# 什么是进程？什么是线程？为什么要使用多线程？

2018年4月3日 21:33

（1）线程是指程序正在执行过程中，能够执行程序代码的一个执行单元，在JAVA中，线程有4中状态：运行、就绪、挂起和结束。

（2）进程是指一段正在执行的程序。一个进程可以拥有多个线程，各个线程之间共享数据的内存空间（如代码段、数据段和堆空间）以及一些进程级的资源，但是各个线程拥有自己的栈空间。

（3）线程有时被称为是轻量级进程，它是程序执行的最小单元。

（4）在操作系统级别上，程序的执行都是以进程为单位的，每个进程通常都会有多个线程互不影响的并发执行

2.为什么使用多线程？

（1）使用多线程可以减少程序的响应时间

（2）与进程相比，线程的创建和切换开销更小

（3）多CPU或多核计算机本身就具有多线程的能力，如果使用单线程将造成一种资源的浪费。

（4）使用多线程能够简化程序的结构，使程序便于理解和维护。一个非常复杂的进程可以分成多个线程来执行

# 同步和异步

2018年4月3日 21:46

1.（1）在多线程的环境中，当遇到数据的共享问题时，即当多个线程需要访问同一个资源时，它们需要以某种顺序来确保该资源在某一时刻只能被一个线程使用，否则，程序的运行结果将会是不可预料的，在这种情况下就必须对数据进行同步。
（2）要想实现同步操作，必须要获得每一个线程对象的锁，这样可以保证在同一时刻只有一个线程能够进入临界区（访问互斥资源的代码块），并且在这个锁被释放前，其他线程就不能再进入临界区（访问互斥资源的代码块），并且在这个锁被释放之前，其他线程不能再进入这个临界区。如果还有其他线程想要获得对象的锁，只能进入等待队列等待。只有当拥有该对象的锁的线程退出临界区时，锁才会被释放，等待队列中优先级最高的线程才能获得该锁，从而进入共享代码区。
2.（3）JAVA中通过synchronized关键字来实现同步操作，但这种方法的代价是系统开销很大，有可能会造成死锁。所以同步控制不是越多越好，要尽量避免无谓的同步控制。
实现同步的方式有两种：①利用同步代码块来实现同步②利用同步方法来实现同步
3.异步
异步过程每个线程都包含了运行时自身所需要的数据或方法，因此，在进行输入输出处理时，不必关心其他线程的状态或行为，也不必等到输入输出处理完毕才返回。如果当应用程序在对象上调用一个需要花费很长时间来执行的方法并且不希望让程序等待方法返回时，就应该使用异步编程，异步能够提高程序的效率。

# 如何实现java中的多线程

2018年4月3日　　22:15

java虚拟机允许应用程序并发的运行多个线程，在JAVA语言中，多线程的实现一般有3种方法，其中前两种是最常用的方法。

1.（1）继承Thread类，重写run()方法

Thread本质上也是实现了Runnable接口的一个实例，这个类代表线程的一个实例。**启动线程的唯一方法是通过Thread类的start()方法。start（）方法是一个本地方法，它将启动一个线程，并执行run（）方法（Thread中提供的run（）方法是一个空方法），这种方法通过自定义直接extend Thread,并重写run（）方法，就可以启动新线程并执行自己定义的run（）方法；但是，调用start（）方法后并不是立即执行多线程代码，而是使得该线程变为Runnable（可运行状态），什么时候运行多线程由操作系统决定的。**

```
public void run(){
    System.out.println("Thread body");//线程的主体
}
public class Test{
    public static void main(String[] args){
        MyThread  thread=new MyThread();
        Thread.start();//开启线程
    }
}
```

**（2）实现Runnable接口，实现run（）方法**

**①自定义类并实现Runnable接口，实现run()方法**

**②创建Thread类，用实现Runnable接口的对象作为参数实例化该Thread对象**

**③调用Thread的start（）方法**

```
class MyThread implements Runnable{
    Public void run(){
        System.out.println("Thread body");
    }
}
public class Test{
    public static static main(String[] args){
        MyThread thread=new MyThread();
        Thread t=new Thread(thread);
        t.start();
```

```
    }
}
```

**2.一个类是否可以同时继承Thread类与实现Runnable接口**

**可以。**

```
Public class Test extends Thread implements Runnable{
    Public static void main(String[] args){
        Thread t=new Thread(new Test());
        t.start();
    }
}
```

# Run()和start()方法

2018年4月4日 13:00

系统通过调用线程类的start（ ）方法来启动一个线程，此时该线程处于就绪状态，而非运行状态，也就意味着这个线程可以被JVM来调度执行。在调度过程中，IVM通过线程类的run()方法来完成实际的操作，当run()方法结束后，此线程就会中止；如果直接调用线程类的run（ ）方法，这会被当做一个普通函数调用，程序中仍然只有主线程这一个线程，start（ ）方法能够异步调用run（ ）方法；直接调用run（ ）方法是同步的，因此也就无法达到多线程的目的。

只有通过调用线程类的start（ ）方法才能真正达到多线程的目的。

# 多线程同步的实现方法

2018年4月4日 13:27

1.JAVA中实现同步机制的方法：

当使用多线程访问同一个资源时，非常容易出现线程安全的问题。因此需要采用同步机制来解决这种问题。JAVA主要提供了3种实现同步机制的方法：

（1）synchronized关键字

在Java语言中，每个对象都有一个对象锁与之相关联，该锁表明对象在任何时候都只允许被一个线程拥有。当一个线程调用对象的一段synchronized代码时，需要先获取这个锁，然后去执行相应的代码，执行结束，释放锁。synchronied关键字主要有两种用法（synchronized方法和synchronized块），此外该关键字还可以作用于静态方法、类或某个实例变量。

①synchronized方法，在方法的声明前加入synchronized关键字

Public synchronized void mutiThreadAccess();

②synchronized块。synchronized块既可以把任意的代码段声明为synchronized，也可以指定上锁的对象，有非常高的灵活性。

（2）wait（）方法和notify（）方法

当使用synchronized来修饰的某个共享资源时，如果线程A1来执行synchronized代码，另外一个线程A2也要同时执行同一对象的同一synchronized代码时，线程A2将要等到线程A1执行完成后，才能继续执行。这种情况下可以使用wait（）方法和notify（）方法。在synchronized代码被执行期间，线程可以调用的wait（）方法，释放对象锁，进行等待状态，并且可以调用notify（）或者notifyAll（）方法通知正在等待的其它线程。notify（）方法仅唤醒一个线程（）（等待队列的第一个线程）并允许它去获得锁，notifyAll（）方法唤醒所有等待这个对象的线程并允许它们去获得锁（并不是让所有的唤醒线程都获得到锁，而是让她们去竞争）

