## .Net 面试题整理(一)

好久的资料了，拿出来分享给大家，主要是记录一些面试题整理。

Main函数是什么？在程序中使用Main函数有什么需要注意的地方？   
Q：程序的入口函数！函数名字不能改变；一个程序中有且只有一个Main函数。

CLR是什么？程序集是什么？当运行一个程序集的时候，CLR做了什么事情？   
Q:Common Language RunTime 公共语言运行时。   
编译后代码的集合。（包括exe和dll）。 加载项目里所有的类到内存，并找到其中的主函数，并作为默认的启动函数调用执行。

值类型的默认值是什么？（情况一：字段或全局静态变量；情况二：局部变量）   
Q：如果是全局变量，并且没赋值，就会有默认值，如：int 是0；bool 是false；枚举enum；结构体struct；   
如果是局部变量，就必须手动赋值。

声明一个变量时在内存中做了什么事情？初始化一个变量的时候又在内存中做了什么事情？   
Q:在栈中开辟空间，并将变量放入空间里，默认值都是null。   
初始化时：   
如果是值类型，则直接放入栈中；   
如果是引用类型，则在堆中开辟一块空间，将堆中对象的地址指针放入栈中

new关键字做的事情？   
Q：开辟空间，创建对象，调用构造函数，返回堆地址。   
还可以 显示的隐藏 父类的同名方法。

数组   
061.数组一旦创建后，能不能修改数组的长度？   
Q：不能。为什么呢？数组是引用类型。它在创建时，已经根据创建的长度在内存中开辟一块连续的空间。

多维数组 本质就是一个一位数组，只不过访问的 时候需要使用多个下标访问   
071.如何声明一个多维数组？   
Q：多维数组string[,] strArr=new string[3,3];

072.如何获取多维数组的总长度？   
Q：各个纬度元素的个数的乘积

交错数组   
081.如何声明一个交错数组？   
Q：交错数组：int[][] arr=new int[3][3];

082.交错数组的本质是什么？   
Q：本质是一个数组的数组。简单的说这个数组的元素是一个数组。

## 面向对象（一）

01.为什么要有方法？   
Q：为了复用。（封装）(重复调用，封装具体实现)

02.如何实现方法的重载？   
Q：方法名相同，但参数列表不同。（和返回值无关）

03.引用参数（ref）和输出参数（out）的相同点与不同点？   
Q：相同点：传递的都是（栈）引用。   
不同点：   
out侧重于输出参数；ref侧重于修改；   
out必须方法返回之前赋值；ref必须在传入之前赋值。

04.在什么情况下使用引用参数（ref）和输出参数（out）？   
Q：如果方法有多个返回值，则可以用out或者ref。只不过ref侧重于修改（即将值传进去修改后再拿出来）；而out是为了从方法中获得一个值后拿到方法外使用。   
05.可变参数与普通数组参数的区别？   
Q：params关键字。   
特点：   
一个方法中只能有一个可变参数；   
而且必须放在参数列表最后；   
如果没有给可变参数赋值，就初始化长度为0；

01.面向对象的三大特征是什么？   
Q：封装；继承；多态；

02.类和对象的关系？   
Q：抽象和具体的关系。类是抽象（模板，设计图纸），对象是具体（按照图纸设计的房子）。类是对对象的抽象描述，而对象是类的具体化。

03.创建某一个类的对象的时候，在内存中做了什么事情？例如 Person p = new Person();   
Q：开辟空间，创建对象，调用构造函数。（在内存的堆中开辟空间，创建Person对象，然后在内存的栈中开辟一个放一个p，然后将Person对象在堆中的引用地址赋值给对象p）   
04.属性存在的目的是什么？   
Q：封装字段。为了对字段的操作设置规则。（本质是两个方法，一个get方法和一个set方法）   
05.访问修饰符有哪些并分别每一个访问修饰符？   
Q：public（公有的），   
private（只有本类可以访问），   
internal（程序集内共享，如果不写修饰符，默认就是internal），   
protected（本类和子类可以访问），   
extern（供外部访问，eg：C#提供接口给C++访问）   
06.为什么要有构造函数呢？   
Q：主要方便程序猿在实例化对象的时候就为对象里的一些属性和字段初始化赋值。   
07.什么是封装？封装的目的是什么？   
Q：封装就是将对象的字段、行为和功能等“弄”到一起。   
封装的目的：隐蔽代码实现/复用/修改方便   
08.类的命名规则是什么？   
Q：类命名：首字母大写(帕斯卡命名)。变量命名:首字母小写(骆驼命名)

## 面向对象（二）

01.什么是类型？   
Q:用来定义某一种数据在内存里开辟空间的大小,还可以预置操作此种类型数据的相关方法

02.this关键字在方法中使用时，它代表什么？   
Q:this指当前类的对象，或者他的父类的类对象。   
base只能指向父类的对象。

03.值类型变量的初始化（内存分配）？（两种情况：一是类的成员变量，二是：方法的局部变量）   
Q：1、当变量是一个类的成员变量的时候，那么该变量是跟随累的对象存在于堆内存。当对象引用断开时，等着垃圾回收器进行清理时便被销毁   
2、当变量是一个方法的局域变量时，那么该变量是在方法被调用时，存在于栈内存，方法执行完毕后被销毁

04.继承   
041.继承的好处？   
Q:1.提高代码的复用；   
2.实现多态；

042.继承的单根性

Q：一个类只能继承一个父类。

043.继承的传递性

Q：指子类只能继承父类的非私有的成员。

044.当存在继承关系的时候，在子类中如何访问父类的非私有成员？

Q：通过base关键字--（只能获取父类的）（base就是存在于子类对象里，用来指向父类对象的指针）

或者通过this关键字--（可以获取父类的非私有的成员和本类的成员）

05.什么是里氏替换原则？   
Q：子类替换父类所在的位置

06.子类与父类对象之间的转换？   
Q：子类可以强制转换为父类，但是父类不能转换为子类。

07.is 和 as 操作符的用处和区别？   
Q：is是对类型的判断，返回bool。（判断A是否是B类，或者A是不是B的子类）。(如果一个对象是某个类型或是其父类型的话就返回true，否则就会返回为false。另外is操作符永远不会抛出异常。   
as是用来做类型转换的。as只能针对引用类型的转换。as 是先判断再转换（as 操作符首先测试转换是否合法，如果合法就转换，否则返回NULL。不会报错）   
这两者都不会抛出异常。

