C# Interview Questions

deepwaterooo

2021 年 12 月 15 日

目录

1 C# Interview Questions

1.1 100 道 C# 面试题

1.1.1 .NET 和 C# 有什么区别

- .NET 一般指.NET FrameWork 框架，它是一种平台，一种技术。
- C# 是一种编程语言，可以基于.NET 平台的应用。

1.1.2 一列数的规则如下: 1、1、2、3、5、8、13、21、34..... 求第 30 位数是多少，用递归算法实现。

```
public class MainClass  {
    public static void Main()  {
        Console.WriteLine(Foo(30));
    }
    public static int Foo(int i)  {
        if (i <= 0)
            return 0;
        else if (i > 0 && i <= 2)
            return 1;
        else
            return Foo(i -1) + Foo(i - 2);
    }
}
```

1.1.3 C# 中的委托是什么？事件是不是一种委托？

- 委托可以把一个方法作为参数代入另一个方法。
- 委托可以理解为指向一个函数的引用。
- 是，是一种特殊的委托

1.1.4 简述 **private**、**protected**、**public**、**internal** 修饰符的访问权限。

- private : 私有成员, 在类的内部才可以访问。
- protected: 保护成员，该类内部和继承类中可以访问。
- public: 公共成员，完全公开，没有访问限制。
- internal: 在同一命名空间内可以访问。

1.1.5 **override** 与重载的区别

- override 与重载的区别。重载是方法的名称相同。参数或参数类型不同，进行多次重载以适应不同的需要
- Override 是进行基类中函数的重写。为了适应需要。

1.1.6 如果在一个 **B/S** 结构的系统中需要传递变量值，但是又不能使用 **Session**、**Cookie**、**Application**，您有几种方法进行处理？

- this.Server.Transfer

1.1.7 请编程遍历页面上所有 **TextBox** 控件并给它赋值为 **string.Empty**？

```
foreach (System.Windows.Forms.Control control in this.Controls) {
    if (control is System.Windows.Forms.TextBox) {
        System.Windows.Forms.TextBox tb = (System.Windows.Forms.TextBox)control ;
        tb.Text = String.Empty ;
    }
}
```

1.1.8 请编程实现一个选择排序算法？

```
public static int min;
public static void SelectionSort(int[] arr) { // 选择排序，从左往右依次增大
    for (int i = 0; i < arr.Length - 1; i++) {
        min = i;
        for (int j = i + 1; j < arr.Length; j++) {
            if (arr[j] < arr[min])
                min = j;
        }
        int tmp = arr[i];
        arr[i] = arr[min];
        arr[min] = tmp;
    }
}
```

1.1.9 请编程实现一个冒泡排序算法？

```
// int [] array = new int [*] ; // 为什么需要这样一行 ?
public static int tmp;
public static void BubbleSort(int [] A) {
    for (int j = 0; j < A.Length - 1; j++) {
        for (int i = j + 1; i < A.Length; i++) {
            if (A[i] < A[j]) {
                tmp = A[i];
                A[i] = A[j];
                A[j] = tmp;
            }
        }
    }
}
```

1.1.10 描述一下 C# 中索引器的实现过程，是否只能根据数字进行索引？

- 不是。可以用任意类型。

1.1.11 求以下表达式的值，写出您想到的一种或几种实现方法：1-2+3-4+……+m

```
int Num = this.TextBox1.Text.ToString() ;
int Sum = 0 ;
for (int i = 0 ; i < Num + 1 ; i++) {
    if (i % 2 == 0) {
        Sum += i ;
    } else { // i % 2 == 1
        Sum = Sum - i ;
    }
}
System.Console.WriteLine(Sum.ToString());
System.Console.ReadLine() ;
```

1.1.12 在下面的例子里

```
using System;
class A {
    public A() {
        //System.Console.WriteLine("A constructor: ");
        PrintFields();
    }
    public virtual void PrintFields() {
        //System.Console.WriteLine("A PrintFields: ");
    }
}
class B: A {
    int x = 1;
    int y;
    public B() {
        //System.Console.WriteLine("B constructor: ");
        y = -1;
        PrintFields(); // 需要这一行，要不然结果不一样
    }
    public override void PrintFields() {
        //System.Console.WriteLine("B: PrintFields");
        Console.WriteLine("x = {0}, y = {1}", x, y);
    }
}
```