（3）Lock接口也可以实现线程的同步

Lock接口提供了方法实现多线程的同步：

2.synchronized与Lock有什么异同

**Java语言提供了两种锁机制来实现对某个共享资源的同步：synchronized和Lock。**

其中synchronized使用Object对象本身的notify、wait、notifyAll调度机制；Lock使用Condition进行线程之间的调度，完成synchronized实现的所有功能。两者的区别：

①用法不一样

②性能不一样

③锁机制不一样

# Collection

2018年5月3日 9:28

```
Collection
    -----List
            -----LinkedList   非同步
            ----ArrayList     非同步，实现了可变大小的元素数组
            ----Vector        同步
                  ------Stack
    -----Set   不允许有相同的元素
Map
    -----HashTable      同步，实现一个key--value映射的哈希表
    -----HashMap        非同步，
    -----WeakHashMap   改进的HashMap，实现了“弱引用”，如果一个key不被引用，则被GC回收
```

List和Set都继承了Collection接口，而Map是和Collection同一等级的接口。

（1）List的子类有ArrayList、LinkedList和Vector；LinkList和Vector的底层都是基于链表，而ArrayList的底层是基于线性表。LinkList和ArrayList的共同点是线程是不安全的；不同点是ArrayList在做增删元素的时候，效率高，查询效率低；而LinkList恰好相反，这是由它们的底层决定的。

（2）Set的实现类有HashSet、TreeSet。这两个类都保存了元素的唯一性，但是HashSet的元素是无序的，而TreeSet的元素是有序的。

（3）Map的实现类：HashMap和HashTable，

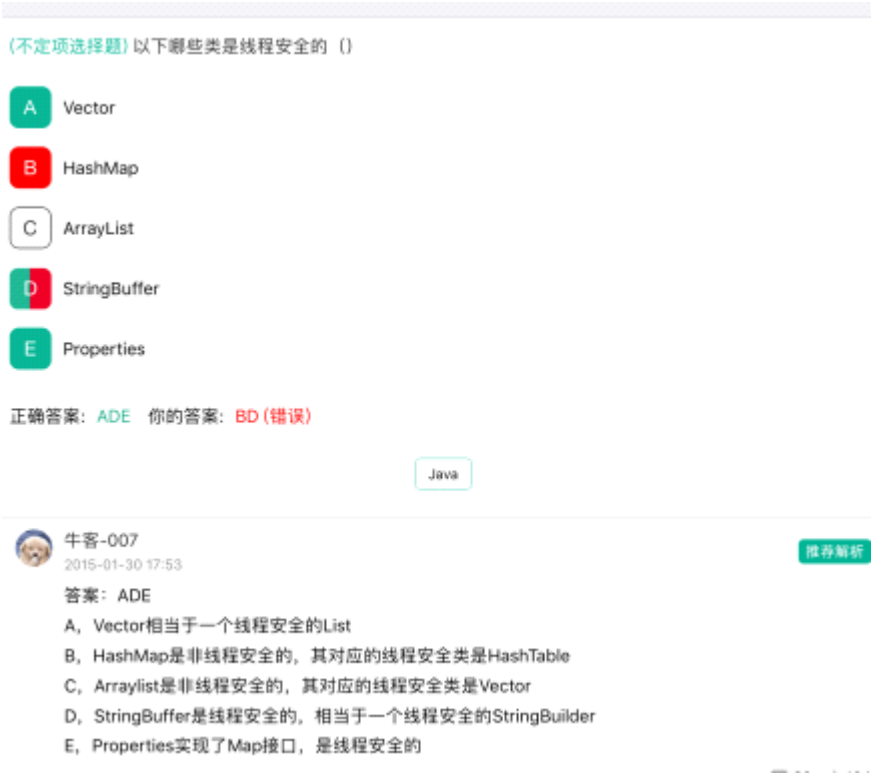
**①HashMap几乎可以等价于Hashtable，除了HashMap是非synchronized的，并可以接受null(HashMap可以接受为null的键值(key)和值(value)，而Hashtable则不行)。**

②HashMap是非synchronized，而Hashtable是synchronized，这意味着Hashtable是线程安全的，多个线程可以共享一个Hashtable；而如果没有正确的同步的话，多个线程是不能共享HashMap的。Java 5提供了ConcurrentHashMap，它是HashTable的替代，比HashTable的扩展性更好。 另一个区别是HashMap的迭代器(Iterator)是fail-fast迭代器，而Hashtable的enumerator迭代器不是fail-fast的。所以当有其它线程改变了HashMap的结构（增加或者移除元素），将会抛ConcurrentModificationException，但迭代器本身的remove()方法移除元素则不会抛出ConcurrentModificationException异常。但这并不是一个一定发生的行为，要看JVM。这条同样也是Enumeration和Iterator的区别。 由于Hashtable是线程安全的也是synchronized，所以在单线程环境下它比HashMap要慢。如果你不需要同步，只需要单一线程，那么使用HashMap性能要好过Hashtable。 HashMap不能保证随着时间的推移Map中的元素次序是不变的。

# 线程安全

2018年5月3日 9:46

线程安全就是指多线程访问时，采用了加锁机制，当一个线程访问该类的某个数据时，进行保护，其他线程不能进行访问直到该线程读取完，其他线程才可以使用，不会出现数据不一致或者数据污染。线程不安全是不提供数据访问保护，有可能出现多个线程先后更改数据造成所得到 的数据是**脏数据**。



# 字符串和子串

2018年5月3日　　12:47

1.子串是指串中任意个连续的字符组成的子序列。

2.模式匹配：是数据结构中字符串的一种基本运算，给定一个子串，要求在某个字符串.中找出与该子串相同的所有的子串。

3.

部分匹配表：

"前缀"指除了最后一个字符以外，一个字符串的全部头部组合；"后缀"指除了第一个字符以外，一个字符串的全部尾部组合；

"部分匹配值"就是"前缀"和"后缀"的最长的共有元素的长度。以"ABCDABD"为例，

- "A"的前缀和后缀都为空集，共有元素的长度为0；
- "AB"的前缀为[A]，后缀为[B]，共有元素的长度为0；
- "ABC"的前缀为[A, AB]，后缀为[BC, C]，共有元素的长度0；
- "ABCD"的前缀为[A, AB, ABC]，后缀为[BCD, CD, D]，共有元素的长度为0；
- "ABCDA"的前缀为[A, AB, ABC, ABCD]，后缀为[BCDA, CDA, DA, A]，共有元素为"A"，长度为1；
- "ABCDAB"的前缀为[A, AB, ABC, ABCD, ABCDA]，后缀为[BCDAB, CDAB, DAB, AB, B]，共有元素为"AB"，长度为2；
- "ABCDABD"的前缀为[A, AB, ABC, ABCD, ABCDA, ABCDAB]，后缀为[BCDABD, CDABD, DABD, ABD, BD, D]，共有元素的长度为0。

| 搜索词 | A | B | C | D | A | B | D |
|---|---|---|---|---|---|---|---|
| 部分匹配值 | 0 | 0 | 0 | 0 | 1 | 2 | 0 |

kmp算法：

1 后移，直到字符串有一个字符，与搜索词的第一个字符相同为止.

