

bw.flush();

}

public void close(){

CloseUtil.closeIO(bw);

}

}

## Server代码如下：

package com.bjsxt.server.demo1;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

/\*\*

 \* 创建服务器，并启动

 \*

 \* 1、请求

 \* 2、响应

 \* @author Administrator

 \*

 \*/

public class Server4 {

private ServerSocket server;

public static final String CRLF="\r\n";

public static final String BLANK=" ";

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Server4 server = new Server4();

server.start();

}

/\*\*

 \* 启动方法

 \*/

public void start(){

try {

server = new ServerSocket(8888);

this.receive();

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

 \* 接收客户端

 \*/

private void receive(){

try {

Socket client =server.accept();

byte[] data=new byte[20480];

int len =client.getInputStream().read(data);

//接收客户端的请求信息

String requestInfo=new String(data,0,len).trim();

System.out.println(requestInfo);

//响应

Response rep=new Response(client.getOutputStream());

rep.println("<html><head><title>HTTP响应示例</title>");

rep.println("</head><body>Hello server!</body></html>");

rep.pushToClient(200);

} catch (IOException e) {

}

}

/\*\*

 \* 停止服务器

 \*/

public void stop(){

}

}

# **第200集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季200集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 手写服务器之Request类的封装及Method方法的封装如下：

## Request类的代码如下：

package com.bjsxt.server.demo1;

import java.io.InputStream;

import java.io.UnsupportedEncodingException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.StringTokenizer;

/\*\*

 \* 封装request

 \* @author Administrator

 \*

 \*/

public class Request {

//请求方式

private String method;

//请求资源

private String url;

//请求参数

private Map<String,List<String>> parameterMapValues;

//内部

public static final String CRLF="\r\n";

private InputStream is;

private String requestInfo;

public Request(){

method ="";

url ="";

parameterMapValues=new HashMap<String,List<String>>();

requestInfo="";

}

public Request(InputStream is){

this();

this.is=is;

try {

byte[] data = new byte[20480];

int len = is.read(data);

requestInfo = new String(data, 0, len);

} catch (Exception e) {

return ;

}

//分析请求信息

parseRequestInfo();

}

/\*\*

 \* 分析请求信息

 \*/

private void parseRequestInfo(){

if(null==requestInfo ||(requestInfo=requestInfo.trim()).equals("")){

return ;

}

/\*\*

 \* =====================================

 \* 从信息的首行分解出 :请求方式    请求路径   请求参数(get可能存在)

 \*   如:GET /index.html?name=123&pwd=5456 HTTP/1.1

 \*

 \* 如果为post方式，请求参数可能在 最后正文中

 \*

 \* 思路:

 \* 1)请求方式 :找出第一个/  截取即可

 \* 2)请求资源:找出第一个/   HTTP/

 \* =====================================

 \*/

String paramString =""; //接收请求参数

//1、获取请求方式

String firstLine =requestInfo.substring(0,requestInfo.indexOf(CRLF));

int idx =requestInfo.indexOf("/"); // /的位置

this.method=firstLine.substring(0, idx).trim();

String urlStr =firstLine.substring(idx,firstLine.indexOf("HTTP/")).trim();

if(this.method.equalsIgnoreCase("post")){

this.url=urlStr;

paramString=requestInfo.substring(requestInfo.lastIndexOf(CRLF)).trim();

}else if(this.method.equalsIgnoreCase("get")){

if(urlStr.contains("?")){ //是否存在参数

String[] urlArray=urlStr.split("\\?");

this.url=urlArray[0];

paramString=urlArray[1];//接收请求参数

}else{

this.url=urlStr;

}

}

//不存在请求参数

if(paramString.equals("")){

return ;

}

//2、将请求参数封装到Map中

parseParams(paramString);

}

private void parseParams(String paramString){

//分割 将字符串转成数组

StringTokenizer token=new StringTokenizer(paramString,"&");

while(token.hasMoreTokens()){

String keyValue =token.nextToken();

String[] keyValues=keyValue.split("=");

if(keyValues.length==1){

keyValues =Arrays.copyOf(keyValues, 2);

keyValues[1] =null;

}

String key = keyValues[0].trim();

String value = null==keyValues[1]?null:decode(keyValues[1].trim(),"gbk");

//转换成Map 分拣

if(!parameterMapValues.containsKey(key)){

parameterMapValues.put(key,new ArrayList<String>());

}

List<String> values =parameterMapValues.get(key);

values.add(value);

}

}

/\*\*

 \* 解决中文

 \* @param value

 \* @param code

 \* @return

 \*/

private String decode(String value,String code){

try {

return java.net.URLDecoder.decode(value, code);

} catch (UnsupportedEncodingException e) {

//e.printStackTrace();

}

return null;

}

/\*\*

 \* 根据页面的name 获取对应的多个值

 \* @param args

 \*/

public String[] getParameterValues(String name){

List<String> values=null;

if((values=parameterMapValues.get(name))==null){

return null;

}else{

return values.toArray(new String[0]);

}

}

/\*\*

 \* 根据页面的name 获取对应的单个值

 \* @param args

 \*/

public String getParameter(String name){

String[] values =getParameterValues(name);

if(null==values){

return null;

}

return values[0];

}

public String getUrl() {

return url;

}

}

## Server代码：

package com.bjsxt.server.demo1;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

/\*\*

 \* 创建服务器，并启动

 \*

 \* 1、请求

 \* 2、响应

 \* @author Administrator

 \*

 \*/

public class Server5 {

private ServerSocket server;

public static final String CRLF="\r\n";

public static final String BLANK=" ";

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Server5 server = new Server5();

server.start();

}

/\*\*

 \* 启动方法

 \*/

public void start(){

try {

server = new ServerSocket(8888);

this.receive();

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

 \* 接收客户端

 \*/

private void receive(){

try {

Socket client =server.accept();

//请求

Request req=new Request(client.getInputStream());

//响应

Response rep=new Response(client.getOutputStream());

rep.println("<html><head><title>HTTP响应示例</title>");

rep.println("</head><body>");

rep.println("欢迎:").println(req.getParameter("uname")).println("回来");

rep.println("</body></html>");

rep.pushToClient(200);

} catch (IOException e) {

}

}

/\*\*

 \* 停止服务器

 \*/

public void stop(){

}

}

第201集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季201集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 对Request的进一步实现存储参数：

   request与Server相应的代码参考第200集即可。

手写tomcat\_封装Request\_方式及url

本节课主要讲解以面向对象封装Request。说白了就是将原有receive()方法中获取的请求的内容进行分析，分割字符串获取请求方式、请求资源、请求参数即可，封装成Request类。