- 当使用 new B() 创建 B 的实例时，产生什么输出？
- X = 1, Y = 0; x = 1, y = -1
- 代码执行顺序如下：

```
A constructor:
B: PrintFields
x = 1, y = 0
B constructor:
B: PrintFields
x = 1, y = -1
```

1.1.13 下面这段代码的执行结果是什么 1

```
public class A {
    public virtual void Fun1(int i) {
        //Console.WriteLine("A Fun1: ");
        Console.WriteLine(i);
    }
```

```
    public void Fun2(A a) {
        //Console.WriteLine("A Fun2: ");
        a.Fun1(1);
        Fun1(5);
    }
}
public class B : A {
    public override void Fun1(int i) {
        //Console.WriteLine("B Fun1: ");
        base.Fun1(i + 1);
    }
    public static void Main() {
        A a = new A();
        B b = new B();
        a.Fun2(b);
        b.Fun2(a);
        Console.Read();
    }
}
```

- 2，5，1，6

```
A Fun2:
B Fun1:
A Fun1:
2
A Fun1:
5
A Fun2:
A Fun1:
1
B Fun1:
A Fun1:
6
```

1.1.14 下面这段代码的执行结果是什么 2

```
class Class1  {
    private string str = "Class1.str";
    private int i = 0;
    static void StringConvert(string str)  {
        str = "string being converted.";
    }
    static void StringConvert(Class1 c)  {
        c.str = "string being converted.";
    }
    static void Add(int i)  {
        i++;
    }
    static void AddWithRef(ref int i)  {
        i++;
    }
    static void Main()  {
        int i1 = 10;
        int i2 = 20;
        string str = "str";
        Class1 c = new Class1();
        Add(i1);
        //Console.WriteLine("i1: " + i1);
        AddWithRef(ref i2);
        //Console.WriteLine("i2: " + i2);
        Add(c.i); // c.i = 1
        //Console.WriteLine("c.i: " + c.i);
        StringConvert(str);
        //Console.WriteLine("str: " + str);
        StringConvert(c);
        //Console.WriteLine("c.str: " + c.str);
        Console.WriteLine(i1);
        Console.WriteLine(i2);
        Console.WriteLine(c.i);
        Console.WriteLine(str);
```

```
        Console.WriteLine(c.str);
    }
}
```

- 10，21，0，str，string being converted.

```
i1: 10
i2: 21
c.i: 0
str: str
c.str: string being converted.
10
21
0
str
string being converted.
```

1.1.15 CTS、CLS、CLR 分别作何解释？

- CTS：Common Type System, 通用语言系统。
- CLS：Common Language Specification, 通用语言规范。
- CLR：Common Language Runtime, 公共语言运行库。

1.1.16 什么是装箱和拆箱？

- 从值类型接口转换到引用类型装箱。从引用类型转换到值类型拆箱。

1.1.17 什么是受管制的代码？

- unsafe：非托管代码。不经过 CLR 运行。

1.1.18 什么是强类型系统？

- RTTI：类型识别系统。

1.1.19 NET 中读写数据库需要用到那些类？他们的作用？

- DataSet: 数据存储器。
- DataCommand: 执行语句命令。
- DataAdapter: 数据的集合，用语填充。

1.1.20 列举 ASP.Net 页面之间传递值的几种方式。

- 1). 使用 QueryString, 如....?id=1; response. Redirect()....
- 2). 使用 Session 变量
- 3). 使用 Server.Transfer

1.1.21 什么是 Code-Behind 技术？

- 代码后植。

1.1.22 在.net 中，配件的意思是？

- 程序集。（中间语言，源数据，资源，装配清单）

1.1.23 常用的调用 WebService 的方法有哪些？

- 1. 使用 WSDL.exe 命令行工具。
- 2. 使用 VS.NET 中的 Add Web Reference 菜单选项

1.1.24 .NET Remoting 的工作原理是什么？

- 服务器端向客户端发送一个进程编号，一个程序域编号，以确定对象的位置。

1.1.25 在 C ＃中，string str = null 与 string str = “” 请尽量使用文字或图象说明其中的区别。

- string str = null 是不给他分配内存空间,
- string str = "" 给它分配长度为空字符串的内存空间。

1.1.26 请详述在.NET 中类 (class) 与结构 (struct) 的异同？

- Class 可以被实例化, 属于引用类型, 是分配在内存的堆上的
- Struct 属于值类型, 是分配在内存的栈上的

1.1.27 分析以下代码，完成填空

```
string strTmp = "abcdefg 某某某";
int i = System.Text.Encoding.Default.GetBytes(strTmp).Length;
int j = strTmp.Length;
```