08.override可以重写哪些 “类型” 的方法？   
Q：可以重写虚方法（virtual）–必须有方法体   
还可以重写抽象方法 （abstract）–没有方法体

09.什么是多态？   
Q：就是用某个子类来实例化父类，调用的是父类的抽象方法（虚方法），但实际执行的是子类实例重写的方法

10.抽象方法只能定义在抽象类中吗？   
Q：是的。   
（virtual or abstract members cannot be private）

## 面向对象（三）

01.CLR什么时候加载类？   
Q：是在第一次在代码中遇到这个类的时候才加载类的静态成员和信息。（为了节约资源，只在用的时候才开始加载进去）

02.当存在继承关系的时候，创建子类对象的时候会不会创建父类对象呢？   
Q：这个很易错！！！！当我们创建一个子类时不会创建父类对象，只会有一个对象。Person c=new Student()，（当我们new一个子类的时候，在内存中，只有一个子类对象）   
！！–！！—-那下面括号里面的就是错误的！！！   
（new子类时，先创建子类对象，调用子类的构造函数时，创建父类对象，调用父类构造函数最后返回来执行子类构造函数方法体代码

子类默认调用父类的无参构造函数

如果父类里没有无参构造函数，则必须显示的调用）

其实在内存中是没有对象概念的！new的时候是根据子类和父类所有静态成员的大小以及类型指针开辟一个总的空间（此空间不包含方法），子类和父类的类型指针指向两个类的方法表，所以，子类对象可以访问到父类和子类的方法。

03.如果子类和父类存在同名但没有重写关系的方法的时候，那么会调用哪个方法？（两种情况：一是使用子类对象，而是使用父类对象）   
Q：如果子类和父类存在同名但没有重写关系的方法，那么调用时，如果是父类变量调用，则调用父类方法，如果是子类变量调用，则调用子类方法

04.如果子类和父类存在同名且有重写关系的方法，那么会调用那个方法？（两种情况：一是使用子类对象，而是使用父类对象）   
Q：如果子类和父类存在同名且有重写关系的方法，那么不管变量是什么类型，调用的都是子类的方法。

05.虚方法和抽象方法的相同点与不同点？   
Q :相同点：都能被子类重写；都不能为私有的   
不同点：   
1.abstract 和virtual的关键字不同   
2.抽象方法一定没有方法体，而虚方法必须有方法体。   
3.virtual方法的子类可以重写父类方法也可不重写，而抽象方法的所在的类的子类必须重写该父类方法   
4.抽象方法必须在抽象类中声明，而虚方法可以存在于任何一个类里面   
5.virtual方法必须有现实（哪怕是空实现），而abstract方法必须没有实现

06.子类中的base关键字指向的是子类对象还是父类对象？它本身的类型是子类类型还是父类类型呢？   
Q：当然是指向的父类对象。 但是本身是子类的类型。   
（base就是子类对象本身,只不过类型是父类）

07.为什么要有抽象类？   
Q：抽象类存在的目的就是指定规则，而不是具体实现，是为了约束子类的行为，具体实现交给子类完成。

08.使用多态的好处是什么？   
Q：1.低耦合，高内聚。（低耦合—代码的内部各个层之间的关联小，相互之间约束少。 高内聚—方法的功能和职责单一，即一个方法只做一件事情。）   
2.实现代码的可维护性 和 可扩展性。   
3.把不同的子类对象都当作父类来看，可以屏蔽不同子类对象之间的差异，写出通用的代码，做出通用的编程，以适应需求的不断变化。

09.什么情况下的类不能被实例化？   
Q：1.抽象类   
2.静态类   
3.类的构造函数被私有化

10.什么情况下抽象类的子类不需要实现父类的抽象成员？   
Q：子类也是抽象类的时候。

11.虚方法（虚拟成员）和抽象方法（抽象成员）能不能被私有化？   
Q：当然不能。（首先编译就会报错）。—深层原因：他们两者存在的目的就是为了让子类去重写，如果私有化了，就没意义了。

12.静态成员能不能被标记为virtual、override 或 abstract？   
Q：不能：

13.接口的命名规则？   
Q：以大写I为开头，以…able结尾。表示有某种能力。

14.什么是接口？   
Q：特殊的抽象类。他的存在完全是为了约束和统一类的行为。

15.接口能包含的成员有哪些？   
Q：接口只有方法、属性、索引器和事件的声明

16.接口的访问权限修饰符只能是哪两种？   
Q：public和internal

17.接口能不能继承接口？   
Q：能。而且可以实现多个接口。

18.如果一个抽象类继承了一个接口，那么将继承的方法设置为抽象方法，还是去实现该方法？   
Q：都可以。 既可以在抽象类的子类中去实现，也可在抽象类中直接实现。

付炯 2月19日   
1、使用接口的注意事项   
Q:1，接口中的成员不能加访问修饰符；   
2，接口中的成员不能有任何实现；   
3，实现接口的子类的必须实现接口的全部 成员；   
4，接口中只能有方法，属性，索引器，事件，不能有字段。   
5，一个类可以同时继承一个类并实现多个接口，如果一个子类同时继承了父类A，并实现了接口IA，那么在语法上A必须写在IA的前面，因为类是单继承的，而接口可以实现多个。   
6，显示实现接口和隐式实现接口时需要注意的问题，（通常我们是隐式实现的），   
类实现接口，可以显式的实现接口里的方法，但是注意：一旦使用显式实现接口里的方法后，那么该方法只能被接口变量调用   
IFlyable fly = new Bird();   
fly.Fly(); //正确   
Bird fly = new Bird();   
fly.Fly(); //错误

