JAVA基础部分

由于考虑公司知识产权问题本PPT只提供JAVA基础部分

源自由嵌入Flash谈起，到java编程实现技术

朱臻

Roger Chu

大纲

- 1.从Flash谈起
- 2.一个IBM的Case
- 3 .JAVA基础
- 4.企业级JAVA
- 5.Java虚拟机优化

3JAVA基础

- 开发与运行环境—JDK和JRE区别

- JDK安装时候包括了JRE，
- JRE java runtime environment 是java运行环境。主要是java虚拟机 bin/client/jvm.dll 和各种类库在lib下面 主要java.exe执行
- jdk除此之外还有 javac 编译

3JAVA基础

- 开发与运行环境—JDK编译和运行应用程序

- 把环境变量中的Path配好的作用是使 javac java可以直接使用，这两个文件都在bin下面
- -d是保留原有的包结构

3JAVA基础

- 开发与运行环境—ClASSPATH的作用

- CLASSPATH环境变量的设置主要是用于在java编译和运行的过程中提供类搜索路径的。
- 不过默认情况会先在jre/lib/rt下面搜索，这里面包括java 核心的一些库
- 一般搜索路径包括当前路径，log4j.jar和lib目录等

3JAVA基础-Java语言概述

- Java与C++程序在编译运行的区别

- c++等其他高级语言可以直接的汇编语言，操作系统通过CC或CL直接变成机器语言后可以直接运行。
- 而java不同是先由java变成class（字节码）这个语言只有jvm 虚拟机才能看得懂，机器或操作系统看不懂，虚拟把它转变成可运行的代码然后执行

3JAVA基础-Java语言概述

- 什么是JVM及其工作原理简介

- Java虚拟机的特点是可以跨平台，它用软件模拟了机器硬件，有自己的寄存器，堆栈，指令系统等。
- java 类 运行时，先用ClassLoader找class文件如果没有则抛出异常，如果有则导入类，先链接各个自己码的类，然后初始化，先初始化静态方法和静态域的内容，再执行main中的内容

3JAVA基础-Java语言概述

- Java程序的垃圾回收

- 首先：堆和栈的概念：栈里面主要存放非静态的变量，函数的返回值，函数的结果或一些临时变量（如果不在寄存器的话）。栈的管理都是由系统自动完成的。
- 堆的管理各语言不同（存放如object子类）：C语言需要手动 malloc（）和free（）非语言级
- C++语言则是语言级别的创建销毁，通过new和delete语句。java 只需要根据需求创建，回收由JVM虚拟机来完成。jvm虚拟机垃圾回收的线程优先级非常低。一般可以在类中finalize（）方法中编写销毁前执行代码。还可以通过System.gc()或Runtime.getRuntime().gc()来要求虚拟机进行垃圾回收，但是什么时候执行得由虚拟机决定。

3JAVA基础-Java语言概述

- 如何利用命名提示符把java程序打包

- lib下面的jar.exe可以进行java打包的多种工作。包括创建jar包，把内容添加到jar包，声称jar包的详细列表等。
- jar {c t x u f}[v m e o m i][-c 目录]文件名

3JAVA基础-Java语言概述

- Java Web项目的生成部署和配置

- java web 是属于java EE 的规范。它的目录结构是需要把所有的文件全都统一的放在一个文件夹（项目总文件夹）下，里面要包含一个Web-inf文件夹里面包括class文件夹（用于存放编译后的class字节码）lib存放jar包还有web.xml(配置网站的信息，包括servlet，监听器，过滤器等)。然后跟web-inf同级的存放html jsp等文件。
- servlet是运行在服务器端，由容器进行创建销毁等管理的。jsp是一种特殊的servlet，它包含一些html内容。
- java web 打包也是通过jar.exe 把文件打包成war包，运行时，只要把war包或者文件夹形式的放在tomcat webapp目录下面就可以运行。别的服务器可能不一样。

3JAVA基础-Java语法基础

变量及其作用范围

- 变量分为静态变量，成员变量，局部变量。静态变量随着类的载入而产生，类的消亡而消亡。成员变量随着java对象的创建而产生，随着对象的消亡而消亡。局部变量在方法开始创建，方法结束而销毁。
- if else for 中的{}和直接的{}代码块里面的局部变量不能与外部成员变量重名，但是方法体里面的局部变量可以与外面的成员变量重名。在使用时 可以通过 this.外部变量名来使用。this 代表的是整个类。