# 目标

原有receive()方法中关于请求代码的片段进行封装，封装成Request类，同时获取请求方式、请求资源、请求参数

|  |
| --- |
| socket = server.accept();  //使用字节输入流  一次性读取请求内容  byte[] data =new byte[20480];  int len =socket.getInputStream().read(data);  String requestInfo=new String(data,0,len).trim();  System.out.println(requestInfo); |

 Request类为:

|  |
| --- |
| /\*\*   \* 目标:   \* 1、获取Request 协议   \* 2、获取请求方式   \* 3、获取请求资源   \* 4、获取请求参数   \* @author 北京尚学堂裴新   \* \*/  public class Request {  //两个常量  private static final String BLANK=" ";  private static final String CRLF="\r\n";  private Request () {  }  } |

# 获取Request协议

加入属性存储Request协议信息

|  |
| --- |
| private String requestInfo; |

使用构造器接收Socket的输入流，通过Socket的输入流即可获取Request协议，将receive()方法中的代码放入构造器即可，获取协议信息后，进行字符串的解析。

|  |
| --- |
| Request(InputStream is){  //使用字节输入流  一次性读取请求内容  byte[] data =new byte[20480];  int len=-1;  try {  len = is.read(data);  requestInfo=new String(data,0,len).trim();  } catch (IOException e) {  e.printStackTrace();  }  System.out.println(requestInfo);  //没有获取信息，不用进行下一步操作  if(null==requestInfo||requestInfo.length()<1){  return ;  }  //解析字符串          parseRequestInfo();  } |

加入解析字符串的方法

|  |
| --- |
| /\*\*   \* 解析字符串   \*/  private void parseRequestInfo(){  } |

从请求协议中

|  |  |
| --- | --- |
| GET请求方式 | POST请求方式 |
| GET /index.html? name=test&pwd=123456  HTTP/1.1  Accept: text/html, application/xhtml+xml, \* / \*  Accept-Language: zh-CN  User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)  Accept-Encoding: gzip, deflate  Host: localhost  Connection: Keep-Alive | POST /index.html  HTTP/1.1  Accept: text/html, application/xhtml+xml, \* / \*  Accept-Language: zh-CN  User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)  Accept-Encoding: gzip, deflate  Host: localhost  Connection: Keep-Alive    name=test&pwd=123456 |

发现请求方式、请求资源在首行，请求参数可能存在，如果存在必须区分，get请求时在首行，post请求时在请求实体中，因此必须加以区分。

加入三个属性，分别存储 请求方式、资源、参数。

|  |
| --- |
| //请求方式  private String method;  //请求资源  private String url;  //请求参数  private String queryStr; |

# 获取method

分解字符串首行，获取请求方式

·    1、首行如何获取?

·     截取字符串 ，从0位置 到HTTP/的位置即可

·    2、请求方式如何获取?

·     截取字符串 ，从0位置到/的位置即可

在 parseRequestInfo()中加入以下代码即可:

|  |
| --- |
| //首行  String firstLine =requestInfo.substring(0,requestInfo.indexOf("HTTP/")).trim();  //请求方式  method =firstLine.substring(0, requestInfo.indexOf("/")).trim().toLowerCase(); |

# 获取url

分解字符串首行，获取url,注意,如果请求方式为get,url中可能存在请求参数，需要将请求参数进行分离。url如何获取?

·    在获取的首行中，截取字符串，从/到最后即可

·    1、get 可能存在请求参数,如果存在 通过 ? 分割字符即可

·    2、post 即为url

在 parseRequestInfo()中加入以下代码即可:

|  |
| --- |
| //url  url=firstLine.substring(requestInfo.indexOf("/")).trim();  //我们这里只考虑两种get|post情况,如果请求方式为get 需要进一步判断  if(method.equals("get")&&url.contains("?")){  String[] urlPara=url.split("\\?");  url=urlPara[0]; //请求资源  queryStr=urlPara[1];//请求参数  }else if(method.equals("post")){  } |

到此，已经获取请求方式和URL，下一步需要获取请求参数和封装请求参数。

# 总结

·    封装Request 协议

·    获取method

·    获取url

第202集

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季202集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# 服务器的分发器的封装：

    加入类与多线程知识：

手写tomcat\_servlet\_分发器

本节课主要讲解业务分离，加入servlet,同时以多线程实现分发器，以达到服务器可以同时处理多个请求。

# 目标

1、    分离业务，加入Servlet。Servlet类处理业务类，将Server中receive方法封装到此类中即可

|  |
| --- |
| /\*\*   \* 使用Servlet专门处理业务   \*   \* @author Administrator   \*   \*/  public class Servlet {  public void service(Request request,Response response){  }  } |

2、服务器可以处理多个请求。封装Dispatcher类。主要使用面向对象和多线程加入Dispatcher类中即可。

Dispatcher类为:

|  |
| --- |
| /\*\*   \* 使用多线程封装分发器    1、一个请求响应 一根管道   \* @author Administrator   \*   \*/  public class Dispatcher implements Runnable{  @Override  public void run() {  }    } |

# 封装Servlet类

将receive中的 处理业务代码 放入到service方法中即可.

|  |
| --- |
| public void service(Request request,Response response){  //获取请求信息  String uname =request.getParameter("uname");  String[] fav =request.getParameterValues("fav");  StringBuilder favStr =new StringBuilder();  if(null!=fav && fav.length>0){  for(String f:fav){  if(f.equals("0")){  favStr.append(" 豪放型 ");  }else if(f.equals("1")){  favStr.append(" 经济节约型 ");  }else if(f.equals("2")){  favStr.append(" 萝莉型 ");  }  }  }  //响应  PrintWriter writer=response.getWriter();  //输出内容到服务器内存中  writer.append("<html><head><title>HTTP响应示例</title>");  writer.append("</head><body>Hello <font color='red'>"+  uname+"</font>,您的爱好为"+favStr.toString()  +"!</body></html>");  } |

这样一来，可见servlet为专门处理业务的类。

# 封装Dispatcher类

加入四个属性:

|  |
| --- |
| private Socket socket;  private Request request;  private Response response;  private int code =200; |

使用构造器接收Socket,通过Socket构建Request和Response

|  |
| --- |
| Dispatcher(Socket socket){  this.socket=socket;  //构建请求  try {  request =new Request(socket.getInputStream());  response =new Response(socket.getOutputStream());  } catch (IOException e) {  e.printStackTrace();  }  } |