7，向上转型

8，单一职责原则 （避免定义体积庞大的接口，因为这样会造成“接口污染”，只把相关联的一组成员定义到一个接口中）

2、接口与抽象类的区别   
Q：相同点：   
接口和抽象类都不能被实例化；   
不同点：   
1，接口中不能存在有具体实现的方法；而抽象类中可以有具体的方法实现。（接口只能定义行为，而抽象类既可以定义行为也可以提供实现）   
2，接口支持多继承，而抽象类不能实现多继承；   
3，接口可以定义方法Method，属性property，索引器Index，事件Event的签名，但不能定义字段和这些东西的具体实现，而抽象类什么都能定义和实现，除了 不能被实例化

老邹的2011年答案：

1) 抽象基类可以定义字段、属性、方法实现。接口只能定义属性、索引器、事件、和方法

声明，不能包含字段。

2) 抽象类是一个不完整的类，需要进一步细化，而接口是一个行为规范。微软的自定义接

口总是后带able字段，证明其是表述一类“我能做。。。”。

3) 接口可以被多重实现，抽象类只能被单一继承。

4) 抽象类更多的是定义在一系列紧密相关的类间， 而接口大多数是关系疏松但都实现某一

功能的类中。

5) 抽象类是从一系列相关对象中抽象出来的概念， 因此反映的是事物的内部共性；接口

是为了满足外部调用而定义的一个功能约定， 因此反映的是事物的外部特性。

6) 接口基本上不具备继承的任何具体特点,它仅仅承诺了能够调用的方法。

7) 接口可以用于支持回调,而继承并不具备这个特点。

8) 抽象类实现的具体方法默认为虚的， 但实现接口的类中的接口方法却默认为非虚的， 当

然您也可以声明为虚的。

9) 如果抽象类实现接口，则可以把接口中方法映射到抽象类中作为抽象方法而不必实现，

而在抽象类的子类中实现接口中方法

3、类型转换   
转换分类   
Q：隐式转换 int num = 1; string str= num.tostring();   
强制转换 int a = 5; (double ) b= a;

转换条件

ToString()方法

Q：它是object对象的一个虚方法，可以自己去重写该方法。如果不重写就返回类型名。

Parse()和TryParse()方法

Q：相同点：都是针对于字符串进行的值类型转换。

不通点：1，返回值不一样。Parse()是返回要转换的类型，而TryParse()返回一个转换是否成功的bool值。

2，Parse()转换失败会拋异常，而TryParse()转换失败返回false，不会拋异常。

3，Parse()直接传入要转换的类型的值，而TryParse()需要一个out输出参数输出转换的结果。

4、静态成员   
加载时机 (什么时候加载静态成员？)   
　Q：在该静态成员第一次被调用的时候加载。

适用情形（什么时候用静态成员？）

Q：在整个程序内部共享的数据才定义为静态的。通常被作为工具类使用。比如SQLHelper

在普通类和静态类中的区别

Q：1，静态类需要用static。静态类不能被实例化。

2，静态类中只能包含静态成员；

3，静态成员属于类所有，非静态成员属于类的实例所有。

4，在实例方法中可以直接调用静态成员，但在静态方法中不能直接调用实例方法。

5，静态类和静态变量创建后始终使用同一块内存，而使用实例的方法会创建多个内存。

6，静态构造函数不能有参数，也不能有访问修饰符（默认是private）

5、静态类的继承   
Q:1,(从子类的角度)静态类不能被任何类继承   
2,(从父类的角度)静态类只能继承Object类，不能继承其他的类。

6、类和成员的访问修饰符   
Q:1，类的默认访问修饰符是internal，   
2，类的成员的默认访问修饰符是private

a.类的访问修饰符只有两种:public，internal（默认）

b.成员的访问修饰符有:public，protected，private（默认）

7、结构   
本质是值类型   
值类型和引用类型的选择   
Q：1.值类型：主要是用来封装一组数据，并为数据提供一种简单的处理方式   
2引用类型：   
1）.主要用来封装数据和行为   
2）使用面向对象的特征；   
3）当类型中的成员比较多的时候用结构（存在堆里）

new关键字的作用

Q：结构

在使用new关键字创建对象后，所有的成员变量都已经存在，并有默认值（值类型）

如果没有用new关键字，则需要程序员手动为所有的 用到了 的成员变量赋值，之后才能调用结构对象里的方法属性

结构不new也可以使用，但是必须给使用到的结构成员赋值才能使用

8、类和结构的区别   
Q：1，结构是值类型，是分配在内存的栈上的。而类是引用类型,是分配在内存的堆上的；   
2，结构不能被继承，因为结构是值类型，隐式继承自System.ValueType   
3，结构是值传递的（复制传递），而类是引用传递的。

9、值类型和引用类型作为参数传递的区别   
Q：值类型是值传递的（复制传递），而类是引用传递的（传的是地址的引用）。

10、访问级别约束   
Q：1，子类的访问级别不能比父类高   
2，方法参数的访问级别 >= 方法的访问级别 （比如当方法的参数传递的是一个类对象时，那么此时这个类对象的访问级别要高于当前方法的访问级别）

11、析构函数   
Q：1，一个类只能有一个析构函数，   
2，无法继承或重载析构函数   
3，我们无法手动去调用析构函数，因为它是被GC（垃圾回收器）自动调用的   
4，析构函数不能有访问修饰符，也不能有参数   
5，不能在结构体中定义析构函数（为什么呢？）–因为结构是值类型，而值类型是存储在栈中的，栈中的数据在用完之后就立即销毁了，而析构函数的目的就是用来释放资源的，一般存储在堆中的引用类型才需要GC去释放，因而结构体中是不能定义析构函数的，只能对类使用析构函数。

析构函数语法：

class MyDispose

{

~MyDispose()

{

......//在这里写释放资源的代码

}

}

12、字符串   
属性   
length   
静态方法   
常用：   
1、（Last）IndexOf：用来查找某个字符或字符串，在一个特定字符串对象里的下标   
2、SubString 截取   
3、Split() 根据特定字符来分割字符串，并返回分割后的字符串的数组，可以用foreach读取   
4、Join静态方法   
5、Format() 静态方法   
6、Replace()，替换完要接收，产生一个新的字符串   
7，Replace().Replace()链式编程   
8、Trim()去首尾空格