2 接着比较字符串和搜索词的下一个字符，直到字符串有一个字符，与搜索词对应的字符不相同为止

3 根据部分匹配表，移动位数 = 已匹配的字符数 - 对应的部分匹配值

4 逐位比较，直到搜索词的最后一位

4.多型数据类型：就是数据元素的类型不确定

5.KMP算法是字符串模式匹配得一个改进的算法，此算法可以在O（m+n）的时间量级上完成串的模式匹配操作，主要功能是求next[]数组。算法的思想是，当主串中的第i个字符与模式中的第j个字符“失配”（即比较不相等时）

(单选题) 用二进制来编码字符串“abcdabaa”，需要能够根据编码，解码回原来的字符串，最少需要()长的二进制字符串?

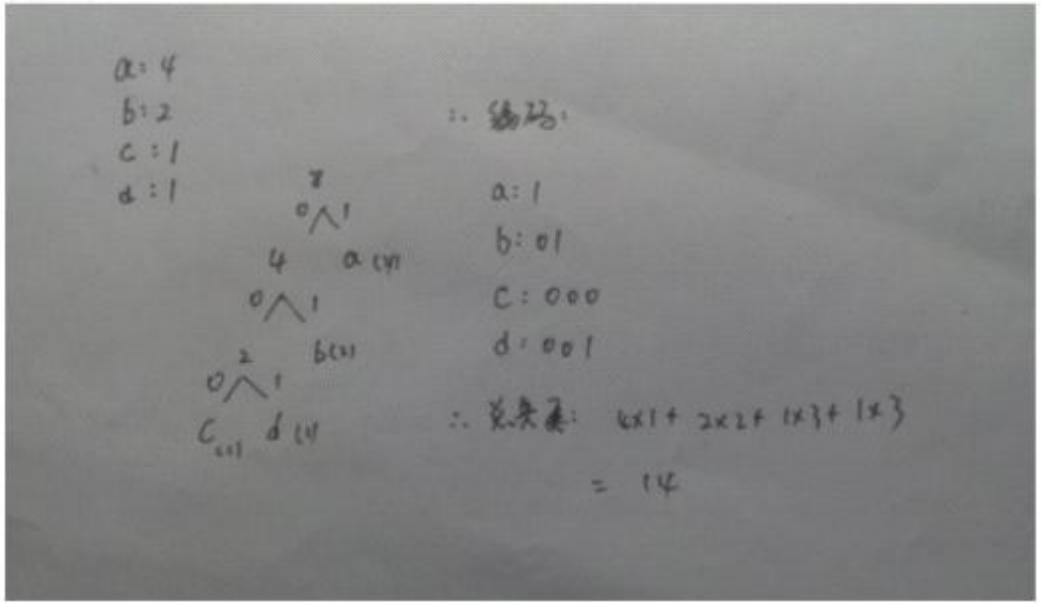
A 12

B 14

C 18

D 24

正确答案: B　　你的答案: C (错误)



# 接口的实现

(单选题) 欲构造ArrayList类的一个实例，此类实现了List接口，下列哪个方法是正确的?

A ArrayList myList=new Object ()

B List myList=new ArrayList()

C ArrayList myList=new List()

D List myList=new List()

正确答案: B 你的答案: C (错误)

Java可以是面向接口的继承，这是Java的一大优点，ArrayList myList=new ArrayList（ ）虽然也可以生成一个myList对象，而且编译也不会报错，但是实际开发中不采用这种提示，提倡使用 **接口名 xxx=new 接口实现类（ ）。**

**选项A错误，因为它只是实例化了一个ArrayList类的对象，但是并没有实现接口；选项B正确，因为它使用的方式就是我们所提倡的方式；选项C错误，因为List是一个接口，不能使用new关键字来实例化；选项D错误，因为接口不能实例化一个借口**

5 根据以下接口和类的定义，要使代码没有语法错误，则类Hero中应该定义方法(　)。

```
interface Action{
    void fly();
}
class Hero implements Action{ //..... }
```

正确答案: D　你的答案: A (错误)

```
private void fly(){}
```

```
void fly(){}
```

```
protected void fly(){}
```

```
public void fly(){}
```

**接口方法默认是public abstract，且实现接口的类的方法的可见性不能小于接口方法的可见性**

# 形参和实参（堆内存和栈内存）

2018年5月3日 19:07

(单选题) 以下代码执行后输出结果为（）

```
public class ClassTest{
     String str = new String("hello");
     char[] ch = {'a','b','c'};
     public void fun(String str, char ch[]){
     str="world";
     ch[0]='d';
 }
 public static void main(String[] args) {
     ClassTest test1 = new ClassTest();
     test1.fun(test1.str,test1.ch);
     System.out.print(test1.str + " and ");
     System.out.print(test1.ch);
     }
 }
```

A hello and dbc

B world and abc

C hello and abc

D world and dbc

正确答案：A 你的答案：D (错误)

学翼无涯
2017-10-13 23:36

这道题里方法中修改值都是修改形参的值，区别是一个通过形参修改堆值，而另一个仅仅只是修改形参。

详细分析一下：

char[] ch = {'a','b','c'};是数组，数组存放在堆中，所以当方法通过形参修改值时会去堆中修改。当成员变量ch再去访问时，堆中的值已经修改，所以输出dbc。

而String str = new String("hello");会进行两步操作：

1、先在堆中创建一个指定的对象"hello"，并让str引用指向该对象。

2、在常量池中 寻找或新建 一个”hello”，并让堆中对象与之关联。

所以当方法为形参赋值时（str="world";），只是为形参在常量池中新建一个”world”并引用，也只修改了形参，成员变量str本身未被修改。

所以，当new了除String以外的对象时，即便通过形参修改值也会去堆中修改。

所以，当String str = new String("hello");时产生了1-2个对象。

Java的栈、堆、常量池和方法区。

在jdk1.7已经将运行时常量池放入堆内存中了。栈存放引用；堆存放new对象和数组；常量池存放常量。

方法区——方法中的局部变量，存放在方法运行时临时建立的方法栈中，其随着栈的销毁而结束

0 1

ど゛低调、
2017-08-30 00:02

java中都是按栈中的值传递，基本数据类型栈中的值就是实际存储的值，引用类型栈中的值就是指向堆中的地址

1）String和char[ ]都是引用类型，所以在方法中传递的都是指向真实数据的地址

2）假设String str指向的hello的地址为d1，str传递到fun函数中的也是地址d1，成员变量str和fun的形参str不是同一个变量，把fun型中的str赋值为world只是修改了该str指向的地址，该地址由d1更改成了world的地址，并没有改变成员变量str指向的地址及堆中的数据，所以str还是hello。