将Servlet 处理 放入到run中

|  |
| --- |
| public void run() {    try {  Servlet servlet =new Servlet();  servlet.service(request, response);  } catch (Exception e) {  code =500;  }  //推送到客户端  try {  response.pushToBrowser(code);  } catch (IOException e) {  e.printStackTrace();  code=500;  try {  response.pushToBrowser(code);  } catch (IOException e1) {  e1.printStackTrace();  }  }  } |

将Server中的receive方法更改为:接收多个处理

|  |
| --- |
| /\*\*   \* 服务   \*/  private void receive(){  while(this.isRunning){  try {  Socket client =server.accept();  System.out.println("一个客户端建立了连接");  new Thread(new Dispatcher(client)).start();  } catch (IOException e) {  }  }  } |

由此，可以处理多次请求，当然每次请求返回相同的结果。

# 总结

·    封装Servlet

·    封装分发器

·    处理多次请求

package com.bjsxt.server.demo1;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

/\*\*

 \* 创建服务器，并启动

 \*

 \* 1、请求

 \* 2、响应

 \* @author Administrator

 \*

 \*/

public class Server6 {

private ServerSocket server;

public static final String CRLF="\r\n";

public static final String BLANK=" ";

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Server6 server = new Server6();

server.start();

}

/\*\*

 \* 启动方法

 \*/

public void start(){

try {

server = new ServerSocket(8888);

this.receive();

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

 \* 接收客户端

 \*/

private void receive(){

try {

Socket client =server.accept();

Servlet serv =new Servlet();

Request req =new Request(client.getInputStream());

Response rep =new Response(client.getOutputStream());

serv.service(req,rep);

rep.pushToClient(200);

} catch (IOException e) {

}

}

/\*\*

 \* 停止服务器

 \*/

public void stop(){

}

}

## servlet类的封装：

package com.bjsxt.server.demo1;

public class Servlet {

public void service(Request req,Response rep){

rep.println("<html><head><title>HTTP响应示例</title>");

rep.println("</head><body>");

rep.println("欢迎:").println(req.getParameter("uname")).println("回来");

rep.println("</body></html>");

}

}

## server7

package com.bjsxt.server.demo1;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import com.bjsxt.util.CloseUtil;

/\*\*

 \* 创建服务器，并启动

 \*

 \* 1、请求

 \* 2、响应

 \* @author Administrator

 \*

 \*/

public class Server7 {

private ServerSocket server;

public static final String CRLF="\r\n";

public static final String BLANK=" ";

private boolean isShutDown= false;

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Server7 server = new Server7();

server.start();

}

/\*\*

 \* 启动方法

 \*/

public void start(){

start(8888);

}

/\*\*

 \* 指定端口的启动方法

 \*/

public void start(int port){

try {

server = new ServerSocket(port);

this.receive();

} catch (IOException e) {

//e.printStackTrace();

stop();

}

}

/\*\*

 \* 接收客户端

 \*/

private void receive(){

try {

while(!isShutDown){

new Thread(new Dispatcher(server.accept())).start();

}

} catch (IOException e) {

//e.printStackTrace();

stop();

}

}

/\*\*

 \* 停止服务器

 \*/

public void stop(){

isShutDown=true;

CloseUtil.closeSocket(server);

}

}

package com.bjsxt.server.demo1;

import java.io.IOException;

import java.net.Socket;

import com.bjsxt.util.CloseUtil;

/\*\*

 \* 一个请求与响应 就一个此对象

 \* @author Administrator

 \*

 \*/

public class Dispatcher implements Runnable{

private Socket client;

private Request req;

private Response rep;

private int code=200;

Dispatcher(Socket client){

this.client=client;

try {

req =new Request(client.getInputStream());

rep =new Response(client.getOutputStream());

} catch (IOException e) {

//e.printStackTrace();

code =500;

return ;

}

}

@Override

public void run() {

Servlet serv =new Servlet();

serv.service(req,rep);

try {

rep.pushToClient(code); //推送到客户端

} catch (IOException e) {

//e.printStackTrace();

}

try {

rep.pushToClient(500);

} catch (IOException e) {

e.printStackTrace();

}

CloseUtil.closeSocket(client);

}

}

关闭流的工具类参考前几集的封装：

# **第203集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季203集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com】](http://www.bjsxt.xn--com-003b/)

# httpServet多请求处理多态：

## Server类代码：

package com.bjsxt.server.demo2;

import java.io.IOException;

import java.net.ServerSocket;

import com.bjsxt.util.CloseUtil;

/\*\*

 \* 创建服务器，并启动

 \*

 \* 1、请求

 \* 2、响应

 \* @author Administrator

 \*

 \*/

public class Server {

private ServerSocket server;

public static final String CRLF="\r\n";

public static final String BLANK=" ";

private boolean isShutDown= false;

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Server server = new Server();

server.start();

}

/\*\*

 \* 启动方法

 \*/

public void start(){

start(8888);

}

/\*\*

 \* 指定端口的启动方法

 \*/

public void start(int port){

try {

server = new ServerSocket(port);

this.receive();

} catch (IOException e) {

//e.printStackTrace();

stop();

}

}

/\*\*

 \* 接收客户端

 \*/

private void receive(){

try {

while(!isShutDown){

new Thread(new Dispatcher(server.accept())).start();

}

} catch (IOException e) {

//e.printStackTrace();

stop();

}

}

/\*\*

 \* 停止服务器

 \*/

public void stop(){

isShutDown=true;

CloseUtil.closeSocket(server);

}

}

## Servlet类代码：

package com.bjsxt.server.demo2;

/\*\*

 \* 抽象为一个父类

 \* @author Administrator

 \*

 \*/

public abstract class Servlet {

public void service(Request req,Response rep) throws Exception{

this.doGet(req,rep);

this.doPost(req,rep);

}

public abstract void doGet(Request req,Response rep) throws Exception;

public abstract void doPost(Request req,Response rep) throws Exception;

}

## ServletContext类代码如下：

package com.bjsxt.server.demo2;

import java.util.HashMap;

import java.util.Map;

/\*\*

 \* 上下文

 \* @author Administrator

 \*

 \*/

public class ServletContext {

//为每一个servlet取个别名

// login  -->LoginServlet

private Map<String,Servlet> servlet ;

//url -->login

//   /log -->login

//   /login -->login

private Map<String,String> mapping;

ServletContext(){

servlet =new HashMap<String,Servlet>();

mapping =new HashMap<String,String>();