3JAVA基础-Java语法基础

Java 的变量数据类型

- java有8种基本数据类型（全小写）：boolean byte char short int long float double
- 因为他们值比较小一般就是1到几个字节所以他们直接存放值
- 其他的全部都是引用类型数据，存放的地址，地址指向在内存堆中具体对象的位置。

3JAVA基础-Java语法基础

- Java包含那些基础数据类型及其包装类型

- 基本数据类型：boolean byte char short int long float double
- 对应包装类：Boolean Byte Character Short Integer Long Float Double
- 基本数据类转成包装类 包装类.valueOf(对应基本类) Integer itg=new Integer("10");
- 包装类转基本数据类型：包装类.intValue() 或者 xxxValue()
- 如List，Set等集合对象中放入的是对象类型

3JAVA基础-Java语法基础

Java中的装箱和拆箱

- java有自动装箱拆箱功能：即可以自动将基本数据类型转变成其包装类，也可以将其包装类自动转变成基本数据类型。
- 如Integer itg =1； 或者 int i=itg等

3JAVA基础-Java语法基础

- Java 的引用和C++的指针有什么区别

- java的引用和C++的指针有相同的地方和不同的地方。
- 相同的：都是用于指向内存中一个具体的内容。相比较而言，引用更安全，指针更灵活。
- 不同：1初始值不同：未赋值时，引用为null，指针指向的位置不确定。
- 2：控制：引用只能在程序内部使用，而指针可以指向任意地方
- 3：可计算：引用不可计算，指针实际是存放地址的int类型，是可以计算的。

3JAVA基础-Java语法基础

- Java 中的main（）方法

- main()是java程序的入口函数，它是公开，静态，无返回值的。并且作为方法必须被包含在某个类中。
- 同时它可以接受控制台传过来的参数。有个String[] args字符串数组接受控制的输入。

3JAVA基础-Java语法基础

- Java 中equal和==区别是什么

- 1 基本类型一样的 2 对象类型== 比较对象是否是同一个，比较引用（地址）是否相同。equal是实际的对象是否相同

3JAVA基础-Java语法基础

Java 的循环结构

- java的循环主要有三种：
- 1 for循环 确定循环次数时使用
- 2 while 条件满足继续循环。
- 3do while 先执行 再判断循环

3JAVA基础-Java语法基础

- Java 中的三元运算符是什么

- java的唯一三元表达式： 表达式1？表达式2：表达式3
- 第一个表达式是一个布尔表达式 返回true 或false，如果true 执行表达式2，如果false执行表达式3。而且2，3表达式的结果要统一。

3JAVA基础-Java语法基础

- Java 的注释有哪些

- 首先由行注释// 不会进入编译层
- 然后是块注释/* .....*/注释掉整个块。也可以作为 方法功能，算法等的解释
- 还有文档注释/**.....*/，这样的注释可以生成html格式的javadoc文档。
- 而且可用用@关键字 解释 参数 返回值等信息
- Annotation 主要是一些配置信息的标记，编译时会用到它

3JAVA基础-Java语法基础

- 类和对象什么区别

- 类用于创建对象，通过new方法，把java类文件调入内存，创建CLASS实例，调用java类的构造器，构造java对象。其中静态成员变量和静态方法属于类，有且只有一份，而普通成员变量属于对象，随创建而产生。构造函数是特殊的方法，只有在创建java实例时才会被使用。如果什么构造函数都不写则使用默认构造器什么都不做。如果只要写了一个，默认构造器就不会起作用。

3JAVA基础-Java语法基础

Java 中使用继承来重用代码

- java是单继承的，继承其父类之后，可以使用父类所有非私有的变量和方法，同时也可以用重写的方式，改造原有类的方法。从而实现多态

3JAVA基础-Java语法基础

Java 中的多态

- 多态：就是多种形式，多种状态。比如一个父类被多个子类所继承，子类可以重写父类的方法，从而表现出不同的形式或状态。。

3JAVA基础-Java语法基础

Java 中的静态成员

- 静态成员包括：静态成员变量，静态方法，静态代码块。都是在类加载进内存CLASS（反射机制中）的时候创建，并且只有一份，对应的java实例都可以访问它。
- 静态成员变量：因为是各个对应java实例共享的，所以可以用来记录数据，如单例的引用。
- 静态成员函数：也是各个实例可以使用，但是一般通过类名.静态方法。使用
- 静态代码块：一般里面可以使用静态变量，不能直接使用外部实例的成员变量和方法。在类加载的时候就执行。 形式如：static{}

3JAVA基础-Java语法基础

- Java派生类的构造方法和父类传递参数

- 通过super(参数)来调用父类的构造方法，
- 也可以调用可能被子类覆盖了的父类的函数 super.method(参数)
- 如果是调用默认无参的构造函数可以不用写。（父类没有编写任何构造函数）