3）假设char[ ] ch指向的abc的地址是d2，传递到fun函数中的地址也是d2，同上成员变量ch和fun的形参ch不是同一个变量，(1)如果把fun中的ch[0]='d'更改为ch = new ch[3]；ch[0]='d',那么成员变量ch的值是没有变化的，还是abc,原理同上String，只是改变了引用ch指向的堆数据的地址，并没有改变成员变量ch指向的地址以及堆中的数据。（2）改变了堆中的数据，所以最终结果编程dbc，此ch只是形参而不是成成员变量ch，如果对ch变化对成员变量ch没有影响，但是ch[i]指向了堆数据的地址，直接修改堆数据，所以成员变量变了

# ArrayList

2018年5月3日 19:30

3 以下代码对其执行后，NumberList里的元素依次为：

```
List<Integer> NumberList =newArrayList<Integer>();
NumberList.add(2);
NumberList.add(4);
NumberList.add(1);
NumberList.add(3);
NumberList.add(5);
for(int i =0;i<NumberList.size();++i)
{
int v = NumberList.get(i);
if(v%2==0)
{
  NumberList.remove(v);
}
}
System.out.println(NumberList);
```

# 日志的级别大小

2018年5月3日 19:31

下列语句哪一个是不正确的（ ）

A. Log4j支持按分钟为间隔生成新的日志文件

B. Log4j是一个打印日志用的组件

C. Log4j支持按年为间隔生成新的日志文件

D. Log4j的日志打印级别可以在运行时重新设置

日志的级别大小有：ALL<TRACE<DEBUG<INFO<WARN<ERROR<FATAL<OFF。其中Log4j建议使用四个级别，优先级从高到低分别是ERROR>WARN>INFO>DEBUG,**注意Log4j在运行期间是不可以进行重新设置的。**

# 类

2018年5月3日 19:41

9 要导入java/awt/event下面的所有类，叙述正确的是？()

正确答案: C 你的答案: A (错误)

import java.awt.*和import java.awt.event.*都可以

只能是import java.awt.*

只能是import java.awt.event.*

import java.awt.*和import java.awt.event.*都不可以

# HttpSeverletRequest

2018年5月3日 19:42

1.HttpSeverletRequest类主要用来处理：①读取和写入HTTP头标②取得和设置cookies③取得路径信息④标志Http会话

# 访问权限

2018年5月3日 19:45

访问修饰符权限如下所示：

类成员访问修饰符与访问能力之间的关系（类成员访问权限）

| 类型 | Private | 无修饰（friendly） | Protected | Public |
|---|---|---|---|---|
| 同一类 | 可访问 | 可访问 | 可访问 | 可访问 |
| 同一包中的子类 | 不可访问 | 可访问 | 可访问 | 可访问 |
| 同一包中的的非子类 | 不可访问 | 可访问 | 可访问 | 可访问 |
| 不同包中 的子类 | 不可访问 | 不可访问 | 可访问 | 可访问 |
| 不同包中的非子类 | 不可访问 | 不可访问 | 不可访问 | 可访问 |

# 循环

2018年5月3日 19:46

18 查看解析

```
public class Test
{
static boolean foo(char c)
{
   System.out.print(c);
   return true;
}
public static void main( String[] argv )
{
   int i = 0;
   for ( foo('A'); foo('B') && (i < 2); foo('C'))
   {
      i++ ;
      foo('D');
   }
}
}
```

What is the result?

```
for(条件1;条件2;条件3) {
   //语句
}
```

执行顺序是条件1->条件2->语句->条件3->条件2->语句->条件3->条件2.....

如果条件2为true，则一直执行。如果条件2位false，则for循环结束

编辑于 2015-09-17 08:07:34

则输出为“ABDCBDCB”

# Spring

2018年5月3日 19:51

Spring Framework是一个开源的Java/Java EE全功能栈（full-stack）的应用程序框架，spring中包含的关键特性：
①强大的基于JavaBeans的采用控制翻转的（Inversion of Control，IOC）原则的配置管理，使得应用程序的组件更加快捷简易。
②它是一个可用于从applet到Java EE等不同运行环境的核心Bean工厂。数据库事物的一般化抽象层，允许声明式（Declarative）事务管理器，简化事务的划分使之与底层无关。
③内建的针对JTA和单个JDBC数据源的一般化策略，是Spring的事务支持不要求Java EE环境，这与一般的JTA或者EJB CMT相反。
④JDBC抽象层提供了有针对性的异常等级（不再从SQL异常中提取原始代码），简化错误处理，大大减少了程序员的编码量。再次利用JDBC时，无需再写出另一个终止（‘finally’）模块。并且面向JDBC的异常与Spring通用数据访问对象（Data Access Object）异常等级一致。
5.以资源容器，DAO实现和事务策略等形式与Hibernate，JDO和iBATIS SQL Maps集成。利用众多的翻转控制方便特性来全面支持，解决了许多典型的Hibernate集成问题。所有这些全部遵从Spring通用事务处理和通用数据访问对象异常等级规范。
6.灵活的基于核心Spring功能的MVC网页应用程序框架。开发者通过策略接口将拥有对该框架的高度控制，因而该框架将适应于多种呈现（View）技术，例如JSP、FreeMarker、Velocity、Tiles、iText以及POI。值得注意的是，Spring中间层可以轻易地结合于任何基于MVC框架的网页层，例如Struts、WebWork或Tapestry。
7.提供诸如事务管理等服务的面向方面编程框架。

另外，Spring并没有提供日志系统，我们需要使用AOP（面向方面编程）的方式，借助Spring与日志系统log4j实现我们自己的日志系统。

总结：Spring是一个轻量级的控制反转（IOC）和面向切面（AOP）的容器框架和开源框架。
Spring框架的好处：
①从大小和开销两个方面而言Spring都是轻量级的
②通过控制反转的技术达到松耦合的目的
③提供了面向切面编程的丰富支持，允许通过分离应用的也无逻辑和系统级服务进行

内聚性的开发。

④包含并管理应用对象的配置和生命周期，从这个意义上来讲它是一个容器

⑤将简单的组件配置、组合成为复杂的应用，从这个意义上讲它是一个框架

# 关键字和保留字

2018年5月3日 20:03

**goto和const**是保留字也是关键字。

**关键字**列表 (依字母排序 共50组)：

abstract, assert, boolean, break, byte, case, catch, char, class, **const**（保留关键字）, continue, default, do, double, else, enum, extends, final, finally, float, for, **goto**（保留关键字）, if, implements, import, instanceof, int, interface, long, native, new, package, private, protected, public, return, short, static, strictfp, super, switch, synchronized, this, throw, throws, transient, try, void, volatile, while

**保留字**列表 (依字母排序 共14组)，Java保留字是指现有Java版本尚未使用，但以后版本可能会作为关键字使用：

byValue, cast, **false**, future, generic, inner, operator, outer, rest, **true**, var, **goto**（保留关键字）, **const**（保留关键字）, **null**

