mvn archetype:generate -DgroupId=com.huangfgui -DartifactId=CLI  
-DarchetypeArtifactId=maven -archetype-quickstart -DinteractiveMode=false

**int arr[] =new int[]{1,2,3}//初始一数组，元素为1,2,3**

**int** arr2[] =**new int**[5];*//初始一长度为5的数组，元素全为0*

**int** a[][]=**new int**[2][]; //初始化一个二维数组，因为无法确定每一维度长度，所有为null  
a[0] = **new int**[]{1,2,3};//  
a[1] = **new int**[4];

ArrayList是一个内部控制动态增长的数组,而且所有参数都是Object类型的,而普通数组

ArrayList myList = **new** ArrayList();  
myList.add(0);

(Array)则可以指定类型,比如:int[] intArray = new int[10];

.class文件

实际上java的每个类被编译成.[class文件](https://www.baidu.com/s?wd=class%E6%96%87%E4%BB%B6&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3ryfLmW61PjI9n1N9uyms0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjDkrjR3P101)的时候，java虚拟机（叫jvm）会自动为这个类生成一个类对象，这个对象保存了这个类的所有信息（成员变量，方法，构造器等），以后这个类要想实例化（也就是创建类的实例或创建类的对象）那么都要以这个class对象为蓝图（或模版）来创建这个类的实例。

日志文件：

日志文件是程序中写日志函数产生的记录程序执行情况的文件。

日志记录器常用级别:

* DEBUG Level指出细粒度信息事件对调试应用程序是非常有帮助的。
* INFO level表明 消息在粗粒度级别上突出强调应用程序的运行过程。
* WARN level表明会出现潜在错误的情形。
* ERROR level指出虽然发生错误事件，但仍然不影响系统的继续运行。

log4j

1.建立包依赖：在项目的pom.xml文件，在dependencies下添加

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency

2、配置日志输出

在 main 下创建 resources 目录，创建 log4j.properties 文本文件[,并把resources文件夹设置为Resources(Project structure)](http://blog.csdn.net/pmlpml/article/details/53537734#fn:2)：

#可以设置级别：debug < info < warn < error

#debug: 显示debug, info, warn, error

#info: 显示info, warn, error

#warn: 显示warn, error

#error: 只显示error

#日志的输出级别由rootLogger和普通Logger设置的最高级别决定。

log4j.rootLogger=warn,appender1

#输出到控制台

log4j.appender.appender1=org.apache.log4j.ConsoleAppender

#样式为TTCCLayout

log4j.appender.appender1.layout=org.apache.log4j.TTCCLayout

#这里配置的是类所在的包test.log4j, 逗号之前未配置日志输出级别，默认为根logger的级别

log4j.logger.test.log4j=, TEST

log4j.appender.TEST=org.apache.log4j.ConsoleAppender

log4j.appender.TEST.layout=org.apache.log4j.TTCCLayout

打包成jar  
maven生命周期中的package

NoClassDefFoundError

NoClassDefFoundError错误的发生，是因为Java虚拟机在编译时能找到合适的类，而在运行时不能找到合适的类导致的错误

。与ClassNotFoundException的不同在于，这个错误发生只在运行时需要加载对应的类不成功，而不是编译时发生

REmote DIctionary Server(Redis) 是一个由Salvatore Sanfilippo写的key-value存储系统。

Redis是一个开源的使用ANSI C语言编写、遵守BSD协议、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，并提供多种语言的API。

它通常被称为数据结构服务器，因为值（value）可以是 字符串(String), 哈希(Map), 列表(list), 集合(sets) 和 有序集合(sorted sets)等类型。

Redis 与其他 key - value 缓存产品有以下三个特点：

* Redis支持数据的持久化，可以将内存中的数据保存在磁盘中，重启的时候可以再次加载进行使用。
* Redis不仅仅支持简单的key-value类型的数据，同时还提供list，set，zset，hash等数据结构的存储。
* Redis支持数据的备份，即master-slave模式的数据备份。

Commons CLI(commands Line Interface命令行接口)

Apache Commons CLI library为用户提供了一个解释命令行的API.它在解释命令行时主要有三个状态，即：定义，解释和询问交互。

参数的配置和常用的命令一样,横杠+参数名+空格+参数值

//CLI定义

Options option = **new** Options();

option.addOption(opt:**"h"**,longOpot:**"help"**)

*// CLI的解析*CommandLineParser parser = **new** PosixParser();  
CommandLine commandLine = **null**;

HelpFormatter hf = **new** HelpFormatter();

commandLine = parser.parse(option, args);

*// CLI询问*

**if** (commandLine.hasOption(**"h"**)) {  
 hf.printHelp(**"Test"**, option, **true**);  
 System.***out***.println(option);  
}