实例方法

13、==运算符和Equals()方法的区别   
Q：”==”比较时：   
如果比较的是值类型，则比较两个对象的值   
如果比较的是引用类型，则比较两个对象的引用地址是否相同（比较堆地址）   
“Equals”比较时：   
此方法是Object类里的一个虚方法，默认就是用的“==”进行比较。（它是对“==”进行的一个封装）   
但是，大部分微软的类，及用户自定义的类，都重写了该虚方法，也就是微软和用户各自为自己编写的Object的子类 定义了相等比较规则。   
注意：这里有一个特例，因为string是一个引用类型，所以按理说string.Equals(…)方法比较的是地址，而这里比较的字符串的值。

14、字符串的恒定性   
Q：当字符串在内存中已经被创建后，程序员在次创建相同值的字符串对象时，CLR做了优化，直接把第一个字符串的引用赋给了第二个变量，也就是说，前后两个字符串变量保存了相同的字符串对象应用

15、StringBuilder对象   
Q：高效的的字符串操作。   
String 在进行运算时（如赋值、拼接等）会产生一个新的实例，而 StringBuilder 则不   
会 。 所以在大量字符串拼接或频繁对某一字符串进行操作时最好使用 StringBuilder ， 不要使   
用 String   
如果要操作一个不断增长的字符串，尽量不用 String 类 , 改用 StringBuilder 类。两个类的工   
作原理不同 :String 类是一种传统的修改字符串的方式 ， 它确实可以完成把一个字符串添加到   
另一个字符串上的工作没错 , 但是在 .NET 框架下 ， 这个操作实在是划不来 。 因为系统先是把   
两个字符串写入内存 ， 接着删除原来的 String 对象 ， 然后创建一个 String 对象 ， 并读取内存   
中的数据赋给该对象。这一来二去的，耗了不少时间。而使用 System.Text 命名空间下面 的   
StringBuilder 类就不是这样了，它提供的 Append 方法，能够在已有对象的原地进行字符串   
的修改 ， 简单而且直接 。 当然 ， 一般情况下觉察不到这二者效率的差异 ， 但如果你要对某个   
字符串进行大量的添加操作 ， 那么 StringBuilder 类所耗费的时间和 String 类简直不是一个数   
量级的。

16、枚举   
本质是类   
枚举项的相关问题   
Q：1、如果为第一个枚举项赋了一个int值，那么后面的枚举项依次递增。   
2、可以将枚举强转成他所代表的int值   
3、因为枚举项都有对应的int值，所以Switch把他当成int看   
4、C#的枚举项都是常量（可以去看IL 代码 literal）   
5、不能定义方法，属性，事件   
6、多个枚举有相同数值时。数值强转时，会返回其中最后一个枚举项   
7、枚举项的数值类型int ,long….

17、IEnumerable接口   
Q：只要实现了该接口，就可以使用foreach进行遍历。 foreach循环的本质就是调用这个接口返回一个迭代器，调用迭代器的MoveNext()方法就可以实现循环。   
如下反编译的源码：   
public interface IEnumerable   
{   
IEnumerator GetEnumerator(); //返回一个迭代器   
}

public interface IEnumerator

{

bool MoveNext();

object Current { get; }

void Reset();

}

从以上反编译源码可以看出：

IEnumerable接口中主要包含GetEnumerable方法（获取迭代器对象），MoveNext方法（检查是否存在循环的下一个元素），GetCurrent方法（获得当前循环到的元素）

2月20日   
1、集合   
概念理解   
Q: 集合就是能装一堆东西的容器。主要分为非泛型集合和泛型集合。   
Arraylist— 里面真正存储数据的是一个Object[]数组，它对应的泛型是List,   
HashTable—非泛型的键值对集合，它对应的泛型是Dictionary

## 11.4 表单JS提交方式

王伟 3.18-3.23 第一部分：**[jQuery](http://lib.csdn.net/base/22" \o "jQuery知识库" \t "http://blog.csdn.net/vfush/article/details/_blank)**

一、 查：当你想在页面中找到某个元素的时候，就要想到他们   
↖(^ω^)↗   
1.1基本选择器：   
Id选择器 ，Calss选择器，tag选择器，\* ，组合选   
择器   
1.2层次选择器：   
后代选择器，子代选择器，匹配选择器，~选择器   
1.3节点遍历：   
next(),nextAll(),prev(),prevAll(),siblings()   
1.4过滤器：   
：first,:last,:not,:even,:odd,:eq,:gt,:lt,   
1.5属性过滤器：   
IMG_256IMG_257IMG_258IMG_259IMG_260IMG_261IMG_262IMG_263IMG_264IMG_265IMG_266IMG_267(“div[title=text]”),   
1.6表单对象过滤器：   
#form1:enabled,#form1:disabled,input:checked,select   
option:select,   
1.7表单选择器   
：input，：text，：password……   
二、修改：修改页面元素就用我吧 (^\_\_^) 嘻嘻……   
2.1jqury对象的方法   
html(),val(),text(),css(),attr(),removeAttr()   
2.2样式操作：   
attr(),addClass(),removeClass(),toggleClass(),hasClass()   
2.3复制节点：   
clone(),clone(true)   
2.4替换   
replaceWith(),replaceAll(),wrap(),   
三、新增：当你要新增元素时就用它们吧！O(∩\_∩)O哈哈~   
3.1创建Dom节点 例：$(“[传智播客官网](http://blog.csdn.net/vfush/article/details/%E2%80%99" \l "’)”)   
3.2 append()，appendTo()   
3.3外部插入节点   
after,before,insertAfter,insertBefore   
四、删除：你不要我们了吗？ ( ^\_^ )/~~ 拜拜   
remove()   
empty()   
五、批量处理   
5.1隐式迭代   
5.2IMG_268IMG_269IMG_270IMG_271IMG_272.each   
5.3元素的map和each   
六、其他内容：   
jquery动画   
jquery cookie

第二部分：Dom补充   
放在这里是不是有点莫名奇妙 (⊙\_⊙?)   
ChildNodes(NodeList类型-有序类数组对象)，   
length属性是实时的计算元素个数   
insertBefore,appendChild,removeChild,replaceChild   
parentNode(父元素),firstChild(第一个子元素),lastChild(最后一个子元   
素),ownerDocument   
nextSibling(下一个兄弟节点) , previousSibling(上一个兄弟节点)   
cloneNode(bool)-复制节点(true-深拷贝,false-浅拷贝)   
style.cssText设置元素的style属性值