# Ajax

2018年5月3日 20:29

（1）Ajax（Asychronous JavaScript and XML）即异步JavaScript与XML,是一种结合了Java技术、XML以及JavaScript的编程技术，其主要目的是在不刷新页面的情况下通过与服务器的少量数据交互提高页面的交互性，减少响应时间，从而提高用户体验。使用Ajax技术后，页面不需要在每次用户提交修改时重新加载了。
（2）使用传统的软件架构开发的应用程序中，当用户需要与服务器频繁交互时，用户只有等整个页面重新加载后才能看到从服务器中获取到的资源信息，页面会被重新加载很多次。当前后两个页面的大部分HTML相同时，这种做法会非常浪费带宽，因为很多信息的获取都是无用的。
（3）使用Ajax带来的好处有：①Ajax技术可以只向服务器发送并取回必需的数据内容，使得数据交互量大幅度降低，从而降低了服务器的网络负载；②由于该技术可以通过SOAP(Simple Object Access Protocol，简单对象访问协议)或其他一些基于XML的WEB SERVICE接口，在客户端采用JavaScript处理来自服务器的响应，也降低了WEB服务器的处理时间③由于不需要加载整个页面，因此系统有更短的响应时间，这样有利于系统的稳定性和可用性，从而增强用户的满意度。
（4）Ajax技术是客户端技术，其核心是JAVASCRIPT对象XmlHttpRequest，该对象支持异步请求技术，使得开发人员可以使用JavaScript向服务器提出请求并处理响应，而不阻塞用户。

# J2EE

2018年5月3日 20:49

J2EE（Java2 Platform，Enterprise and Edition）是JAVA平台企业版的简称，用来开发和部署企业级应用的一个架构，他提供的一种统一的、开放标准的多平台，该平台有三部分组成。

（1）**构件**。包含**客户端构件**和**服务器端构件**两种类型。其中客户端构件主要包括Applets和Application Cilents；服务器端构件主要分为两类WEB构件（Servelet与JSP）和EJBs(Enterprise Java Beans)。

（2）**服务**。由J2EE平台提供商提供，分为Service API（开发时使用）和运行时服务。

（3）**通信**。由容器提供的支持协作构件之间的通信。

# IOC

2018年5月3日　　21:07

**1.IOC（Inversion of Control）控制反转，有时也被称为依赖式注入，是一种降低对象之间耦合关系的设计思想。**

①通过IOC方式，使得上层不依赖于下层的接口，即通过采用一定的机制来选择不同的下层实现，完成控制反转，使得由调用者来决定被调用者。IOC通过注入一个实例化的对象来达到解耦和的目的。使用这种方法，对象不会被显示的调用，而是根据需求通过IOC容器（例如Spring）来实现。

②采用IOC方式能够提高系统的可拓展性

③使用IOC，有以下两个方面的优点：

（1）通过IOC容器，开发人员不需要关注对象如何被创建的，同时增加新类也非常方便，只需要修改配置文件即可实现对象的“热插拔”。

（2）IOC容器可以通过配置文件来确定需要注入的实例化对象，因此非常便于单元测试。

④IOC的缺点

（1）对象是通过反射机制实例化出来的，因此对系统的性能有一定的影响

（2）创建对象的流程变得比较复杂

2.对象之间通过显示调用进行交互会导致调用者与被调用者存在非常紧密的联系，其中一方的改动将会导致程序出现很大的改动。而IOC控制翻转可以用过设计模式中的工厂模式来把创建对象的行为包装起来，进入工厂模式后，只需要在一个地方改动就可以满足要求，这样就增强了系统的可扩展性。

# AOP

2018年5月3日 22:31

面向切片的编程（Aspect-Oriented Programming，AOP）是面向对象开发的一种补充，它允许开发人员在不改变原来模型的基础上动态的修改模型以满足新的需求。例如，开发人员可以不改变原来业务逻辑模型的可以动态的增加日志、安全或异常处理的功能。

步骤：

（1）创建一个接口；

（2）创建实现这个接口的类

（3）配置SpringCongig.xml，使得这个类的实例化对象可以被注入到使用这个对象的测试类中

（4）完成配置文件后，编写测试代码

# Spring框架

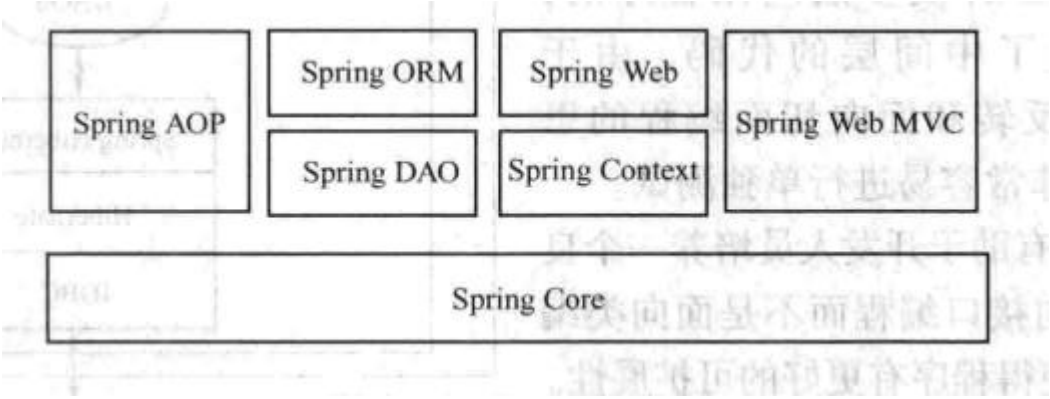
2018年5月3日 22:58

1.什么是Spring框架？

Spring框架是一个J2EE框架，这个框架提供了对轻量级IOC的良好支持，同时也提供了对AOP技术非常好的封装。

优点：相比于其他框架，Spring框架的设计更加模块化，框架内每个模块都能完成特定的工作，而且各个模块都可以独立的运行，不会相互牵制。因此，使用Spring框架时，开发人员可以使用整个框架，也可以只使用框架内的一部分模块。例如，只使用Spring 中的AOP模块来实现日志管理的功能，而不需要使用其他的模块。

2.Spring框架主要有7个模块组成。①Spring AOP ②Spring ORM ③Spring DAO ④Spring Web ⑤Spring Context ⑥Spring Web MVC ⑦Spring Core




3.使用Spring的好处

（1）使用J2EE开发多层应用程序时，Spring有效的管理了中间层的代码。由于Spring采用了控制反转和面向切面编程的思想，因此这些代码非常容易进行单独测试。