3JAVA基础-Java语法基础

接口和抽象类的区别

- 接口和抽象类都不能被实例化，接口一定程度上属于抽象类。
- 抽象类里面有方法有属性和抽象方法，只要里面包含一些没有实现的方法只能是抽象类 。关键词abstract class，通过extend继承。继承时必须实现所有抽象方法。
- 接口 里面所有方法都没有被实现，属性必须是静态不可变的 static final 通过implement关键字
- 单继承多实现。

3JAVA基础-Java语法基础

简述内部类

- 内部类分为：1静态内部类 2内部成员类3局部内部类 4匿名内部类
- 1 静态内部类是静态内部成员类，使用起来就像静态内部成员变量，可以和外部类完全看作完全不同的两个类，可以使用外部静态方法（类名访问都是可以的），不能使用外部成员变量。 static class。外部可以直接使用它（就像使用该外部类的静态方法）
- 2成员内部类：相当于普通内部成员，相当于范围缩小了，成员内部类可以访问外部类成员。外部想要使用它 必须先创建外部类对象，再创建内部类对象。内部成员类new （传入外部类对象作为构造器参数o，同时this.o=o）后可以使用其外部类的成员变量。
- 3局部内部类：相当于 方法体中的变量。它只能由abstract 和final 变量和只能访问外部final的变量。外部给内部构造器传入一个对象本身o 主要用于做一些方法操作。。
- 4匿名内部类 和局部内部类一样只是没有名字而已。如方法中直接new fun(){一些方法}

3JAVA基础-Java语法基础

包创建和使用

- package关键词用于定义类所在包的位置，用于解决重名的问题
- 对于不是同包的类需要引入import进来

3JAVA基础-Java语法基础

访问控制符

- public 公开的，可以最大权限被访问，一般用作对外的接口API
- protected 是保护级访问权限。可以被同包下或者继承该类的子类（可以不同包）所访问
- default 什么都不写， 可以被同包下其他类和自己所访问
- private 只能是自己可以访问
- 其实即使是私有的也可以通过反射访问他 通过改变其访问权限设置

3JAVA基础-Java语法基础

- Int和Integer区别

- int是基本数据类型 4个字节，内容放的是值， 保存在栈中，可以进行计算。
- Integer是对象类型 保存的是对象引用，放在堆中，不可以计算，传递参数的时候是引用而不是值本身。

3JAVA基础-Java语法基础

- Int的取值范围

- int占4个字节 一共32bit 第一位表示正负符号，其他为值，用补码表示，
- 从-2^32~~2^32-1

3JAVA基础-Java语法基础

- 八进制和十六进制

- 八进制的0开头， 十六进制的0x开头

3JAVA基础-Java语法基础

- Long取值范围

- long类型的数字占8个字节，64bit 第一位表示正(0)负(1)，其余为数值，也是以补码形式保存的。表示的数从-2^63~2^(63-1) 必须在数后面加 l或 L。否则表示的是int型

3JAVA基础-Java语法基础

- Float和double的取值范围

- float类型 能表示3.4E-38~~3.4E+38 数值后面需要加F 或者f 4字节
- double类型能表示1.7E-308~~1.7E+308 数值后面的D或者d 可加不加，带小数的默认情况就是双进度 8字节

3JAVA基础-Java语法基础

浮点型和整型的转换

- 整型数值转换为浮点型数值可以直接转换，而浮点型数值转换为整型数值需要强制类型转换，而且小数会损失掉

3JAVA基础-Java语法基础

如何用BigDecimal类进行精确运行

- BigDecimal是大的小数的意思。这是个专门用来商业计算的工具（因为普通的浮点数计算，因为计算机实现的过程中会有些微笑误差，使得数字不好看）而Big Decimal 可以解决这个问题。
- 它的使用方法有点像用包装类。
- 1可以通过构造方法（把指定浮点数字 toString()）
- 2通过valueOf（浮点数）其对象方法
- 转变成BIgDecimal
- 用 add() substract() mutiply() divide() 其对象方法 进行计算
- 最后通过 floatValue() 和doubleValue()类方法转变回基本数据类型

3JAVA基础-Java语法基础

Java可以用非0表示ture吗

- 布尔类型的数据 true 和false 两个值，不能像C语言那样用非0 和0来表示
- 所以初始化时就得给其赋值 不能直接用。

3JAVA基础-Java语法基础

- Boolean和他的包装类的区别

- Boolean是boolean的包装类。除了true， false 还多一个null
- 默认值是null 如果不初始化，会通过编译但是运行时会抛出空指针错误。