- 以上代码执行完后，i= j=

```
i: 16 // 我的运行结果
j: 10
// i = 13, j = 10 // 为什么会是这样呢?
```

1.1.28 SQLSERVER 服务器中，给定表 table1 中有两个字段 ID、LastUpdateDate，ID 表示更新的事务号，LastUpdateDate 表示更新时的服务器时间，请使用一句 SQL 语句获得最后更新的事务号

```
Select ID FROM table1
Where LastUpdateDate = (Select MAX(LastUpdateDate) FROM table1)
```

1.1.29 简要谈一下您对微软.NET 构架下 remoting 和 webservice 两项技术的理解以及实际中的应用。

- WS 主要是可利用 HTTP，穿透防火墙。
- 而 Remoting 可以利用 TCP/IP，二进制传送提高效率。

1.1.30 公司要求开发一个继承 System.Windows.Forms.ListView 类的组件，要求达到以下的特殊功能：点击 ListView 各列列头时，能按照点击列的每行值进行重排视图中的所有行(排序的方式如 DataGrid 相似)。根据您的知识，请简要谈一下您的思路

- 根据点击的列头, 包该列的 ID 取出, 按照该 ID 排序后, 再给绑定到 ListView 中。

1.1.31 写出一条 Sql 语句：取出表 A 中第 31 到第 40 记录（SQLServer, 以自动增长的 ID 作为主键, 注意：ID 可能不是连续的。

```
select top 10 * from A
where id not in (select top 30 id from A)
```

```
select top 10 * from A
where id > (select max(id) from (select top 30 id from A ) as A)
```

1.1.32 面向对象的语言具有 __ ____ __ _____ __ ____

- 封装、继承、多态。

1.1.33 能用 foreach 遍历访问的对象需要实现 _IEnumerable_ _GetEnumerator_

- IEnumerable 、GetEnumerator。

1.1.34 GC 是什么? 为什么要有 GC?

- GC 是垃圾收集器。程序员不用担心内存管理，因为垃圾收集器会自动进行管理。要请求垃圾收集，可以调用下面的方法之一：
  - System.gc()
  - Runtime.getRuntime().gc()

1.1.35 Strings = new String("xyz"); 创建了几个 String Object?

- 两个对象，一个是“xyx”, 一个是指向“xyx”的引用对象 s。

1.1.36 启动一个线程是用 run() 还是 start()?

- 启动一个线程是调用 start() 方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由 JVM 调度并执行。这并不意味着线程就会立即运行。run() 方法可以产生必须退出的标志来停止一个线程。

1.1.37 接口是否可继承接口? 抽象类是否可实现 (implements) 接口? 抽象类是否可继承实体类 (concrete class)?

- 接口可以继承接口。
- 抽象类可以实现 (implements) 接口，
- 抽象类是可以继承实体类，但前提是实体类必须有明确的构造函数。

1.1.38 构造器 Constructor 是否可被 override?

- 构造器 Constructor 不能被继承，因此不能重写 Overriding，但可以被重载 Overloading。

1.1.39 是否可以继承 **String** 类**?**

- String 类是 final 类故不可以继承。

1.1.40 **try{}** 里有一个 **return** 语句，那么紧跟在这个 **try** 后的 **finally {}** 里的 **code** 会不会被执行，什么时候被执行，在 **return** 前还是后**?**

- 会执行，在 return 前执行。

1.1.41 两个对象值相同 **(x.equals(y)== true)**，但却可有不同的 **hash code**，这句话对不对**?**

- 不对，有相同的 hash code。

1.1.42 **swtich** 是否能作用在 **byte** 上，是否能作用在 **long** 上，是否能作用在 **String** 上**?**

- switch（expr1）中，expr1 是一个整数表达式。因此传递给 switch 和 case 语句的参数应该是 int、short、char 或者 byte。
- long, string 都不能作用于 swtich。

1.1.43 当一个线程进入一个对象的一个 **synchronized** 方法后，其它线程是否可进入此对象的其它方法**?**

- 不能，一个对象的一个 synchronized 方法只能由一个线程访问。

1.1.44 **abstract** 的 **method** 是否可同时是 **static,** 是否可同时是 **native**，是否可同时是 **synchronized?**