（2）使用Spring有助于开发人员培养一个良好的编程习惯：面向接口的编程不是面向对象的编程。面向接口的编程使得程序有良好的可扩展性。

（3）Spring对数据的存取提供了一个一致的框架（不论是JDBC还是O/R映射的框架，例如Hibernate或JDO）

（4）Spring通过支持不同事务的处理API（如JTA、JDBC、Hibernate等）的方法对事务的管理提供了一致的抽象方法。

（5）使用Spring框架编写的大部分业务对象不需要依赖Spring。

4.使用Spring的原理

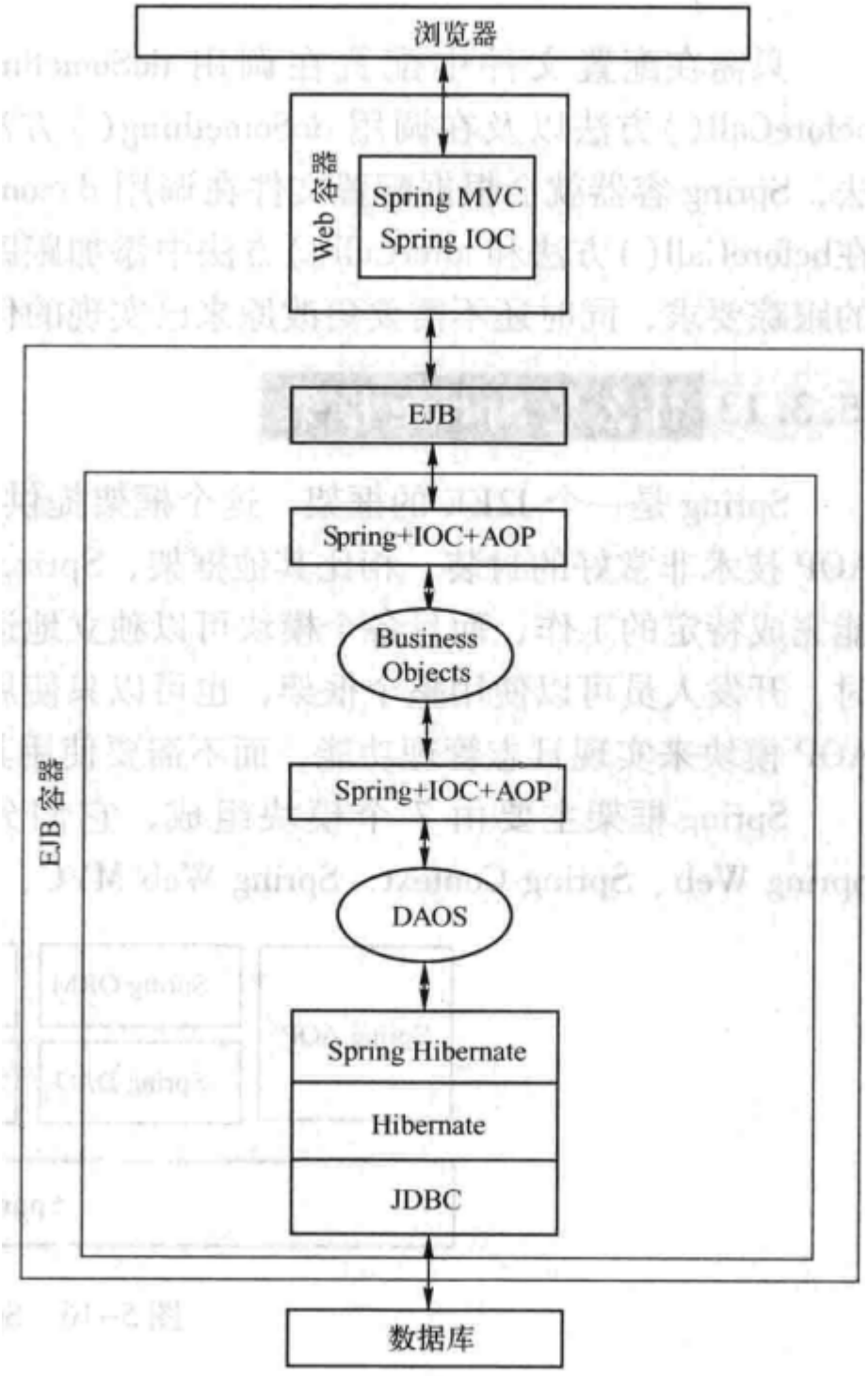



图 5-17　Spring 的工作原理

Spring有着广泛的用途：

（1）可以在Web容器中用来管理Web服务器端的模块，例如Servlet

（2）可以用来访问数据库的Hibernate

（3）Spring在管理Business Object（业务对象）和DAO时使用了IoC和AOP的思想，因此这些被Spring管理的对象都可以脱离EJB容器单独进行运行和测试。在需要被Spring容器管理时，只需要增加配置文件，Spring框架就会根据配置文件与相应的机制实现这些对象的管理。

# Hibernate

2018年5月3日 22:59

1.什么是Hibernate？

①Hibernate是一个开放源代码的对象关系映射框架（Object Relation Mapping，ORM，一种用来完成对象模型到关系模型的映射技术）。

②它不仅可以运行在J2EE容器中，也可以在J2EE容器外运行。

③Hibernate对JDBC进行了非常轻量级的对象封装，所以在任何使用JDBC的地方都可以使用Hibernate来替代。

④Hibernate实现了Java对象与关系型数据库记录的映射关系，极大的提高了软件的开发效率。

2.Hibernate的核心接口：

Hibernate的核心接口一共有5个：

①Session

②SessionFactory

③Transaction

④Query

⑤Configuration。

通过这些接口不仅可以完成对数据库的访问（增删改查）还可以实现对事务的控制。

**表 5-8 Hibernate 模块介绍**

| 接 口 名 | 描 述 |
| --- | --- |
| Session | 一个轻量级的非线程安全的对象，主要负责被持久化对象与数据库的操作。可以使用 SessionFactory 来创建一个 Session，当对数据库的所有操作都执行完成后，就可以关闭 Session。Session 在访问数据库时会建立与数据库的连接，这个连接只有在需要时才会被建立 |
| SessionFactory | 负责初始化 Hibernate。它可以被看作数据源的代理，可以用来创建 Session 对象。此外，SessionFactory 是线程安全的，因此可以被多个线程同时访问。一般而言，SessionFactory 会在 Hibernate 启动时创建一次，因此，为了便于使用，SessionFactory 应该用一个单例模式来实现 |
| Transaction | 负责事务相关的操作。它的主要方法有 commit( ) 和 rollback( )，其中 commit( ) 方法负责事务的提交，rollback( ) 方法负责事务的回滚，可以通过 Session 的 beginTrancation( ) 方法来创建 |
| Query | 负责执行各种数据库查询。可以使用 Hibernate 查询语言（Hibernate Query Language，HQL）或 SQL 语句两种方式进行查询（这两种查询方式非常类似，与 SQL 不同的是，HQL 语言使用类和属性而不是表与列名进行查询）。可以通过 Session 的 createQuery( ) 方法来创建 Query。此外，Hibernate 还提供了另外一种查询方式 QBC（Query By Criteria），其使用方法为：先使用 Session 实例的 createCriteria( ) 方法创建 Criteria 对象，接着使用工具类 Restrictions 的方法为 Criteria 对象设置查询条件，同时还可以用 Order 工具类的方法设置排序方式，最后用 Projections 工具类的方法进行统计和分组，使用 Criteria 对象的 list( ) 方法进行查询并返回结果。需要注意的是，QBC 是一种类型安全的面向对象的查询方式 |
| Configuration | 用于读取 Hibernate 配置文件，并生成 SessionFactory 对象。其中配置文件主要有两类：一类是 hibernate. cfg. xml 或 hibernate. properties；另一类是映射文件，例如 * * * . hbm. xml。其中 hibernate. cfg. xml 或 hibernate. properties 用来配置 Hibernate 服务的信息（例如连接数据库使用的驱动类、数据库连接的 URL、数据库的用户名和密码等信息）。如果同时提供了 hibernate. cfg. xml 和 hibernate. properties 文件，那么 hibernate. cfg. xml 会覆盖 hibernate. properties 中的配置信息。映射文件（ * . hbm. xml）用来配置 java 对象与关系数据库记录的映射关系。为了便于管理与维护，通常会给每个对象创建一个单独的映射文件 |