3JAVA基础-Java语法基础

Char的取值范围

- char是2个字节，用unicode进行编码 可以存储汉字（适应国际化）
- 使用时用单引号 char a ='b';
- 一共可以存储2^16个字
- 本质里面放的是二进制 所以可以和short进行类型转换而不损失

3JAVA基础-Java语法基础

- Char能否存储汉字

- char为2个字节和short相等，int 4个字节，char能存放汉字，汉字两个字节

3JAVA基础-Java语法基础

- 使用转义字符

- \用于表示转义字符，系统遇到反斜杠\(捺)就会紧接着去读后面的内容
- \t 水平制表符 \r 回车 \\输出反斜线 \" 双引号 \'单引号 等

3JAVA基础-Java语法基础

- 字符串自动生成string对象

- “”创建字符串被优化，通过string对象池中是否有对应对象存在来决定是否创建。
- new 方式创建是一定会创建string 对象并把引用赋值出去。

3JAVA基础-Java语法基础

- 字符串对象池的作用是什么

- 对象池是为了避免频繁的创建销毁对象，这里自己创建了一个对象池，通过静态的 newinstance方法创建对象。在list中遍历寻找创建并保存过的对象。

3JAVA基础-Java语法基础

- StringBuffer和StringBuilder的作用

- string在拼接对象过程中会产生许多无用的对象，放入string对象池中，影响性能。StringBuffer通过append动态添加不会过多创建对象，但不是线程安全。stringbuilder是线程安全的

3JAVA基础-Java语法基础

- 如何输出反转过后的字符串

- 反转过过来手动的有两种方法：1 用循环 charAt i递减取出
- 2 转化成char数组然后循环再取出。
- 3 直接用stringbuffer 现成的reverse方法直接取出

3JAVA基础-Java语法基础

- Int的取值范围

- int占4个字节 一共32bit 第一位表示正负符号，其他为值，用补码表示，
- 从-2^32~~2^32-1

3JAVA基础-Java语法基础

- 设置指定字符串来创造String对象

- string对象解决编码问题。可以通过把指定字符集重新创建一次字符串对象。参数一个是二进制（byte）数组，和重构采用的字符集。
- 目的是使指定字符串从乱码变成可读的。

3JAVA基础-Java语法基础

- Java数组是一个java类对象

- 数组是引用类型对象 如 int[] a =new int[]{1，2，3}或者是new int[3]。 数组是一个特殊的类，他的成员变量要通过 [] 来访问。数组在定义时就要给出数组的长度。如果数组成员是基本类型，则存放的是变量值，默认为0，如果是引用类型的则存放的是对象的引用，默认为null

3JAVA基础-Java语法基础

- New Object[5]语句是否创建了5个对象

- 成员为引用类型的数组在创建的时候，只是会创建一个数组对象。其中的成员初值为null，直到到该成员指向某一对象后

3JAVA基础-Java语法基础

数组的拷贝

- 当数组拷贝直接通过=赋值进行的话，拷贝过去的只是对象的引用，被拷贝的和拷贝的对象其实本质是指向同一个对象。当其中一个引用所指的对象修改的时候，另一个引用指的对象将会同时被修改。
- 避免这一问题可以使用System.arraycopy（）方法进行深度拷贝。

3JAVA基础-Java语法基础

二维数组的长度是否固定

- 二位数组长度可以不固定，因为多维数组的机制为，首先创建一个一维数组，（只要这一维度的长度明确了就行），再每个成员指向第二维级的数组，依次类推，第二维的数组长度可以是任意的。第二维的大小并并不需要初始的时候定义。
- 还可以遍历访问 for（int[] a : biArray）for(int i : a)如此般遍历

3JAVA基础-Java I/O操作

- 目录和文件操作

- File用于操作文件，统一操作文件和目录的接口。打开文件的方式是统一的。File file =new File（“C:/text.txt”）然后可以用file.exists()判断文件是否存在。用file.createNewFile()来创建文件或者目录。
- 通过isFile（）来判断是文件吗？isDirectory（）来判断是否目录？用.list()获得目录下文件和目录的列表，.getName（）

3JAVA基础-Java I/O操作

手动复制文件

- 硬复制文件：1首先通过被复制文件的目录和将复制的文件的目录创建输入流和输出流，即 FileInputStream和FileOutputStream 参数就是文件目录
- 2 创建二进制字节数组 Byte{} buff =new Byte[256]
- 3 把文件中二进制内容通过输入流read方法读入内存（二进制字节数组） int len = fileInputStream.read(buff)
- 4 通过输出流的write方法内存（二进制字节数组）写入指定数组下标范围的二进制内容到文件 fileOutputStream.write(buff,0,len)
- 5 最后都读完后关闭输入输出流，否则文件无法再打开或者浪费资源。