}

public Map<String, Servlet> getServlet() {

return servlet;

}

public void setServlet(Map<String, Servlet> servlet) {

this.servlet = servlet;

}

public Map<String, String> getMapping() {

return mapping;

}

public void setMapping(Map<String, String> mapping) {

this.mapping = mapping;

} }

## Request类：

package com.bjsxt.server.demo2;

import java.io.InputStream;

import java.io.UnsupportedEncodingException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.StringTokenizer;

import com.bjsxt.util.CloseUtil;

/\*\*

 \* 封装request

 \* @author Administrator

 \*

 \*/

public class Request {

//请求方式

private String method;

//请求资源

private String url;

//请求参数

private Map<String,List<String>> parameterMapValues;

//内部

public static final String CRLF="\r\n";

private InputStream is;

private String requestInfo;

public Request(){

method ="";

url ="";

parameterMapValues=new HashMap<String,List<String>>();

requestInfo="";

}

public Request(InputStream is){

this();

this.is=is;

try {

byte[] data = new byte[20480];

int len = is.read(data);

requestInfo = new String(data, 0, len);

} catch (Exception e) {

return ;

}

//分析请求信息

parseRequestInfo();

}

/\*\*

 \* 分析请求信息

 \*/

private void parseRequestInfo(){

if(null==requestInfo ||(requestInfo=requestInfo.trim()).equals("")){

return ;

}

/\*\*

 \* =====================================

 \* 从信息的首行分解出 :请求方式    请求路径   请求参数(get可能存在)

 \*   如:GET /index.html?name=123&pwd=5456 HTTP/1.1

 \*

 \* 如果为post方式，请求参数可能在 最后正文中

 \*

 \* 思路:

 \* 1)请求方式 :找出第一个/  截取即可

 \* 2)请求资源:找出第一个/   HTTP/

 \* =====================================

 \*/

String paramString =""; //接收请求参数

//1、获取请求方式

String firstLine =requestInfo.substring(0,requestInfo.indexOf(CRLF));

int idx =requestInfo.indexOf("/"); // /的位置

this.method=firstLine.substring(0, idx).trim();

String urlStr =firstLine.substring(idx,firstLine.indexOf("HTTP/")).trim();

if(this.method.equalsIgnoreCase("post")){

this.url=urlStr;

paramString=requestInfo.substring(requestInfo.lastIndexOf(CRLF)).trim();

}else if(this.method.equalsIgnoreCase("get")){

if(urlStr.contains("?")){ //是否存在参数

String[] urlArray=urlStr.split("\\?");

this.url=urlArray[0];

paramString=urlArray[1];//接收请求参数

}else{

this.url=urlStr;

}

}

//不存在请求参数

if(paramString.equals("")){

return ;

}

//2、将请求参数封装到Map中

parseParams(paramString);

}

private void parseParams(String paramString){

//分割 将字符串转成数组

StringTokenizer token=new StringTokenizer(paramString,"&");

while(token.hasMoreTokens()){

String keyValue =token.nextToken();

String[] keyValues=keyValue.split("=");

if(keyValues.length==1){

keyValues =Arrays.copyOf(keyValues, 2);

keyValues[1] =null;

}

String key = keyValues[0].trim();

String value = null==keyValues[1]?null:decode(keyValues[1].trim(),"gbk");

//转换成Map 分拣

if(!parameterMapValues.containsKey(key)){

parameterMapValues.put(key,new ArrayList<String>());

}

List<String> values =parameterMapValues.get(key);

values.add(value);

}

}

/\*\*

 \* 解决中文

 \* @param value

 \* @param code

 \* @return

 \*/

private String decode(String value,String code){

try {

return java.net.URLDecoder.decode(value, code);

} catch (UnsupportedEncodingException e) {

//e.printStackTrace();

}

return null;

}

/\*\*

 \* 根据页面的name 获取对应的多个值

 \* @param args

 \*/

public String[] getParameterValues(String name){

List<String> values=null;

if((values=parameterMapValues.get(name))==null){

return null;

}else{

return values.toArray(new String[0]);

}

}

/\*\*

 \* 根据页面的name 获取对应的单个值

 \* @param args

 \*/

public String getParameter(String name){

String[] values =getParameterValues(name);

if(null==values){

return null;

}

return values[0];

}

public String getUrl() {

return url;

}

public void close(){

CloseUtil.closeIO(is);

}

}

## Response类：

package com.bjsxt.server.demo2;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.OutputStream;

import java.io.OutputStreamWriter;

import java.net.Socket;

import java.util.Date;

import com.bjsxt.util.CloseUtil;

/\*\*

 \* 封装响应信息

 \* @author Administrator

 \*

 \*/

public class Response {

//两个常量

public static final String CRLF="\r\n";

public static final String BLANK=" ";

//流

private BufferedWriter bw ;

//正文

private StringBuilder content;

//存储头信息

private StringBuilder headInfo;

//存储正文长度

private int len =0;

public Response(){

headInfo =new StringBuilder();

content =new StringBuilder();

len =0;

}

public Response(Socket client){

this();

try {

bw= new BufferedWriter(new OutputStreamWriter(client.getOutputStream()));

} catch (IOException e) {

headInfo=null;

}

}

public Response(OutputStream os){

this();

bw= new BufferedWriter(new OutputStreamWriter(os));

}

/\*\*

 \* 构建正文

 \*/

public Response print(String info){

content.append(info);

len+=info.getBytes().length;

return this;

}

/\*\*

 \* 构建正文+回车

 \*/

public Response println(String info){

content.append(info).append(CRLF);

len+=(info+CRLF).getBytes().length;

return this;

}

/\*\*

 \* 构建响应头

 \*/

private void createHeadInfo(int code){

//1)  HTTP协议版本、状态代码、描述

headInfo.append("HTTP/1.1").append(BLANK).append(code).append(BLANK);

switch(code){

case 200:

headInfo.append("OK");

break;

case 404:

headInfo.append("NOT FOUND");

break;

case 505:

headInfo.append("SEVER ERROR");

break;

}

headInfo.append(CRLF);

//2)  响应头(Response Head)

headInfo.append("Server:bjsxt Server/0.0.1").append(CRLF);

headInfo.append("Date:").append(new Date()).append(CRLF);

headInfo.append("Content-type:text/html;charset=GBK").append(CRLF);

//正文长度 ：字节长度

headInfo.append("Content-Length:").append(len).append(CRLF);

headInfo.append(CRLF); //分隔符

}

//推送到客户端

void pushToClient(int code) throws IOException{

if(null==headInfo){

code =500;