第三部分：js高级   
1. Js面向对象编程(继承)   
2. 闭包   
3. apply和call   
4. 函数的属性（arguments）   
5. Eval()方法   
6. 数据类型（array，object，function（不是）……）   
7. Instanceof,typeof

## 8. 声明式和表达式执行顺序

张连印 3.25-3.30   
一：反射   
1.同过反射动态调用方法 Invoke   
2.Type.IsAssignableFrom(Type t);判断type是否是T的父类，可判断接口   
3.Type.IsSubclassOf(Type T)//与上面方法的区别   
4.通过反射获得Type的方法，如私有方法、共有方法、静态方法????   
5.开发一个基于插件的记事本   
6.特性，通过反射获得特性信息，如何定义特性。

二．多线程：   
1.多线程的概念   
2.多线程的实现   
3.线程重入   
4.前台线程和后台线程   
5.线程的调度方式

三.   
1.什么是socket？   
2.Socket的常用两种通信协议：Tcp/Udp   
3.Tcp、Udp的区别？   
4.编写基于Socket Tcp协议的一个简单即使通信软件   
5.Http服务器的运行方式。   
6.长连接和短连接的区别？   
7.Web服务器的工作过程   
7.1.监听请求   
7.2.处理请求   
7.3.生成响应报文   
7.4.关闭通信套接字Socket

四．一般处理程序   
1.IIS Web服务器的处理过程：映射表、外部扩展程序、静态文件、动态文件、能处理、不能处理   
2.HttpContext上下文对象的作用，其中包含哪些内容？   
3.Asp.net系统对象：   
Page、response、request、application、session、server、cookies   
4.浏览器提交表单的两种方式   
5.Get/Post的区别是什么？   
6.使用一般处理程序制作用户登录程序。   
7.Request获取Get、Post请求参数的方式分别是什么？Params   
8.Response.Redirect() 方法作用。   
9.Response中的写出器   
10.浏览器能够提交表单需满足什么条件？   
11.哪些Html标签的值能够被表单提交？   
12.若Html标签的disabled属性有设置   
13.读取Html模板文件，处理Html字符串   
14.页面的跳转：window.location,window.parent.location,window.top.location   
15.一般处理程序进行增删查改

李荣壮 4.1-4.6   
———-4.1 一般处理程序   
1. 上传图片   
2. 添加水印   
3. 生成缩略图   
4. 生成验证码   
5. 设置浏览器打开文件打式为下载   
———-4.2 - 4.5 WebForm和状态保持   
1.Ashx与Aspx的关系   
2.Aspx前台页与后台页的关系   
CodeBehind（代码后置）   
3.前台页与后台页之间传递数据   
4.Aspx、cs、dll文件之间的关系   
5.WebForm的运行流程   
ASP.NET 运行图   
控件树   
6.Request   
6.1 常用成员   
UrlReferrer   
UserHostAddress   
MapPath(Server.MapPath调用的是Request.MapPath)   
7.Response   
7.1 常用成员   
End()   
ContentType   
8.Servers   
8.1 常用成员   
8.2 Transfer与Redirect区别   
Transfer不能内部重定向到ashx，否则会报错“执行子请求出错”   
9.无状态Http   
9.1 其根本原因 :Http协议是无状态的   
9.2 对网站的影响   
10.ASP.NET中的状态保持方案（有哪些，各自的特点，应用场景）   
10.1 客户端：   
10.1.1 ViewState   
10.1.1.1 特点：WebForm特有，页面级的   
10.1.1.2 两种使用方式   
10.1.1.2.1 用户数据保存方式   
10.1.1.2.2 非单值服务器控件的状态自动保存于ViewState   
10.1.1.3 使用ViewState的前提   
10.1.1.4 ViewState的禁用（WebForm的IsPostBack依赖于\_\_ViewState）   
10.1.2 HiddenField   
10.1.3 Cokies   
10.1.3.1 保存Cookie的两种方式   
1. 保存于内在中   
2. 保存于浏览器所在的电脑的硬盘中。   
10.1.3.2 使用场景   
10.1.3.3 原理   
10.1.4 ControlState   
10.1.5 QueryString   
10.2 服务器：   
10.2.1 Session   
每个客户端的Session是独立的   
开发场景   
如何使用   
一般处理程序如果使用Session需要 实现RequiresSessionState接口   
与Cookies的区别   
10.2.2 Application 服务器端保存共享数据的一种方式   
10.2.3 Caching   
10.2.4 Database   
11. Web应用程序与网站的区别   
12. Iframe   
13. 控制父页跳转   
14. 反射方式生成sql查询条件   
15. MD5加密   
——–4.5 AJAX   
1. 优点   
2. 使用步骤   
3. 浏览器兼容方式创建异步对象   
4. post 设置 ContentType   
5. get 设置 不读取浏览器缓存   
6. 地区级联选择   
7. 服务器端与客户端通过json交换数据

## 8. 将对象序列化为json字符串。

姚羽 4.8-4.13   
1.Ajax的使用   
1.1四个步骤（核心！重要！）   
1.1.1 创建异步对象：Create XMLHttpRequest   
1.1.2 设置访问方式：open (…)—get 和 post （各自的请求头设置语法）   
1.1.3 设置回调函数：onreadystatechange = function(){…}   
1.1.4 发送请求：send (传参)   
1.2处理ajax请求响应的结果responseText。 处理json格式的字符串，   
注意引号问题

2.Ajax完成 增删改查 一套功能，注意一些细节   
2.1 细节一：查询时Js创建表格   
2.2 细节二：新增时Js创建行和删除行的操作

3.Ajax简单分页   
3.1 自己利用row\_number写一个简单的分页存储过程（带参数）   
3.2 C#内部的事务操作（自己做成规范的方法自己将来在需要的时候可   
以直接拿来用）   
3.3 照片上传 和 异步上传（原理：利用Iframe）   
3.4 JS自执行函数的使用和JS匿名函数的使用