3JAVA基础-Java I/O操作

- 随机存取文件RandomAccessFile

- RandomAccessFile类是文件随机访问流，创建时，第一个参数是路径，第二参数可以设置流的读写权限。对单文件进行读写操作。
- .length()可以获取文件的字节数（byte）。seek()可以把读写操作指针指向指定位置之前。如seek（0）默认从0开始，把读写指针放在初始第一个位置之前。
- .read()或者.write()指读or覆盖写一个字节。同时指针也会向后移动一个字节。
- 读写操作完，和其他流一样都要关闭此流。其中读出来的事byte二进制类型的需要进行强制类型转换如（char）

3JAVA基础-Java I/O操作

字符流的处理方式

- 字节流是最基本流，是其他流的基础，操作的对象是byte数组，字节数组。用于不需要考虑具体内容的，但是使用的时候不如其他特定流的方便。
- 具体有 File读写字节流，Object读写字节流，Buffered读写字节流

- Tips：流关闭问题：1流先关闭里面的流再关闭外面的流
- 2如果把外面的关闭了，自动已经把里面的也关闭了

3JAVA基础-Java I/O操作

字符流的处理方式

- 字符流需要建立在字节流的基础上。
- 1 首先创建InputStream 可以通过new FileInputStream（“路径”）从文件中读字节
- 2 然后创建InputStreamReader 第一个参数为输入流，第二参数是字符集类型。主要作用用于指定输入流的格式。（这里read()已经可以是字符了）
- 3然后创建字符流BufferedReader 里面放入 （输入流reader）（bufferedReader多了行读）
- 4 这样可以直接读取字符read(),和readLine（）读取一行
- 5 最后要关闭 字符流 ps：其中 .append(字符串)可以加入StringBuffer。输出 PrintWriter要包装输出流（OutputStream）有println，write和print

3JAVA基础-Java I/O操作

- 序列化

- 序列化：把内存中的object对象首先转换成二进制方式的字节数据，然后放入存储介质中。反序列化就是反过来，先从介质中读出二进制流再转换对象。
- 可序列化的类是实现类了serializable接口。有的有serialVersionUID

3JAVA基础-Java I/O操作

- 序列化和反序列化一个java对象

- 序列化：1首先把ObjectOutputStream参数中包装FileOutputStream。2通过writeObject（对象）读取object对象序列化后写入文件。3用完之后要关闭object流。
- 反序列化：1ObjectInputStream包装FileInputStream 从文件中读取对象，然后强制类型转换成特定类型，用完也要关闭object流。

3JAVA基础-Java 多线程

什么是多线程

- 多线程机制在java中可以不受操作系统影响，程序员决定线程的调度。
- 线程是独立运行的指令集，在进程中共享资源和内存，可以提高效率
- 进程是操作系统负责调度的。拥有独立代码，内存等资源和cpu程序员不能控制

3JAVA基础-Java 多线程

- 进程和线程的区别

- 1线程比较小，进程比较大，进程包含线程
- 2操作系统直接控制进程，线程则是由程序控制
- 3进程通信比较麻烦，线程共享资源通信容易
- 4进程独立使用资源，线程间共享进程中的资源

3JAVA基础-Java 多线程

- 如何编写线程类

- 一个线程其实就是一个类的run()方法的执行过程
- 有两种使用线程的方式：extend Tread类 和implement Runable接口
- 1Tread类 中帮我们实现多个方法并包括实现了了run()方法，但是实际上是空的，需要自己重写进去
- 2 Runable接口 ： 普通类可以通过实现该接口 通过实现run（）方法。来实现线程

3JAVA基础-Java 多线程

- Runnable接口与Thread类的区别

- 1在创建多线程后，实现Runable接口可以更方便访问线程类中同一变量
- 而集成Tread类 则需要在内部创建内部类来访问实现访问同一变量。
- 2而Tread类同时通过一些方便的方法 比如获取进程ID 进程名和进程状态等
- 但Runable接口关于这些方法都要自己实现

3JAVA基础-Java 多线程

- 如何启动线程

- 1如果是继承Thread类，直接用new 创建即可
- Thread t1=new ThreadTest();
- 2如果是实现Runable接口，则需要把实现Runable接口的类所创建的实例作为参数传给Thread，作为构造函数的参数。
- Thread t2=new Thread（new RunTest()）；
- 然后通过start() 方法启动 t1.start(); t2.start();