}

createHeadInfo(code);

//头信息+分割符

bw.append(headInfo.toString());

//正文

bw.append(content.toString());

bw.flush();

}

public void close(){

CloseUtil.closeIO(bw);

}

}

## Dispatch类代码如下：

package com.bjsxt.server.demo2;

import java.io.IOException;

import java.net.Socket;

import com.bjsxt.util.CloseUtil;

/\*\*

 \* 一个请求与响应 就一个此对象

 \* @author Administrator

 \*

 \*/

public class Dispatcher implements Runnable{

private Socket client;

private Request req;

private Response rep;

private int code=200;

Dispatcher(Socket client){

this.client=client;

try {

req =new Request(client.getInputStream());

rep =new Response(client.getOutputStream());

} catch (IOException e) {

//e.printStackTrace();

code =500;

return ;

}

}

@Override

public void run() {

try {

Servlet serv =WebApp.getServlet(req.getUrl());

if(null==serv){

this.code=404; //找不到处理

}else{

serv.service(req, rep);

}

rep.pushToClient(code); //推送到客户端

}catch (Exception e) {

e.printStackTrace();

this.code=500;

}

try {

rep.pushToClient(500);

} catch (IOException e) {

e.printStackTrace();

}

req.close();

rep.close();

CloseUtil.closeSocket(client);

}

}

## RegisterServlet类代码：

package com.bjsxt.server.demo2;

public class RegisterServlet extends Servlet{

@Override

public void doGet(Request req,Response rep) throws Exception {

}

@Override

public void doPost(Request req,Response rep) throws Exception {

rep.println("<html><head><title>返回注册</title>");

rep.println("</head><body>");

rep.println("你的用户名为:"+req.getParameter("uname"));

rep.println("</body></html>");

}

}

## loginServlet代码：

package com.bjsxt.server.demo2;

public class LoginServlet extends Servlet{

@Override

public void doGet(Request req,Response rep) throws Exception {

String name = req.getParameter("uname");

String pwd =req.getParameter("pwd");

if(login(name,pwd)){

rep.println("登录成功");

}else{

rep.println("登录失败");

}

}

public boolean login(String name,String pwd){

return name.equals("bjsxt") && pwd.equals("12346");

}

@Override

public void doPost(Request req,Response rep) throws Exception {

}

}

## WabApp类代码如下：

package com.bjsxt.server.demo2;

import java.util.Map;

public class WebApp {

private static ServletContext contxt;

static{

contxt =new ServletContext();

Map<String,String> mapping =contxt.getMapping();

mapping.put("/login", "login");

mapping.put("/log", "login");

mapping.put("/reg", "register");

Map<String,Servlet> servlet =contxt.getServlet();

servlet.put("login", new LoginServlet());

servlet.put("register", new RegisterServlet());

}

public static Servlet getServlet(String url){

if((null==url)||(url=url.trim()).equals("")){

return null;

}

return contxt.getServlet().get(contxt.getMapping().get(url));

}

}

# **第204集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季204集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com](http://www.bjsxt.xn--com-003b/)】

# 手写服务器之反射机制：

## 反射的代码如下：

package com.bjsxt.reflect;

/\*\*

 \* 获取结构信息Class对象(源头)

 \* @author Administrator

 \*

 \*/

public class Demo01 {

public static void main(String[] args) throws ClassNotFoundException {

String  str ="abc";

//Class对象

//对象.getClass()

Class<?> clz =str.getClass();

//类.class

clz =String.class;

//完整路径

clz=Class.forName("java.lang.String");

}

}

package com.bjsxt.reflect;

import com.bjsxt.server.demo3.Servlet;

/\*\*

 \* 创建实例  调用空构造

 \*

 \*/

public class Demo02 {

/\*\*

 \* @param args

 \* @throws ClassNotFoundException

 \* @throws IllegalAccessException

 \* @throws InstantiationException

 \*/

public static void main(String[] args) throws ClassNotFoundException, InstantiationException, IllegalAccessException {

Class<?> clz =Class.forName("com.bjsxt.server.demo03.LoginServlet");

// 调用空构造 确保空构造存在

Servlet ser=(Servlet)clz.newInstance();

//retrun ser;

}

}

## Server类的代码如下：

package com.bjsxt.server.demo3;

import java.io.IOException;

import java.net.ServerSocket;

import com.bjsxt.util.CloseUtil;

/\*\*

 \* 创建服务器，并启动 \*

 \* 1、请求

 \* 2、响应

 \*/

public class Server {

private ServerSocket server;

public static final String CRLF="\r\n";

public static final String BLANK=" ";

private boolean isShutDown= false;

/\*\*

 \* @param args

 \*/

public static void main(String[] args) {

Server server = new Server();

server.start();

}

/\*\*

 \* 启动方法

 \*/

public void start(){

start(8888);

}

/\*\*

 \* 指定端口的启动方法

 \*/

public void start(int port){

try {

server = new ServerSocket(port);

this.receive();

} catch (IOException e) {

//e.printStackTrace();

stop();

}

}

/\*\*

 \* 接收客户端

 \*/

private void receive(){

try {

while(!isShutDown){

new Thread(new Dispatcher(server.accept())).start();

}

} catch (IOException e) {

//e.printStackTrace();

stop();

}

}

/\*\*

 \* 停止服务器

 \*/

public void stop(){

isShutDown=true;

CloseUtil.closeSocket(server);

}

}

## Servlet代码如下：

package com.bjsxt.server.demo3;

/\*\*

 \* 抽象为一个父类

 \* @author Administrator

 \*

 \*/

public abstract class Servlet {

public void service(Request req,Response rep) throws Exception{

this.doGet(req,rep);

this.doPost(req,rep);

}

public abstract void doGet(Request req,Response rep) throws Exception;

public abstract void doPost(Request req,Response rep) throws Exception;

}

## ServletContext代码：

package com.bjsxt.server.demo3;

import java.util.HashMap;

import java.util.Map;

/\*\*

 \* 上下文

 \* @author Administrator \*

 \*/

public class ServletContext {

//为每一个servlet取个别名

// login  -->com.bjsxt.server.demo03.LoginServlet

private Map<String,String> servlet ;

//url -->login

//   /log -->login

//   /login -->login

private Map<String,String> mapping;

ServletContext(){

servlet =new HashMap<String,String>();

mapping =new HashMap<String,String>();

}

public Map<String, String> getServlet() {

return servlet;