- 都不能。

1.1.45 **List,Set, Map** 是否继承自 **Collection** 接口**?**

- List 和 Set 是, Map 不是

1.1.46 **Set** 里的元素是不能重复的,那么用什么方法来区分重复与否呢**?** 是用 **==** 还是 **equals()?** 它们有何区别**?**

- Set 里的元素是不能重复的，那么用 iterator() 方法来区分重复与否。
- equals() 是判读两个 Set 是否相等。
- equals() 和 == 方法决定引用值是否指向同一对象
- equals() 在类中被覆盖，为的是当两个分离的对象的内容和类型相配的话，返回真值。

1.1.47 数组有没有 **length()** 这个方法**? String** 有没有 **length()** 这个方法？

- 数组没有 length() 这个方法，有 Length 的属性。
- String 有有 length() 这个方法。

1.1.48 sleep() 和 wait() 有什么区别?

- sleep() 方法是使线程停止一段时间的方法。在 sleep 时间间隔期满后，线程不一定立即恢复执行。这是因为在那个时刻，其它线程可能正在运行而且没有被调度为放弃执行，除非
  - (a) “醒来”的线程具有更高的优先级
  - (b) 正在运行的线程因为其它原因而阻塞。
- wait() 是线程交互时，如果线程对一个同步对象 x 发出一个 wait() 调用，该线程会暂停执行，被调对象进入等待状态，直到被唤醒或等待时间到。

1.1.49 short s1 = 1; s1 = s1 + 1; 有什么错? short s1 = 1; s1 += 1; 有什么错?

- 有错，s1 是 short 型，s1+1 是 int 型, 不能显式转化为 short 型。

```
short s1 = 1; s1 = s1 + 1;
```

```
short s1 = 1;
// Cannot implicitly convert type 'int' to 'short'.
// An explicit conversion exists (are you missing a cast?)
// s1 = s1 + 1;
s1 = (short)(s1 + 1);
System.Console.WriteLine("s1: " + s1);
short s2 = 1;
s2 += 1;
System.Console.WriteLine("s2: " + s2);
```

- 可修改为

```
s1 =(short)(s1 + 1)

short s1 = 1;
s1 += 1 // 正确。
```

1.1.50 谈谈 final,finally, finalize 的区别。

- final —修饰符（关键字）如果一个类被声明为 final，意味着它不能再派生出新的子类，不能作为父类被继承。因此一个类不能既被声明为 abstract 的，又被声明为 final 的。将变量或方法声明为 final，可以保证它们在使用中不被改变。被声明为 final 的变量必须在声明时给定初值，而在以后的引用中只能读取，不可修改。被声明为 final 的方法也同样只能使用，不能重载
- finally —在异常处理时提供 finally 块来执行任何清除操作。如果抛出一个异常，那么相匹配的 catch 子句就会执行，然后控制就会进入 finally 块（如果有的话）。
- finalize —方法名。Java 技术允许使用 finalize() 方法在垃圾收集器将对象从内存中清除出去之前做必要的清理工作。这个方法是由垃圾收集器在确定这个对象没有被引用时对这个对象调用的。它是在 Object 类中定义的，因此所有的类都继承了它。子类覆盖 finalize() 方法以整理系统资源或者执行其他清理工作。finalize() 方法是在垃圾收集器删除对象之前对这个对象调用的。

1.1.51 如何处理几十万条并发数据?

- 用存储过程或事务。取得最大标识的时候同时更新.. 注意主键不是自增量方式这种方法并发的时候是不会有重复主键的.. 取得最大标识要有一个存储过程来获取.

1.1.52 Session 有什么重大 BUG，微软提出了什么方法加以解决？

- 是 iis 中由于有进程回收机制，系统繁忙的话 Session 会丢失，可以用 Sate server 或 SQL Server 数据库的方式存储 Session 不过这种方式比较慢，而且无法捕获 Session 的 END 事件。

1.1.53 进程和线程的区别？

- 进程是系统进行资源分配和调度的单位；
- 线程是 CPU 调度和分派的单位，
- 一个进程可以有多个线程，这些线程共享这个进程的资源。

1.1.54 堆和栈的区别？

- 栈：由编译器自动分配、释放。在函数体中定义的变量通常在栈上。
- 堆：一般由程序员分配释放。用 new、malloc 等分配内存函数分配得到的就是在堆上。

1.1.55 成员变量和成员函数前加 static 的作用？

- 它们被称为常成员变量和常成员函数，又称为类成员变量和类成员函数。分别用来反映类的状态。比如类成员变量可以用来统计类实例的数量，类成员函数负责这种统计的动作。

1.1.56 ASP。NET 与 ASP 相比，主要有哪些进步？

- asp 解释形，aspx 编译型，性能提高，有利于保护源码。

1.1.57 请说明在.net 中常用的几种页面间传递参数的方法，并说出他们的优缺点。

- session(viewstate) 简单，但易丢失
- application 全局
- cookie 简单，但可能不支持，可能被伪造
- inputtype = "hidden" 简单，可能被伪造
- url 参数简单，显示于地址栏，长度有限
- 数据库稳定，安全，但性能相对弱