4.服务器控件   
4.1 三种控件的区别（html控件，有runat=server的html控件，服务器   
控件）   
4.2 Jsonp 的作用以及怎么实现跨域   
4.3 主要掌握服务器控件的 Repeater （Eval和Bind的区别）   
4.4 掌握服务器控件的ListView（各个模板的使用，以及内置分页功能   
，高效分页做法）   
4.5 页面生命周期 （表述这个过程和原理）

1. 缓存 （有哪几种缓存（页面级别缓存，数据源缓存，自定义缓存）—各自   
   用法，回忆）   
   5.1 缓存的条件 （为什么要有缓存？）   
   5.2 缓存的缺点（脏数据）   
   5.3 如何解决这个缺点 （缓存依赖） （原理）   
   5.4 Session和Cache的相同点 和 不同点   
   5.5 自定义缓存的 绝对失效时间 和 相对失效时间 （那两句代码，不   
   同的写法）

6.母版页   
6.1 母版页的使用（占位符PlaceHolder），以及母版页和子页面的关系，   
先执行母版页还是先执行子页面   
6.2 关于这阶段的最重要的那张图（请求流行执行图），只要在面试过   
程中设计到这阶段的问题，就可以想办法口述出那张图的流程，需要流利表达和自信

个人补充（核心的加分回答）：   
1，页面静态化（伪静态 和 真静态）（可以提到在mvc中静态化的方便之处）：   
1.1，目的：   
Q: 1，最大限度的方便搜索引擎的抓取页面（百度谷歌等网络蜘蛛的爬取），利于SEO   
2，直接生成静态页面，，服务器端不用每次访问都去运算，这样就减轻了服务器压力   
3，给用户更加友好的展示，地址栏显示的直接是一个纯的html，而不是像….aspx?id=1… 这样一长串的参数，干净，好记忆   
1.2，如何实现页面静态化：

2，页面几大对象的深层次原理（Session，Viewstate，Application，cookie等）

2.1，关于Session：

2.1.1，Session出现的背景:

Q： 由于Http协议的无状态性，浏览器和服务器通信完毕之后，连接就被断开。这样就造成服务器就无法保存浏览器端的状态.

而为了解决这个问题有些聪明的人就发明了状态保持对象，而Session就是其中的一种存放在服务器端的状态保持对象。

2.1.2，那么Session是如何进行状态保持的呢？:

Q： 比如在登录的时候，浏览器端发送账号密码到服务器端，通过验证之后，服务器端就将这些数据以键值对的方式保持到Session池中，

并且服务器端生成响应报文，并以内存型的Cookie的形式发送SessionId.

在浏览器未关闭的条件下，当浏览器再次访问这个站点下的页面时，就会将这个站点接收到的SessionId包含在请求报文中一并发送到服务器端。

服务器端一旦拿到这个SessionId，就会去Session中根据SessionId查找对应的值，如果找到，就不再去进行验证。因而就实现了这种状态保存机制。

就是这样，服务器端就能够保持客户端的状态。

2.1.3，用Session会出现那些问题呢？那有如何解决呢？

Q：会出现的问题 ： Session的性能不是很好；当网站的访问量大的时候容易造成内存泄漏。

如何解决 ：

2.1.3.1，采用进程外Session

2.1.3.2，我们可以模拟一个Session，比如把一个用户的登录信息和一个Guid放在一起，这样就可以作为一个键值对来模拟Session，然后将其存入到数据库中，这样就可以避免Session的丢失。我们甚至可以将其存入缓存（因为缓存的性能高），这样就可以中和数据库的硬盘慢速访问和内存的高速访问。

具体方案：

对于进程外Session，主要有这几种。。。。。。。， 通过配置文件，这是微软帮我们提供的方式，我们只需要配置一下就OK。<sessionState mode="StateServer" cookieless="UseCookies" stateConnectionString="tcpip=127.0.0.1:42424"/>

3，asp.net的运行机制，页面生命周期相关

Q：当客户端浏览器发送一个请求的时候，比如用户在浏览器输入一个域名地址，那么此时先进行DNS寻址，就是到DNS服务器去找该域名对象的IP，找到之后就会跟这个对应的IP建立连接，这个连接是通过Socket套接字建立起来的，然后通过套接字连接发送连接请求，此时就跟我们的IIS服务器建立了连接了，IIS拿到请求之后会将请求交给扩展程序，扩展程序就将请求交给Framework来处理，aspnet\_isapi.dll ，这时会创建一个HttpRuntime对象，调用其PR方法创建了HttpWorkerRequest对象，并将请求报文信息封装在里面（当然，这个HttpWorkerRequest我们程序员无法直接调用，它是供.Net运行时来用的），然后创建HttpContext对象，它的里面就包含了请求报文对象HttpRequest和响应报文对象HttpResponse，此时通过HttpApplicationFactory创建了一个HttpApplication对象，然后调用该对象的PR方法，并传入我们上面已经创建好的上下文对象HttpContext，然后会执行19个委托对象，这也就是我们常说的请求管道，在执行到管道事件的第8个事件时会创建被请求的页面类对象，并将页面类对象转成IHttpHandler接口。然后会在执行到第11到12个事件之间的时候调用页面类对象的PR方法，执行完毕后，这时就要分两步走，如果当前请求的是一般处理程序ashx，那么就会直接调用前面创建的HttpRuntime的一个FinisheRequest()方法，这个FinishRequest就会生成响应报文，然后将响应报文发回给IIS，再由IIS通过套接字发回给浏览器，那么浏览器就砍刀了我们请求的响应报文。