}

public void setServlet(Map<String, String> servlet) {

this.servlet = servlet;

}

public Map<String, String> getMapping() {

return mapping;

}

public void setMapping(Map<String, String> mapping) {

this.mapping = mapping;

}

}

## Request代码如下：

package com.bjsxt.server.demo3;

import java.io.InputStream;

import java.io.UnsupportedEncodingException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.StringTokenizer;

import com.bjsxt.util.CloseUtil;

/\*\*

 \* 封装request

 \* @author Administrator

 \*

 \*/

public class Request {

//请求方式

private String method;

//请求资源

private String url;

//请求参数

private Map<String,List<String>> parameterMapValues;

//内部

public static final String CRLF="\r\n";

private InputStream is;

private String requestInfo;

public Request(){

method ="";

url ="";

parameterMapValues=new HashMap<String,List<String>>();

requestInfo="";

}

public Request(InputStream is){

this();

this.is=is;

try {

byte[] data = new byte[20480];

int len = is.read(data);

requestInfo = new String(data, 0, len);

} catch (Exception e) {

return ;

}

//分析请求信息

parseRequestInfo();

}

/\*\*

 \* 分析请求信息

 \*/

private void parseRequestInfo(){

if(null==requestInfo ||(requestInfo=requestInfo.trim()).equals("")){

return ;

}

/\*\*

 \* =====================================

 \* 从信息的首行分解出 :请求方式    请求路径   请求参数(get可能存在)

 \*   如:GET /index.html?name=123&pwd=5456 HTTP/1.1

 \*

 \* 如果为post方式，请求参数可能在 最后正文中

 \*

 \* 思路:

 \* 1)请求方式 :找出第一个/  截取即可

 \* 2)请求资源:找出第一个/   HTTP/

 \* =====================================

 \*/

String paramString =""; //接收请求参数

//1、获取请求方式

String firstLine =requestInfo.substring(0,requestInfo.indexOf(CRLF));

int idx =requestInfo.indexOf("/"); // /的位置

this.method=firstLine.substring(0, idx).trim();

String urlStr =firstLine.substring(idx,firstLine.indexOf("HTTP/")).trim();

if(this.method.equalsIgnoreCase("post")){

this.url=urlStr;

paramString=requestInfo.substring(requestInfo.lastIndexOf(CRLF)).trim();

}else if(this.method.equalsIgnoreCase("get")){

if(urlStr.contains("?")){ //是否存在参数

String[] urlArray=urlStr.split("\\?");

this.url=urlArray[0];

paramString=urlArray[1];//接收请求参数

}else{

this.url=urlStr;

}

}

//不存在请求参数

if(paramString.equals("")){

return ;

}

//2、将请求参数封装到Map中

parseParams(paramString);

}

private void parseParams(String paramString){

//分割 将字符串转成数组

StringTokenizer token=new StringTokenizer(paramString,"&");

while(token.hasMoreTokens()){

String keyValue =token.nextToken();

String[] keyValues=keyValue.split("=");

if(keyValues.length==1){

keyValues =Arrays.copyOf(keyValues, 2);

keyValues[1] =null;

}

String key = keyValues[0].trim();

String value = null==keyValues[1]?null:decode(keyValues[1].trim(),"gbk");

//转换成Map 分拣

if(!parameterMapValues.containsKey(key)){

parameterMapValues.put(key,new ArrayList<String>());

}

List<String> values =parameterMapValues.get(key);

values.add(value);

}

}

/\*\*

 \* 解决中文

 \* @param value

 \* @param code

 \* @return

 \*/

private String decode(String value,String code){

try {

return java.net.URLDecoder.decode(value, code);

} catch (UnsupportedEncodingException e) {

//e.printStackTrace();

}

return null;

}

/\*\*

 \* 根据页面的name 获取对应的多个值

 \* @param args

 \*/

public String[] getParameterValues(String name){

List<String> values=null;

if((values=parameterMapValues.get(name))==null){

return null;

}else{

return values.toArray(new String[0]);

}

}

/\*\*

 \* 根据页面的name 获取对应的单个值

 \* @param args

 \*/

public String getParameter(String name){

String[] values =getParameterValues(name);

if(null==values){

return null;

}

return values[0];

}

public String getUrl() {

return url;

}

public void close(){

CloseUtil.closeIO(is);

}

}

## Response类代码如下：

package com.bjsxt.server.demo3;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.OutputStream;

import java.io.OutputStreamWriter;

import java.net.Socket;

import java.util.Date;

import com.bjsxt.util.CloseUtil;

/\*\*

 \* 封装响应信息

 \* @author Administrator

 \*

 \*/

public class Response {

 //两个常量

 public static final String CRLF="\r\n";

 public static final String BLANK=" ";

 //流

 private BufferedWriter bw ;

 //正文

 private StringBuilder content;

 //存储头信息

 private StringBuilder headInfo;

 //存储正文长度

 private int len =0;

 public Response(){

 headInfo =new StringBuilder();

 content =new StringBuilder();

 len =0;

 }

 public Response(Socket client){

 this();

 try {

 bw= new BufferedWriter(new OutputStreamWriter(client.getOutputStream()));

 } catch (IOException e) {

 headInfo=null;

 }

 }

 public Response(OutputStream os){

 this();

 bw= new BufferedWriter(new OutputStreamWriter(os));

 }

 /\*\*

 \* 构建正文

 \*/

 public Response print(String info){

 content.append(info);

 len+=info.getBytes().length;

 return this;

 }

 /\*\*

 \* 构建正文+回车

 \*/

 public Response println(String info){

 content.append(info).append(CRLF);

 len+=(info+CRLF).getBytes().length;

 return this;

 }

 /\*\*

 \* 构建响应头

 \*/

 private void createHeadInfo(int code){

 //1)  HTTP协议版本、状态代码、描述

 headInfo.append("HTTP/1.1").append(BLANK).append(code).append(BLANK);

 switch(code){

 case 200:

 headInfo.append("OK");

 break;

 case 404:

 headInfo.append("NOT FOUND");

 break;

 case 505:

 headInfo.append("SEVER ERROR");

 break;

 }

 headInfo.append(CRLF);

 //2)  响应头(Response Head)

 headInfo.append("Server:bjsxt Server/0.0.1").append(CRLF);

 headInfo.append("Date:").append(new Date()).append(CRLF);

 headInfo.append("Content-type:text/html;charset=GBK").append(CRLF);