1.1.58 请指出 GAC 的含义？

- GAC: Global Assembly Cache
- 全局程序集缓存。

1.1.59 向服务器发送请求有几种方式？

- get, post。get 一般为链接方式，post 一般为按钮方式。

1.1.60 DataReader 与 Dataset 有什么区别？

- 一个是只能向前的只读游标，一个是内存中虚拟的数据库。

1.1.61 软件开发过程一般有几个阶段？每个阶段的作用？

- 需求分析，架构设计，代码编写，QA，部署

1.1.62 在 **c#** 中 **using** 和 **new** 这两个关键字有什么意义，请写出你所知道的意义？**using** 指令和语句 **new** 创建实例 **new** 隐藏基类中方法。

- using 引入名称空间或者使用非托管资源
- new 新建实例或者隐藏父类方法

1.1.63 需要实现对一个字符串的处理，首先将该字符串首尾的空格去掉，如果字符串中间还有连续空格的话，仅保留一个空格，即允许字符串中间有多个空格，但连续的空格数不可超过一个.

```
using System.Text.RegularExpressions;
string inputStr = "  xx    xx  ";
inputStr = Regex.Replace(inputStr.Trim(),"\\s+", " ");
// System.Console.WriteLine("inputStr: A" + inputStr + "A;");
// inputStr: Axx xxA;

string input = "This is   text with   far  too   much   " + "whitespace.";
string pattern = "\\s+";
string replacement = " ";
Regex rgx = new Regex(pattern);
string result = rgx.Replace(input, replacement);
Console.WriteLine("Original String: {0}", input);
Console.WriteLine("Replacement String: {0}", result);
// The example displays the following output:
// Original String:    This is   text with   far  too   much   whitespace.
// Replacement String: This is text with far too much whitespace.
```

- 正则表达式符号描述：http://www.jb51.net/article/73929.htm
  - § 匹配任何非空白字符
  - 配一个制表符
  - ˇ配一个垂直制表符。等价于 0b 和
  - 匹配包括下划线的任何单词字符。等价于'[A-Za-z0-9]'
  - 匹配任何非单词字符。等价于'[^{A}-Za-z0-9]'

1.1.64 什么叫做 **SQL** 注入，如何防止？请举例说明。

- 利用 sql 关键字对网站进行攻击。过滤关键字等
- 利用 sql 语言漏洞获得合法身份登陆系统。如身份验证的程序设计成：

```
SqlCommand com = new SqlCommand("Select * from users where username =  '"+t_name.text+" ' and pwd= '"+t_pwd.text+" '"
object obj=com.ExcuteScale();
if (obj!=null) {
    // 通过验证
}
```

- 这段代码容易被 sql 注入。如用户在 t_{name} $_{tpwd\ 1'}$ and 1= '1 就可以进入系统了。

1.1.65 什么是反射？

- 动态获取程序集信息

1.1.66 用 **Singleton** 如何写设计模式

- static 属性里面 new , 构造函数 private

1.1.67 什么是 **Application Pool**?

- Web 应用，类似 Thread Pool，提高并发性能。

1.1.68 什么是虚函数？什么是抽象函数？

- 虚函数：没有实现的，可由子类继承并重写的函数。
- 抽象函数：规定其非虚子类必须实现的函数，必须被重写。

1.1.69 什么是 **XML**?

- XML 即可扩展标记语言。eXtensible Markup Language.
- 标记是指计算机所能理解的信息符号，通过此种标记，计算机之间可以处理包含各种信息的文章等。
- 如何定义这些标记，即可以选择国际通用的标记语言，比如 HTML，也可以使用象 XML 这样由相关人士自由决定的标记语言，这就是语言的可扩展性。
- XML 是从 SGML 中简化修改出来的。它主要用到的有 XML、XSL 和 XPath 等。

1.1.70 什么是 **WebService**？**UDDI**?