3.Hibernate的使用过程

（1）应用程序通过Configuration类读取配置文件并创建SessionFactory对象。

SessionFactory sessionfactory=new

Configuration（）.configure（）.buildSessionfactory（）；

（2）通过SessionFactory生成一个Session对象

Session session=SessionFactory.openSession（）；

（3）通过Session对象的beginTrancation（）方法创建一个事务；接着可以通过Session对象的get（）、load（）、save（）、update（）、delete（）和saveOrUpdate（）等方法实现加载、保存、更新和删除的操作。也可以通过session生成一个Query对象，然后利用Query对象执行查询操作；最后通过commit（）方法或rollback（）方法完成事务操作。

（4）在完成所有持久化操作和事务操作后需要关闭Session与SessionFactory

**4.Hibernate框架的好处**

①（1）提高开发效率

（2）使得开发完全采用面向对象的思想，不需要关心数据库的关系模型

（3）使用Hibernate开发的系统具有很好的可移植性，可以很容易地实现不同数据库之间的一直而不需要关心不同数据库SQL语句的差异

（4）支持透明持久化，Hibernate的API没有侵入性，当保存一个对象时，这个对象不需要继承Hibernate中的任何类和实现和实现任何接口。

②缺点：Hibernate只适用于针对单一对象简单的增、删、改、查，而对于批量的修改/删除的场合，则不适用，因此当要使用数据库的特定优化机制时，不适合使用Hibernate。

# Java中的值传递

2018年5月4日 10:13

对于Java的所有方法来说，它的参数都是以值传递的形式传递到函数中去的。对于参数是基本类型的数据传递的是基本类型的自变量值的拷贝，即在方法里对参数进行改变，只要这个参数是基本类型的就不能影响到参数实际的值；如果参数是引用类型的，那么将传递的是参数所引用的对象在堆中地址值的拷贝，即传递了一个同样的内存值进去，那既然内存值一样的话，那么对对象进行改变的话是能够影响到外面的

# ASCII码和ANSI码

2018年5月4日 20:40

1.标准的ASCII码只使用7bit，扩展的ASCII码使用8bit
2.ANSI码通常使用0x00-0x7f范围内的一个字节表示一个英文字符。超出这个范围用0x80-0xFFFF来编码，即扩展的ASCII编码。不同的ANSI编码之间互不相容

# 接口和抽象类

2018年5月4日 21:11

区别1：抽象类体现继承关系，一个类只能单继承。接口体现实现关系，一个类可以实现多个接口。
区别2：抽象类中可以定义抽象方法和非抽象方法，子类继承后可以直接使用非抽象方法。接口中的方法都是抽象的，必须由子类去实现，接口中只能声明方法，另外，接口中的成员有固定的修饰符。
区别3：抽象类有构造方法，用于子类对象初始化。而接口没有构造方法。
特点1：抽象类不可以实例化，即不能使用new创建对象。抽象类必须由其子类覆盖了所有的抽象方法后，该子类才能实例化。否则，这个子类也是抽象类。
特点2：抽象类abstract关键字不能和final、static、private关键字共存
特点3：接口中声明变量必须是final、public、static的。接口中定义的方法都是abstract、public的，并且接口里的数据成员必须初始化，且全都是常量，不是变量。
特点4：接口是抽象类的变体，接口可以使用extends关键字来继承其他接口。

```
Class 类名称 implements 接口A，接口B{
    //接口的实现
}
Interface 子接口名称 extends 父接口1，父接口2，...{}
```

# Http中Get方法和Post方法的区别

2018年5月7日 8:47

（1）Get方法的作用是从服务器端获取用户所需的资源，并将其作为相应返回客户端。Get获取服务器端信息时，就如同数据库中查询操作一样，不会影响到资源自身的状态。

（2）Post方法提供了比Get方法更强大的功能，它除了从服务器端获取资源外，同时还可以向服务器上传数据。

注意，Get方法也可以像服务器端上传数据，但是一般不推荐使用，原因有两个：①使用GET方法上传数据时，一般数据都添加在URL后面，并且用“？”连接，各个变量之间用“&”连接。由于URL长度存在限制，通常在1024Byte左右。而POST传递数据使用HTTP请求的附件进行，传送的数据量更大一些，一般默认不受限制。②由于GET方法上传的数据都是添加在URL中的，因此上传的数据存在安全隐患。而POST方法中向服务器提交的内容在URL中并没有明文显示。对用户都是不可见的，安全性更高。

# Servlet

2018年5月8日 8:50

（1）什么是Servlet？

Servlet是采用Java语言编写的服务器端程序，它运行于Web服务器的Servlet容器中，其主要功能是提供请求、响应的Web服务模式，可以生成动态的Web内容，这是HTML做不到的

（2）优点

①具有较好的移植性。无需代码修改就可以部署到多种不同的Web服务器上

②执行效率较高。Servlet针对每一个请求创建线程，由于创建线程的开销更小，所以Servlet交互过程中响应时间更小，响应效率更高

③功能强大

④使用方便：Servlet提供了多种不同的接口来读取和设置HTTP头信息，处理Cookies和会话跟踪。

⑤可扩展性强。由于Servlet是Java语言编写的，而Java语言是健壮的、面向对象的和可扩展的语言，因此Servlet也是可扩展的

（3）Servlet的处理流程

①用户通过单击一个连接向Servlet发送请求；

②Web服务器接收到这个请求后，会把该请求交给相应的容器来处理。当容器发现这是Servlet发送的请求后，容器会创建两个对象：HttpServletResquest和HttpServletReponse；

③容器可以根据请求消息中的URL消息找到相对应的用户请求响应，然后针对请求单独创建一个线程，同时将②创建的两个对象以参数的形式传递到新的线程中

④容器调用Servlet的service（）方法完成对用户请求的响应，service方法会调用doGet（）或doPost（）方法完成具体的任务，同时将生成的动态网页返回给容器；