//正文长度 ：字节长度

 headInfo.append("Content-Length:").append(len).append(CRLF);

 headInfo.append(CRLF); //分隔符

 }

 //推送到客户端

 void pushToClient(int code) throws IOException{

 if(null==headInfo){

 code =500;

 }

 createHeadInfo(code);

 //头信息+分割符

 bw.append(headInfo.toString());

 //正文

 bw.append(content.toString());

 bw.flush();

 }

 public void close(){

CloseUtil.closeIO(bw);

 }

}

## DisPatcher类代码如下：

package com.bjsxt.server.demo3;

import java.io.IOException;

import java.net.Socket;

import com.bjsxt.util.CloseUtil;

/\*\*

 \* 一个请求与响应 就一个此对象

 \* @author Administrator

 \*

 \*/

public class Dispatcher implements Runnable{

 private Socket client;

 private Request req;

 private Response rep;

 private int code=200;

Dispatcher(Socket client){

 this.client=client;

 try {

 req =new Request(client.getInputStream());

 rep =new Response(client.getOutputStream());

 } catch (IOException e) {

 //e.printStackTrace();

 code =500;

 return ;

 }

 }

 @Override

 public void run() {

 try {

Servlet serv =WebApp.getServlet(req.getUrl());

 if(null==serv){

 this.code=404; //找不到处理

 }else{

 serv.service(req, rep);

 }

 rep.pushToClient(code); //推送到客户端

 }catch (Exception e) {

 e.printStackTrace();

 this.code=500;

 }

 try {

 rep.pushToClient(500);

 } catch (IOException e) {

 e.printStackTrace();

 }

 req.close();

 rep.close();

CloseUtil.closeSocket(client);

 }

}

## WebApp代码如下：

package com.bjsxt.server.demo3;

import java.util.Map;

public class WebApp {

 private static ServletContext contxt;

 static{

 contxt =new ServletContext();

Map<String,String> mapping =contxt.getMapping();

 mapping.put("/login", "login");

 mapping.put("/log", "login");

 mapping.put("/reg", "register");

Map<String,String> servlet =contxt.getServlet();

 servlet.put("login", "com.bjsxt.server.demo3.LoginServlet");

 servlet.put("register", "com.bjsxt.server.demo3.RegisterServlet");

 }

 public static Servlet getServlet(String url) throws InstantiationException, IllegalAccessException, ClassNotFoundException{

 if((null==url)||(url=url.trim()).equals("")){

 return null;

 }

 //根据字符串(完整路径)创建对象

 //return contxt.getServlet().get(contxt.getMapping().get(url));

 String name=contxt.getServlet().get(contxt.getMapping().get(url));

 return (Servlet)Class.forName(name).newInstance();//确保空构造存在

 }

}

# **第205集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季205集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com](http://www.bjsxt.xn--com-003b/)】

# 服务器之xml配置文件的解析：

 解析:基本有两种:

SAX  基于事件流的解析

DOM基于XML文档树结构的解析。  
常见的四种:DOM   SAX  DOM4J  JDOM  
sax 步骤  
1、获得解析工厂  SAXParserFactory  
2、工厂获取解析器  SAXParser  
3、加载文档 Document 注册处理器

## Person类代码：

package com.bjsxt.xml;

public class Person {

private String name;

private int age;

public Person() {

}

public Person(String name, int age) {

super();

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

## Person.xml

<?xml version="1.0" encoding="UTF-8" ?>

<persons>

<person>

<name>至尊宝</name>

<age>9000</age>

</person>

<person>

<name>白晶晶</name>

<age>7000</age>

</person>

</persons>

## ParseDemo01代码：

package com.bjsxt.xml;

import java.io.IOException;

import java.util.List;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.parsers.SAXParser;

import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.SAXException;

public class ParseDemo01 {

/\*\*

 \* @param args

 \* @throws SAXException

 \* @throws ParserConfigurationException

 \* @throws IOException

 \*/

public static void main(String[] args) throws ParserConfigurationException, SAXException, IOException {

//1、获取解析工厂

SAXParserFactory factory=SAXParserFactory.newInstance();

//2、从解析工厂获取解析器

SAXParser parse =factory.newSAXParser();

//3、加载文档 Document 注册处理器

//4、编写处理器

PersonHandler handler=new PersonHandler();

parse.parse(Thread.currentThread().getContextClassLoader()

.getResourceAsStream("com/bjsxt/xml/person.xml")

,handler );

List<Person> persons =handler.getPersons();

for(Person p:persons){

System.out.println(p.getName()+"-->"+p.getAge());

}

}

}

## PersonHandler代码：

package com.bjsxt.xml;

import java.util.ArrayList;

import java.util.List;

import org.xml.sax.Attributes;

import org.xml.sax.SAXException;

import org.xml.sax.helpers.DefaultHandler;

/\*\*

 \* 存储对象

 \* @author Administrator

 \*

 \*/

public class PersonHandler extends DefaultHandler {

private List<Person> persons;

private Person person;

private String tag;//记录标签名

@Override

public void startDocument() throws SAXException {

// TODO Auto-generated method stub

//System.out.println("处理文档开始");

persons =new ArrayList<Person>();

}

@Override

public void startElement(String uri, String localName, String qName,

Attributes attributes) throws SAXException {

System.out.println("开始一个元素" +qName);

if(null!=qName){

tag=qName;

}

if(null!=qName &&qName.equals("person")){

person =new Person();

}

}

@Override

public void characters(char[] ch, int start, int length)

throws SAXException {

String str =new String(ch,start,length);

if(null!=tag &&tag.equals("name")){

//System.out.println(new String(ch,start,length));

person.setName(str);

}else if(null!=tag &&tag.equals("age")){

Integer age = Integer.valueOf(str);

person.setAge(age);

}

}

@Override

public void endElement(String uri, String localName, String qName)

throws SAXException {

//System.out.println("结束一个元素" +qName);

if(qName.equals("person")){

this.persons.add(person);

}

tag =null;

}

@Override

public void endDocument() throws SAXException {

System.out.println("文档处理结束");

}

public List<Person> getPersons() {

return persons;

}

public void setPersons(List<Person> persons) {

this.persons = persons;

}

}

# **第206集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季206集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com](http://www.bjsxt.xn--com-003b/)】

# xml配置文件，sax解析的应用：

    此版本的服务器是对上一个版本的升级，所以好多相同代码可以参考上一级的代码的，新增的代码如下，也可以直接下载三百集的源代码直接导入到本机项目逐步调试，观看项目的效果：

## Entity类代码

package com.bjsxt.server.demo4;