3JAVA基础-Java 多线程

- 如何使用sychronized来让线程同步

- 可以通过synchronized关键字开进行通过，从而实现线程安全。其实就是所谓的线程锁的概念。
- 1可以加在代码块上面，通过一个静态对象（object）的引用作为锁。public static Object obj=new Object();
  synchronized(object){代码执行块}
- 线程进入代码先看看这个对象是否被占用，如果被占用则等待（占用对象的代码执行完毕后会释放对象，等其释放了，该线程才可进入），
- 2 也可以直接把锁加在方法，同步方法
- public synchronized void test(){
- 方法体}

3JAVA基础-Java 多线程

- 生产者与消费者模型的多线程例子程序

- 线程通过主要通过wait()(等待)notify（）notifyAll（）唤醒和唤醒全部
- 这三个同步方法外面需要加 synchronized关键字。否则会报错。
- 线程对象.setName(线程名字符串);可以设置该Thread子类的名字 Thread.currentThread().toString()可以打印出来。
- 算法java实现：首先定义一个Store类，里面有add 和remove 方法用来供生产者和消费者存取（包含同步控制的等待和唤醒）还有int类型的count计数。然后定义生产者和消费者继承Thread类 说明可以按线程单独运行。里面有个引用指向共同的Store对象。然后大家通过操作这个唯一的Store对象进行存取控制。

3JAVA基础-Java 多线程

- 如何使用java的线程池

- ThreadPoolExecutor（int corePoolSize，int maximumPoolSize,long keepAliveTime,TimeUnit unit,BlockingQueue<Runable> workQueue,RejectedExecutionHandler handler）线程池
- int corePoolSize标准的线程数
- int maximumPoolSize最大线程数
- long keepAliveTime线程保持活动的时间
- TimeUnit unit时间的单位BlockingQueue<Runable> workQueue等待线程队列 RejectedExecutionHandler handler当等待队列满，线程达到最大数的处理方式
- 通过ThreadPoolExecutor对象的 .execute(线程)调入并启用线程。当线程数过少，没达到标准线程数时，来一个创建一个线程。当达到标准时 再进来就得进阻塞队列里面等着。如果阻塞队列满了，再创建线程直到到达最大线程数。这时候再来线程就用对应策略进行替换或拒绝等处理。
- 使用线程池的好处是使得业务代码和线程处理代码分离。

3JAVA基础-Java 反射机制

- 反射基础

- 反射（java实例反而通过CLASS实例映射出该类的信息）就是指可以在程序运行状态下可以动态调用指定java类的类名，方法和构造函数等功能。主要是通过一个Class类（即所谓的类的类）在java类NEW出实例时，需要先创建一个包含该java类信息的Class类的实例。通过这个实例可以获得很多关于该java类的一些属性 方法等信息。通过这些信息可以直接构造出该java类的任何对象。spring 等框架都用反射

3JAVA基础-Java 反射机制

- Class类的含义和作用

- 在下列三种情况会创建CLASS对象并把具体的java类加载到JVM：
- 1new出新的java对象时。2访问java类的静态成员变量时。3访问 Class.forName("类路径")方法
- 通过如下三种方法可以获取特定的Class类对象
- 1 通过Class类属性， Class.forName("com.test.Student")
- 2 通过JAVA类的静态属性 Class<Student> clazz=Student.class
- 3通过java对象的方法如new出student实例后Class<Student>clazz= stu.getClass()。 Class类支持泛型

3JAVA基础-Java 反射机制

- 操作类的成员变量（Field）

- 通过反射机制，可以定义一般化的比较方法而不用涉及到具体实例
- Field对象 是通过Class类对象的getDeclaredField（属性字段）方法和getDeclaredFields（）方法获取Field对象。Field指 某类的某一个字段。而不涉及具体java实例。
- 然后 用field.get("java实例") 获取该java对象这一对应字段的值，再强制类型转换出来。当然也可以set方法 把值放进该java对象中。

3JAVA基础-Java 反射机制

操作类的方法（Method）

- Method反射对象是通过method.invoke（java类实例，object类型参数1，object类型参数2，....）方法调用。
- 首先Method m2=Class类的.getDeclaredMethod(方法名)
- 然后通过设置参数（java类的实例）用invoke调用方法