⑤容器把响应消息组装成HTTP格式返回给客户端，此时，线程结束，并删除②中创建的两个对象。

# JSP&&&JSP AND Servlet

2018年5月8日 9:11

（1）什么是JSP？

JSP是由Sun公司倡导的、许多企业参与建立的一种动态技术标准，就是嵌入Java代码的HTML文件（但是JSP文件最好少写Java代码）

（2）JSP和Servlet的比较

①相同点：JSP可以被看做是一个特殊的Servlet，它只是Servlet的一个扩展，只要是JSP可以完成的工作，Servlet都可以完成。JSP页面最终都是要转化成Servlet来运行的，因此处理请求实际上是处理编译后的Servlet

②不同点：（A）Servlet的实现方式是Java中嵌入HTML代码，编写和修改HTML非常不方便，它比较适合做流程控制、业务处理；而JSP的实现方式是在HTML中嵌入Java代码，比较适合页面显示（B）Servlet没有内置对象，JSP的内置对象必须通过HttpServletRequest对象、HttpServletReponse对象和HttpServlet对象得到。

# 如何使用JSP和Servlet实现MVC模型

2018年5月9日 8:54

MVC是Model（模型）、View（视图）和Controller控制器3个单词首字母的组合。是目前广泛流行的应用模型，其目的是实现Web系统的职能分工。

（1）其中模型层实现系统中的业务逻辑，通常可以使用JavaBean或EJB来实现；视图层用于用户之间的交互，通常用JSP实现；控制层是视图和模型之间沟通的桥梁，他可以把用户的请求分派并选择恰当的视图来显示它们，同时它可以解释用户的输入并将其映射为模型层可以执行的操作。

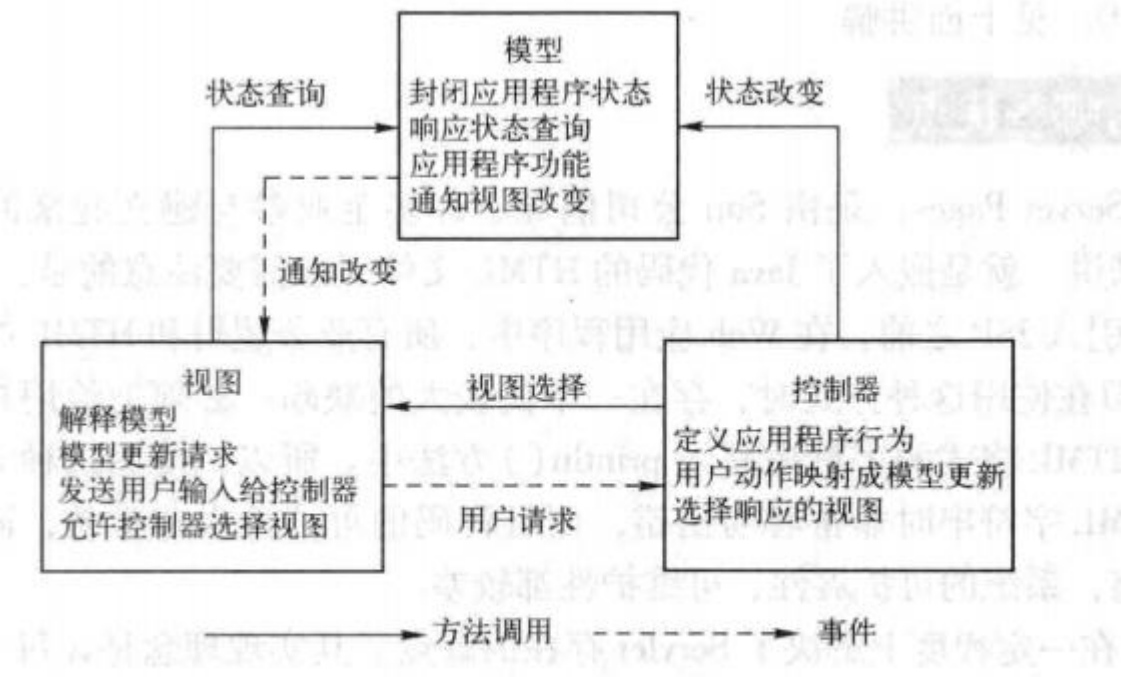



图 5-4 MVC 模型关系图

（2）MVC强制性地分离Web应用的输入、处理和输出，使得MVC应用程序被分成3个核心部件：模型、视图和控制器。

①模型（业务逻辑层）

模型表示企业的数据和业务逻辑，它是应用程序的主体部分，其处理过程对于其它层来说都是黑箱操作。模型接收视图请求数据，并返回最终的结果。（A）MVC把应用程序按照一定的规则抽象出来，抽象的层次很重要，这是判断设计人员是否优秀的主要依据。MVC并没有提供模型的设计方法，而只告诉设计人员应该如何组织管理这些模型，以便于模型的重构和提高重用性。（B）业务模型还有一个很重要的模型就是数据模型，数据模型指的是实体对象的数据持续化。比如将一张订单保存到数据库

②视图（表示层）

视图是用户看到并与之交互的界面。视图功能强大，主要表现在以下两个方面：（1）根据客户类型显示信息（2）显示商业逻辑（模型）结构，而不关心信息是如何获得何时获得的。

③控制器

控制器接收用户的输入并调用模型和视图去完成用户的需求。所以当用户单击Web页面中的超链接和发送HTML表单时，控制器（例如Servlet）本身不输出任何东西，也不执行任何处理，它只是接收请求并决定调用哪个模型构件去处理请求，然后确定用哪个视图来显示模型处理返回的数据。

MVC的处理过程是这样的：对于每一个用户输入的请求，先被控制器接收，并确定由哪个模型来进行处理，然后模型通过业务逻辑层处理用户请求并返回数据，最后控制器用相应的视图格式化返回的数据，并通过显示页面返回给用户。

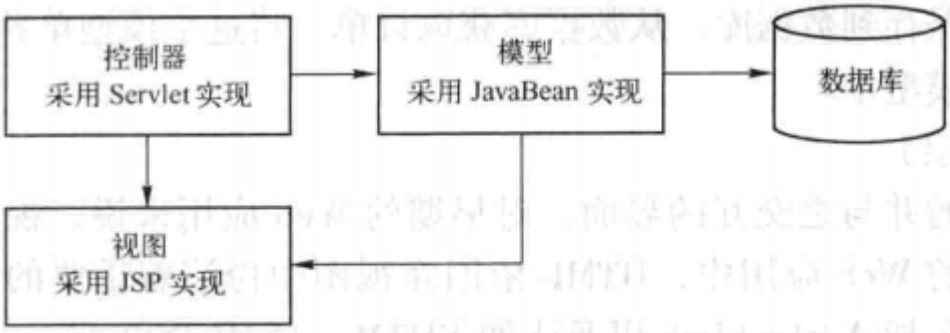



图 5-5　用 JSP 与 Servlet 实现的 MVC 模型