//调用

String arg[] = { **"-h"**};或String arg[] = { **"--help"**};

servlet

为了扩展java作为web服务的功能，其作用就是为java程序提供一个统一的web应用的规范，方便程序员统一的使用这种规范来编写程序，应用容器可以使用提供的规范来实现自己的特性。

工作方式:一个http请求到来，容器将请求封装成servlet中的request对象，在request中你可以得到所有的Http信息，然后可以取出来操作，最后再把数据封装成servlet的response对象，应用容器将response对象解析之后封装成一个http reponse。

servlet就是你写在容器（如tomcat）里的动态资源，远程客户端点击或者一个url请求会封装成一个http请求，到你的服务端后由容器解析http请求，并封装为java servlet请求和响应，最后放进servlet的service方法根据请求作出服务。

Dao主要做数据库的交互工作 Modle 是模型 存放你的实体类 Service 做相应的业务逻辑处理 Action是一个控制器

Action像是服务员，顾客点什么菜，菜上给几号桌，都是ta的职责；Service是厨师，action送来的菜单上的菜全是ta做的；Dao是厨房的小工，和原材料(通过[**hibernate**](http://lib.csdn.net/base/javaee)操作数据库)打交道的事情全是ta管。

service是业务层，DAO是数据访问层。

解耦说的意思是你更改某一层代码，不会影响我其他层代码

如果你会像spring这样的框架，你会了解[面向接口编程](https://www.baidu.com/s?wd=%E9%9D%A2%E5%90%91%E6%8E%A5%E5%8F%A3%E7%BC%96%E7%A8%8B&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLnAc3nvDvm1mLPW0znvu-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1mYrj0zrjRL)，表示层调用控制层，控制层调用业务层，业务层调用[数据访问层](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E8%AE%BF%E9%97%AE%E5%B1%82&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLnAc3nvDvm1mLPW0znvu-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1mYrj0zrjRL)

java注解

元注解:

元注解的作用就是负责注解其他注解。Java5.0定义了4个标准的meta-annotation类型，它们被用来提供对其它 annotation类型作说明。java5.0定义的元注解：

1.@Target

2.@Retention

3.@Documented

4.@Inherited

@Target: **作用：用于描述注解的使用范围（即：被描述的注解可以用在什么地方）**

在Annotation类型的声明中使用了target可更加明晰其修饰的目标。

取值：1.CONSTRUCTOR:用于描述构造器  
　　　　2.FIELD:用于描述域  
　　　　3.LOCAL\_VARIABLE:用于描述局部变量  
　　　　4.METHOD:用于描述方法  
　　　　5.PACKAGE:用于描述包  
　　　　6.PARAMETER:用于描述参数  
　　　　7.TYPE:用于描述类、接口(包括注解类型) 或enum声明

**接口实现的注意点：**

**抽象 abstract**

**接口interface**

**接口实现implements**

**继承 extends**

a）实现一个接口就是要实现该接口的所有的方法(抽象类除外)。  
b）接口中的方法都是抽象的。  
c）多个无关的类可以实现同一个接口，一个类可以实现多个无关的接口。

**extends与implements的区别：**

extends 是继承父类，只要那个类不是声明为final或者那个类定义为abstract的就能继承，JAVA中不支持多重继承，但是可以用接口来实现，这样就用到了implements，继承只能继承一个类，但implements可以实现多个接口，用逗号分开就行了。

类通过实现java.io.Serializable接口可以启用其序列化功能.序列化接口没有方法或字段，仅用于标识可序列化的语义。　Java的"对象序列化"能让你将一个实现了Serializable接口的对象转换成byte流，这样日后要用这个对象时候，你就能把这些byte数据恢复出来，并据此重新构建那个对象了。要想序列化对象，你必须先创建一个OutputStream，然后把它嵌进ObjectOutputStream。这时，你就能用writeObject()方法把对象写入OutputStream了 . 读的时候，你得把InputStream嵌到ObjectInputStream里面，然后再调用readObject()方法。

**1、序列化是干什么的？**

　　简单说就是为了**保存在内存中的各种对象的状态**，并且可以把保存的对象状态再读出来。虽然你可以用你自己的各种各样的方法来保存Object States，但是Java给你提供一种应该比你自己好的保存对象状态的机制,那就是序列化。

example:

//序列化

FileOutputStream fs=**new** FileOutputStream(**"F:\\foo.ser"**);  
ObjectOutputStream os=**new** ObjectOutputStream(fs);  
os.writeObject(myBox);  
os.close();

//反序列化  
FileInputStream fi=**new** FileInputStream(**"F:\\foo.ser"**);  
ObjectInputStream oi=**new** ObjectInputStream(fi);  
App box=(App)oi.readObject();  
oi.close();