/\*

 <servlet>

<servlet-name>login</servlet-name>

  <servlet-class>com.bjsxt.server.demo4.LoginServlet</servlet-class>

</servlet>

 \*/

public class Entity {

private String name;

private String clz;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getClz() {

return clz;

}

public void setClz(String clz) {

this.clz = clz;

}

}

## Mapping代码如下：

package com.bjsxt.server.demo4;

import java.util.ArrayList;

import java.util.List;

/\*

 <servlet-mapping>

  <servlet-name>login</servlet-name>

  <url-pattern>/login</url-pattern>

 </servlet-mapping>

<servlet-mapping>

  <servlet-name>login</servlet-name>

  <url-pattern>/log</url-pattern>

 </servlet-mapping>

 \*/

public class Mapping {

private String name;

private List<String> urlPattern;

public Mapping(){

urlPattern =new ArrayList<String>();

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public List<String> getUrlPattern() {

return urlPattern;

}

public void setUrlPattern(List<String> urlPattern) {

this.urlPattern = urlPattern;

}

}

## web.xml

<?xml version="1.0" encoding="UTF-8"?>

 <web-app>

<servlet>

  <servlet-name>login</servlet-name>

  <servlet-class>com.bjsxt.server.demo4.LoginServlet</servlet-class>

 </servlet>

 <servlet-mapping>

  <servlet-name>login</servlet-name>

  <url-pattern>/login</url-pattern>

  <url-pattern>/g</url-pattern>

 </servlet-mapping>

<servlet>

  <servlet-name>reg</servlet-name>

  <servlet-class>com.bjsxt.server.demo4.RegisterServlet</servlet-class>

 </servlet>

 <servlet-mapping>

  <servlet-name>reg</servlet-name>

  <url-pattern>/reg</url-pattern>

 </servlet-mapping>

 </web-app>

## WebApp代码：

package com.bjsxt.server.demo4;

import java.util.List;

import java.util.Map;

import javax.xml.parsers.SAXParser;

import javax.xml.parsers.SAXParserFactory;

public class WebApp {

private static ServletContext contxt;

static{

try {

//获取解析工厂

SAXParserFactory factory = SAXParserFactory.newInstance();

//获取解析器

SAXParser sax = factory.newSAXParser();

//指定xml+处理器

WebHandler web = new WebHandler();

sax.parse(Thread.currentThread().getContextClassLoader()

.getResourceAsStream("com/bjsxt/server/demo4/web.xml"), web);

//将list 转成Map

contxt =new ServletContext();

Map<String,String> servlet =contxt.getServlet();

//servlet-name  servlet-class

for(Entity entity:web.getEntityList()){

servlet.put(entity.getName(), entity.getClz());

}

//url-pattern servlet-name

Map<String,String> mapping =contxt.getMapping();

for(Mapping mapp:web.getMappingList()){

List<String> urls =mapp.getUrlPattern();

for(String url:urls ){

mapping.put(url, mapp.getName());

}

}

} catch (Exception e) {

}

}

public static Servlet getServlet(String url) throws InstantiationException, IllegalAccessException, ClassNotFoundException{

if((null==url)||(url=url.trim()).equals("")){

return null;

}

//根据字符串(完整路径)创建对象

//return contxt.getServlet().get(contxt.getMapping().get(url));

String name=contxt.getServlet().get(contxt.getMapping().get(url));

return (Servlet)Class.forName(name).newInstance();//确保空构造存在

}

}

## WebHandler代码：

package com.bjsxt.server.demo4;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.parsers.SAXParser;

import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.Attributes;

import org.xml.sax.SAXException;

import org.xml.sax.helpers.DefaultHandler;

public class WebHandler extends DefaultHandler{

private List<Entity> entityList;

private List<Mapping> mappingList;

private Entity entity;

private Mapping mapping;

private String beginTag ;

private boolean isMap;

  @Override

public void startDocument() throws SAXException {

//文档解析开始

  entityList =new ArrayList<Entity>() ;

  mappingList =new ArrayList<Mapping>() ;

}

@Override

public void startElement(String uri, String localName, String qName,

Attributes attributes) throws SAXException {

//开始元素

if(null!=qName){

beginTag=qName;

if(qName.equals("servlet")){

isMap=false;

entity=new Entity();

}else if(qName.equals("servlet-mapping")){

isMap=true;

mapping=new Mapping();

}

}

}

@Override

public void characters(char[] ch, int start, int length)

throws SAXException {

//处理内容

if(null!=beginTag){

String str =new String(ch,start,length);

if(isMap ){

if(beginTag.equals("servlet-name")){

mapping.setName(str);

}else if(beginTag.equals("url-pattern")){

mapping.getUrlPattern().add(str);

}

}else{

if(beginTag.equals("servlet-name")){

entity.setName(str);

}else if(beginTag.equals("servlet-class")){

entity.setClz(str);

}

}

}

}

@Override

public void endElement(String uri, String localName, String qName)

throws SAXException {

//结束元素

if(null!=qName){

if(qName.equals("servlet")){

entityList.add(entity);

}else if(qName.equals("servlet-mapping")){

mappingList.add(mapping);

}

}

beginTag=null;

}

@Override

public void endDocument() throws SAXException {

//文档解析结束

}

public static void main(String[] args) throws ParserConfigurationException, SAXException, IOException {

//获取解析工厂

SAXParserFactory factory =SAXParserFactory.newInstance();

//获取解析器

SAXParser sax =factory.newSAXParser();

//指定xml+处理器

WebHandler web = new WebHandler();

sax.parse(Thread.currentThread().getContextClassLoader()

.getResourceAsStream("com/bjsxt/server/demo4/web.xml")

,web);

System.out.println(web.getEntityList());

}

public List<Entity> getEntityList() {

return entityList;

}

public void setEntityList(List<Entity> entityList) {

this.entityList = entityList;

}

public List<Mapping> getMappingList() {

return mappingList;

}

public void setMappingList(List<Mapping> mappingList) {

this.mappingList = mappingList;

}

}

# **第207集**

尚学堂JAVA最全教程

配合JAVA300集视频课程，一集一个文档

本文档配合：java300集第一季207集

【最新文档更新请加入尚学堂[www.bjsxt.cn】](http://www.bjsxt.xn--cn-223a/)

【专业JAVA培训机构,真正零首付入学[www.bjsxt.com](http://www.bjsxt.xn--com-003b/)】

# 手写Tomcat服务器的总结版总结：

    把一前版本的代码放到一个新的项目里,形成一个新的服务器项目，

    代码参考以前版本和这集的视频即可：