3JAVA基础-Java 反射机制

- 通过反射实例化一个类

- 利用反射实例化对象涉及到两种情况：
- 1无参的时候：只用通过类名创建类对象，再通过.newInstance()方法创建java对象。Class<edu.Student> clazz =Student.class or Class.forName("edu.Student") or stu.getClass() 然后clazz.newInstance();
- 2 有参的时候：1创建类对象
- 2然后创建构造器Constructor<Student> con=clazz.getConstructor(String.class,int.class)
- 3通过参数和构造器创建对象Student stu2=con.newInstance("LiLi","18");

3JAVA基础-Java 反射机制

- 通过反射访问类私有成员

- field。setAcesssable(true)可以让私有成员变量变得可访问

3JAVA基础-Java 反射机制

- 反射来覆盖数据对象的toString（）方法
- 通过 clazz.getDeclaredFields获取所有字段的数组给Fields[] 然后覆盖toString（）方法改变方法输出

3JAVA基础-Java 网络编程

TCP/IP协议

- TCP/IP协议目前是最流行的网络协议。一共有4层，不同于ISO定义的7层结构。该协议包括TCP，UDP，IP，ICMP。
- 应用层：程序员操作，有http协议 ftp，smtp，telnet
- 传输层：端到端，传输控制
- 互联网络层：主机到主机
- 网络接口层：帧传输的具体物理特性
- 程序员操作一般在应用层。大多在java,net包下，包括ServerSocket 和Socket 和URL

3JAVA基础-Java 网络编程

TCP/IP协议通信特点

- TCP协议是可靠传输，按字节传输，端到端，面向连接。
- 使用它作传输层协议的应用层协议有 http 80，telnet 23，ftp 21， smtp 25
- 端口号从0~6万5千多，2个字节 16bit大小。 默认端口是可重新定义的

3JAVA基础-Java 网络编程

- TCP/IP编程模型

- TCP下的Socket编程：服务器端：1先创建SocketServer（定义端口号），2然后调用其.accept()方法给一个Socket对象。3Socket可以获取字节输入流或字节输出流getInputStream()和getOutputStream()。4然后可以用InputStreamReader与BufferedReader包装，然后就可以readline（）或者输出的话用PrintWriter包装，然后print(内容)，flush（）刷新缓冲区中的流。最后用完要分别关闭输入输出流，关闭Socket，关闭ServerSocket对象。

3JAVA基础-Java 网络编程

- UDP协议

- UDP 不提供可靠连接，没有可靠传输机制。传输时的报文头比较小。应用它的应用层协议：DHCP DNS TFTP 还有视频传输等

3JAVA基础-Java 网络编程

- UDP编程模型

- 接受端：1首先创建数据报文套接字DatagramSocket ds =new DatagramSocket(9999);
- 2创建一个缓冲区Byte[] buff=new Byte[1024];
- 3创建一个数据包用于接受数据包DatagramPacket dp=new DatagramPacket(buff，1024)
- 4用数据报文套接字把数据接到数据包中 ds.receive(dp);
- 5读取数据包中的数据String str=new String(dp.getData(),0,dp.getlength())
- 6最后关闭套接字 ds.close();

3JAVA基础-Java 网络编程

UDP编程模型（发送端）

- 发送端：1首先创建数据报文套接字提供端口)DatagramSocket ds=new DatagramSocket(9998);
- 2 然后使用创建字符串 String str ="data";
- 3 把数据添加到数据包中 DatagramPacket dp=new DatagramPacket(str.getByte(),0,str.length(),InetAddress.getByName("localhost"),9999);
- 4.ds.send(dp);
- 5.ds.close();

3JAVA基础-Java 网络编程

创建Web服务器

- 服务器和客户端的实现和前面定义相似。只不过这里是使用http协议，传输层仍然是通过tcp实现的。只不过服务器端要监听80接口（http专用）。服务器端写入的是一个（html语言的文本）。这样浏览器就可以键入地址之后能正确显示出html

3JAVA基础-Java 网络编程

- 使用java访问web站点

- URL是包装了TCP 可以进行http访问的一个类。
- 1首先，URL url =new URL("htttp:\\www.baidu.com");
- 2获取连接HttpURLConnetion con= (HttpURLConnection)url.openConnection();
- 3打开连接 con.connect();
- 4操作连接如 con.getHeaderFields()获取http头 con.getInputStream获取HTML
- 5断开连接 con.disconnect();
- 还可以通过设置属性，配置代理服务器

3JAVA基础-对数据库的操作

- SQL

- SQL语句是structured Query Language 用于从数据库中查询数据，几乎所有的DBMS都支持它。基本上格式就是 干什么， 从哪来 ， 怎么干

3JAVA基础-对数据库的操作

- SQL检索数据

- select name，age
- from student
- where age>20 and age<100
- order by name <desc/asc>