4，mvc的运行原理，路由的相关的操作（可以谈到领域先行的开发模式，先关的，比如如何管理EF上下文，线程内实例唯一，Callcontext）

5，JS高级中的几个概念，比如js闭包，apply和call，原型模式prototype，js继承等等

6，缓存，哪几种？cache，有什么好处，会有什么问题？如何解决？ 什么是缓存的滑动机制？谈到缓存依赖，几大依赖方法

6.1，概念：

缓存是指系统或应用程序将频繁使用的数据保存到内存中，当系统或应用程序再次使用时，能构快速的获取数据。

6.2，有哪几种：

主要有三种： 页面级缓存（网页输出缓存）； 数据源缓存； 自定义缓存

6.3，各是怎么实现的？

6.4，利弊是什么？

利：

弊：它的弊端在于显示的内容可能不是最新，最精确的（即通常我们说的脏数据）

ASP.Net 缓存主要分为两大类： 网页输出缓存和应用程序缓存

网页输出缓存针对ASP.NET Pages 页面中的HTML进行缓存，是可视化内容对象，如图片，GridView表格控件，用户控件等

应用程序缓存是针对应用程序内的数据缓存，如：将DataSet等数据存储到缓存

缓存的滑动机制：其实就是指缓存的相对失效时间

7，IOC（inverse Object control），依赖注入，依赖倒置，控制反转（DI（Dependency Injection）），

Q：依赖工厂，依赖高层。比如依赖于高层模块，不要直接new。就是引用本身不依赖

说白了，就是new的时候通过外部容器来。

如果从应用程序的角度就叫IOC（控制反转）；如果从容器的角度来说就叫DI（依赖注入）。

8，AOP，面向切面编程 （这里可以对比webform和mvc谈里面的过滤器，前者HttpModule和后者的Filter），就可以提到比如日志处理，权限处理等，顺便就可以提log4net

通俗：说白了，就是在调用目标事件之前先执行事件委托。

9，进程外Session

Q：出来的目的：在访问量很大的时候，Session会造成数据丢失，那么我们就用另外一个

10，会口述一些模式和设计的内涵：反射抽象工厂，面向接口编程，门面模式Facade，归约模式，

11，常用的提高程序性能的办法

从两个角度来思考：（实际上我们的程序性能的瓶颈一般都在数据库上面，因为数据库比较耗费CPU，内存和IO，而不在程序端，因此我们首先要从数据库角度来做性能优化，而数据库优化也是个很研究很深的问题，我这里可以从我所知道的层面上来说一些解决方案）（我们要从CPU，内存，IO，宽带）

1，从数据库角度：

1.1，给数据库建索引，加快检索查询速度。（但是尽量不要在主库上添加索引，因为主库主要是用来增删改的，从库中的查询可以通过索引来提高检索速度）

1.2，分库，分表。

1.2.1，分库：数据库主从分离

1.2.2，分表：减少表中的数据量，就可以在某种程度上加快查询数据。（当然了，数据库服务器和web站点分离，这是最基本的，让他们不相互争抢资源）

1.3，假如是SQL脚本查询，SQL语句优化，可以优化join的方式，

比如通过冗余字段，尽可能减少表之间的Join的次数。因为一般情况下，

当表的join个数超过3个，在大访问量的时候，基本上就可以废掉了。

1.4，减少使用主外键关系，因为在更新数据时，外键在会自动更新对应的表的数据。我们应该尽可能的采用逻辑外键，减少校验的过程。

1.5，使用更高性能的数据库，NoSql引入，解放SQLServer。当然啦，因为一般情况下，是不会轻易是更换数据库的，但是某些情况下这可以作为一种解决问题的思路

1.5.1，比如：可以尝试采用现在比较流行的非关系型数据库，NoSQL----MongoDB,它是内存型的键值对型的数据库（key，value），因而性能很高

1.5.2，还比如 Redis，它主要是将数据全部放到内存中，因此，它非常高效，现在的新浪微博就是用的这玩意

1.6，尽量减少和数据库的交互，尽量实现批量提交数据。比如将一系列的sql语句放到一个队列里，然后从队列里批量提，取后再批量和数据库交互。

1.7，做集群 （站点集群？数据库集群？）

1.7.1，合理分配服务，避免资源竞争。比如应用集群方案NLB（NetWork Loading Balance，网络负载均衡）

1.7.2，数据库集群读写分离（主库Main DB和 从库Slav DB）。比如增删改的操作放在主DB上，查询的操作放在从DB上，这样就减小了他们各自访问的压力

关于集群不是特别了解，只是看过一些相关的文章，因为我之前的开发还没设计到如此超大并发的情况。

1.7.3，垂直分库，水平分库。 前者需要程序在设计阶段就将模块设计的低耦合状态。后者比如按时间，这个月的数据消息在这台库上，下个月的数据放在另一台库上。

2，从程序端的角度：

2.1，使用缓存，减少每次访问服务器都去都去读取数据量的性能消耗。

2.2，页面静态化，减少服务器端运算所带来的消耗，这样就可以尽可能的减少连接数据库。

2.3，使用离线型的类型，实现懒加载（lazy Load），延迟加载。假如是EF领域开发模式，则可以使用延迟加载机制。

使用返回IQueryable---这是一个离线类型的。另外我们还可以优化linq。

2.4，SOA实践（Service Oriented Architecture，即面向服务架构），比如将不同的应用服务部署在不同的机器，这样当并发访问的时候，就可以减少压力。

2.5，将IIS配置成允许10W的访问，这样可以提高一点访问性能。

2.6，使用异步方式，用异步单独用线程去进行异步操作。只要设计到高并发，我们就咬尽可能使用异步编程。

12，关于委托，事件，反射，多线程等相关的东西？

如何解决死锁：   
1，操作资源的时候，尽量遵循操作顺序，不要打乱顺序   
2，Select查询的时候（默认是加上了一个S锁），而且还有可能为X锁（拍他锁）—这种锁会排除其他请求，其他所有的请求会一直等待   
解决的办法：所有的查询都不加锁 ，例如：Select \* from Table with （nolock）   
另外可以将锁的级别降到行级别，具体命令我也记不住

发布订阅机制：可以来解决**[数据库](http://lib.csdn.net/base/14" \o "MySQL知识库" \t "http://blog.csdn.net/vfush/article/details/_blank)**的主从两个库之间的同步   
但它也有缺陷：有一个10秒的延迟

压力测试工具：   
1，LR — Loader Rander，是惠普出的一款工具   
2，AQTime — 这个很牛X，甚至能直接看到代码中的哪一行代码性能最差，直接确定到某一行代码级别   
3，TD — 管理Bug

简历自我介绍：

熟悉分布式应用开发方式，熟悉面向服务的开发方式，亲自配置Windows NLB负载均衡、SqlServer 发布订阅机制实现数据集群数据同步。项目中应用高可用性分布式缓存、高可用性NoSql存储方案、高可用行Redis队列实现高并发访问事务处理。有相当的部署SOA的经验，以及一些分布式项目开发的经验和优化的经验。   
熟悉微软的Asp.Net MVC+EF +IOC+AOP+多层+WCF?开发中大型web网站技术，熟悉SOA分布式应用开发，熟悉WCF，在金和软件从事大型SOA平台开发经验   
熟练掌握编写sql语句，编写简单存储过程。熟悉?Sqlserver2005/2008、Oracle10g。熟悉ORM技术，对于微软的Entity Framework技术做过深入了解，对于基础知识Ado.Net熟练掌握！Linq更是非常熟练运用。   
熟悉**[spring](http://lib.csdn.net/base/17" \o "Java EE知识库" \t "http://blog.csdn.net/vfush/article/details/_blank)**.Net、Nhibernate等开源框架，对设计模式有自己独到见解   
能够熟练运用**[JavaScript](http://lib.csdn.net/base/18" \o "JavaScript知识库" \t "http://blog.csdn.net/vfush/article/details/_blank)**操作Dom进行web前端的动态开发。熟悉JQuary。熟练使用Css+Div对整个网页进行布局，熟练使用float和clear对div进行布局。对position也有深入的理解。   
理解Asp.Net?生命周期，熟悉Http协议。深刻理解web开发的请求、处理、响应模型，熟悉asp.net基础性原理知识。   
熟悉AspNet MVC开发技术，深入理解视图引擎处理机制，粗略阅读AspNet MVC源码，非常喜欢Razor引擎对于前台编码的优化   
熟悉 静态化设计、有SEO方面的经验   
有相关互联网开发经验，对于研究过大型Web网站部署以及**[架构](http://lib.csdn.net/base/16" \o "大型网站架构知识库" \t "http://blog.csdn.net/vfush/article/details/_blank)**方面知识，对于负载均衡、分布式缓存、集群也有相关自己的认识。

SQL 面试题，去网上找一些题目

什么是多态？

什么是面向对象？

什么是持久化？

请讲一下IsPostBack的原理？

请描述一下Nhibernate的对象状态管理和生命周期? （Nhibernate中的POCO对象状态）

请描述一下Nhibernate中有哪几种集合？他们各自的区别是什么？（Bag，Set，List，Map）

----=================================================

很多了，数据库方面问的很多。

千万级数据，删除100万行如何最高效。   
写一个linq语句，要求三层嵌套。   
使用递归写一个打印输出（这个我想了很久，差点都放弃了，最后勉强想出了一个思路，主要处在那种环境中感觉头脑晕晕的。）   
还有自己使用的设计模式，并进行讲解。   
自己的特长等。?

答案:下面的经验是针对SQL Server的，但SYBASE也是同理。希望对你有帮助。

我们在SQL Server上面删除1.6亿条记录，不能用truncate（因为只是删除其中部分数据）。经过实验，每次删除400万条要花1.5 - 3小时，而且是越到后面越慢，正常的话，需要大约102个小 时，大约4天半时间。这在生产环境下是不能接受的。

经过一个处理之后，我每次删除400万条记录花5 - 6分钟，删除全部1.6亿条记录花了4 - 5个小时！

为什么？？

第一:每次删除记录，数据库都要相应地更新索引，这是很慢的IO操作。而且后面索引碎片越来越多，就更慢。这就是为什么一开始只花1.5小时，后面要3小时才能删除400万条记录的原因。

我在删除前先保存当前索引的DDL，然后删除其索引，   
然后根据使用的删除条件建立一个临时的索引（这是提高速度的另外一个重要原因！）   
开始删除操作，完成之后再重建之前的索引。

第二:删除的时候，不要再记录日志的模式下，否则日志要爆.并且索引越少，删除速度越快!

打电话过来让你去面试，公司环境不错，中环大厦20几楼，记不清了。先是做题，一共3到，一个SQL查询，按时间横向显示，time字段24小时的记录   
数据格式   
DATE value   
查询结果每小时有多少次评论   
2题，字符串排序   
sdfrtger字符串，能有多少种不同顺序的显示   
3题，将一组数字，跟加减乘除小括号迅运算符组合，能得到的结果等于24的，公式有多少种

问了不少，数据库的查询，类继承，泛型，静态属性与常量有什么区别，varchar与nvarchar有什么区别，流行的JS框架，重要的是WCF，他们的开发商用JS+WCP开发的?

早上9点，来到双井的京城几点大厦。首先见到了HR，凡客的部门间没有明确的隔断区分，HR就坐在门口。领到了个人信息表，开始填写。写完交给HR，说安排面试。很快一个技术人员就来接 待我。我被安排到了一个员工餐厅，问了一些技术方面的问题。主要是关于**[大数据](http://lib.csdn.net/base/20" \o "Hadoop知识库" \t "http://blog.csdn.net/vfush/article/details/_blank)**量数据处理的。我回答说没有太多这方面的经验。考官也问了我是否有什么想问的，我简单问了一下凡客的软 件架构。于是离开。

一共四轮，HR，技术Lead，部门经理和副总裁。没有笔试题，技术部分就是聊工作经验，项目心得，然后写了几个简单的SQL，问了问设计模式，总体来讲比较简单。部门经理和副总裁比较关 注职业规划，未来想往什么方向发展，还有对于合作和领导方式的看法。最后HR常规性的谈谈，就结束了。

说说值对象与引用对象的区别？?

简单的算法，sql查一张表重复的数量，委托，反射，多线程。ViewState?

Nhibernate 中POCO状态对象?

瞬时状态   
持久状态   
托管状态

数据库在大数据量或者是大并发的情况下，一般表与表之间不建立真正的主外键关系，而只建立逻辑外键。在Nhibernate中采用\*.hbm.xml配置文件来关联表与表的关系！那么用EF做持久层的 时候，如何在程序中将表与表的模型关联到一起？

说一说面向对象？ 说一说你对多态的理解？

能不能手写SQL脚本？手写function？手写事务？

Http协议？

构造函数重载和运算符重载？ 他们分别是干什么的？

前绑定和后绑定？

谈谈你对架构的理解？

前绑定—编译的时候就确定调用那个办法（重载是前绑定）

后绑定—运行的时候才知道调用哪个（多态就是后绑定）