3JAVA基础-对数据库的操作

SQL更改数据

- 三种对数据库中数据进行修改的操作
- 1 insert into user(name，age) values ("zhangsan",20)
- insert into usercopy select * from user
- 2 update user set(age=80) where name="zhangsan"
- 3 delete from user where name="zhangsan

3JAVA基础-对数据库的操作

JDBC工作原理

- JDBC是提供给用户操作数据用到的API，用面向对象的思想提供接口和方法。
- 接口的另外一头是SPI 由各个数据库厂商实现，具体细节千差万别。程序员只需把对应的数据库driver 的jar包导入 classpath or web-inf/lib下面就可直接用

3JAVA基础-对数据库的操作

- JDBC操作数据库的编程步骤

- 用JDBC进行数据库访问一般分为6个步骤：
- 1注册驱动器 Class.forName("驱动器的路径")
- 2创建一个连接 Connection con =DriverManager,getConnection("URL","username","password")
- 3 在连接上创建 会话 Statement stmt=con.createStatement();
- 4 在会话上执行查询语句并返回到解决集ResultSet rs= stmt.executeQuery("sql语句");
- 5 结果及上操作 rs.next() 6 依次关闭结果集 会话 和连接

3JAVA基础-对数据库的操作

JDBC实现事务

- 事务性要求有4点：原子性，一致性，隔离性，持久性
- JDBC的事务处理的方式：
- 1首先关闭JDBCsql的自动的事务提交功能。con.setAutoCommit（false）
- 2 设置Try/catch 保证出错时能回滚
- try{创建statment；stat.executeQuery("sql");con.commit()}
- catch(Exception e){con.rollback()}
- finally{如果未关闭则stat.close();con.close();}

3JAVA基础-对数据库的操作

- JDBC实现数据访问对象层（DAO）

- JDBC实现DAO层：目的是提供一个接口下面是用sql访问数据库。上层是展现一个面向对象的接口供service层使用。一般至少实现五种方法：1通过id查询 返回装载数据后的java类对象。2通过批量查询 返回java类对象放入list中。3以java类对象作为参数存入数据 4 以java类对象作为参数修改数据 5用id来删除数据。

3JAVA基础-对数据库的操作

使用连接池技术

- 对于数据操作比较多的应用时，可以考虑使用连接池技术。与JDBC不同的是：不是通过DriverManager对象创建连接，而是通过提供JNDI的 Data source对象创建连接，close（）不是真正关闭连接而是回收连接。需要设置最大活动连接，最大空闲连接和连接超时时间。较常用
- InitialContext ict=new InitialContext（）；
- DataSource ds =（DataSource）ict.lookup(“JNDI路径”);
- Connection con=ds.getConnection();

3JAVA基础-对数据库的操作

- 使用可滚动的结果集

- 通过Statement和PrepareStatement使用三参数 分别设置 sql语句，type，是否可更新。
- 可滚动主要是通过type来设置：参数有 1ResultSet.TYPE_FORWARD_ONLY 不可滚动；
- 2 .TYPE_SCROLL_INSENSITIVE 可滚动，但是数据库变化，结果不变化。
- 3 .TYPE_SCROLL_SENSITIVE可滚动，数据库变化它也有反应。
- 操作时如 rs.last(), rs,previous() rs.next()

3JAVA基础-对数据库的操作

使用可更新的结果集

- 通过 Statement或PrepareStatement第三个参数设置可否更新concurrency：CONCUR_READ-ONLY只可读但不可更新。CONCUR_UPDATABLE可更新
- rs.updataString("2","aaaa")
- rs.updateRow()

3JAVA基础-对数据库的操作

JDBC操作Oracle数据库

- Oracle的jdbc url：jdbc:oracle:thin:@localhost:1521:tiger

- Class.forName("oracle.jdbc.driver.OracleDriver");
- Connection con=DriverManager.getConnection("jdbc.oracle:thin:localhost:1521:tiger");
- Statement stat=con.createStatment();
- ResultSet rs=stat.executeQuery("select * from user")
- if(rs.next()){System.out.println(rs.getString("name"));}
- rs.close();stat.close();con.close();

3JAVA基础-对数据库的操作

- JDBC操作MySQL数据库

- MySQL现在是Sun下面的一个开源数据库 只有一种连接方式
- jdbc:mysql://localhost:3306/test

3JAVA基础-对数据库的操作

- JDBC操作SQL Server数据库

- jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=Northwind

3JAVA基础-对数据库的操作

- JDBC操作Access或Excel
- 桥接方式：要配置ODBC 然后 jdbc:odbc:test

Thanks!