- Web Service 便是基于网络的、分布式的模块化组件，它执行特定的任务，遵守具体的技术规范，这些规范使得 Web Service 能与其他兼容的组件进行互操作。
- UDDI 的目的是为电子商务建立标准；UDDI 是一套基于 Web 的、分布式的、为 WebService 提供的、信息注册中心的实现标准规范，同时也包含一组使企业能将自身提供的 Web Service 注册，以使别的企业能够发现的访问协议的实现标准。

1.1.71 什么是 **ASP.net** 中的用户控件？

- 用户控件一般用在内容多为静态, 或者少许会改变的情况下.. 用的比较大.. 类似 ASP 中的 include.. 但是功能要强大的多。

1.1.72 列举一下你所了解的 **XML** 技术及其应用

- xml 用于配置, 用于保存静态数据类型. 接触 XML 最多的是 web Services.. 和 config

1.1.73 **ADO.net** 中常用的对象有哪些？分别描述一下。

- Connection 数据库连接对象
- Command 数据库命令
- DataReader 数据读取器
- DataSet 数据集

1.1.74 什么是 **code-Behind** 技术。

- ASPX, RESX 和 CS 三个后缀的文件，这个就是代码分离. 实现了 HTML 代码和服务器代码分离. 方便代码编写和整理.

1.1.75 什么是 **SOAP,** 有哪些应用。

- simple object access protocal, 简单对象接受协议. 以 xml 为基本编码结构, 建立在已有通信协议上 (如 http, 不过据说 ms 在搞最底层的架构在 tcp/ip 上的 soap) 的一种规范 WebService 使用的协议..
- SOAP（Simple Object Access Protocol ）简单对象访问协议是在分散或分布式的环境中交换信息并执行远程过程调用的协议，是一个基于 XML 的协议。使用 SOAP，不用考虑任何特定的传输协议（最常用的还是 HTTP 协议），可以允许任何类型的对象或代码，在任何平台上，以任何一直语言相互通信。这种相互通信采用的是 XML 格式的消息。
- SOAP 也被称作 XMLP，为两个程序交换信息提供了一种标准的工作机制。在各类机构之间通过电子方式相互协作的情况下完全有必要为此制定相应的标准。
- SOAP 描述了把消息捆绑为 XML 的工作方式。它还说明了发送消息的发送方、消息的内容和地址以及发送消息的时间。SOAP 是 Web Service 的基本通信协议。SOAP 规范还定义了怎样用 XML 来描述程序数据（Program Data）, 怎样执行 RPC（Remote Procedure Call）。大多数 SOAP 解决方案都支持 RPC-style 应用程序。SOAP 还支持 Document-style 应用程序（SOAP 消息只包含 XML 文本信息）。
- 最后 SOAP 规范还定义了 HTTP 消息是怎样传输 SOAP 消息的。MSMQ、SMTP、TCP/IP 都可以做 SOAP 的传输协议。
- SOAP 是一种轻量级协议，用于在分散型、分布式环境中交换结构化信息。SOAP 利用 XML 技术定义一种可扩展的消息处理框架，它提供了一种可通过多种底层协议进行交换的消息结构。这种框架的设计思想是要独立于任何一种特定的编程模型和其他特定实现的语义。
- SOAP 定义了一种方法以便将 XML 消息从 A 点传送到 B 点。为此，它提供了一种基于 XML 且具有以下特性的消息处理框架：1) 可扩展，2) 可通过多种底层网络协议使用，3) 独立于编程模型。

1.1.76 C# 中 **property** 与 **attribute** 的区别，他们各有什么用处，这种机制的好处在哪里？

- 一个是属性，用于存取类的字段，一个是特性，用来标识类，方法等的附加性质

1.1.77 **XML** 与 **HTML** 的主要区别

- 1. XML 是区分大小写字母的，HTML 不区分。
- 2. 在 HTML 中，如果上下文清楚地显示出段落或者列表键在何处结尾，那么你可以省略 </p> 或者 之类的结束标记。在 XML 中，绝对不能省略掉结束标记。
- 3. 在 XML 中，拥有单个标记而没有匹配的结束标记的元素必须用一个 / 字符作为结尾。这样分析器就知道不用查找结束标记了。
- 4. 在 XML 中，属性值必须分装在引号中。在 HTML 中，引号是可用可不用的。
- 5. 在 HTML 中，可以拥有不带值的属性名。在 XML 中，所有的属性都必须带有相应的值。

1.1.78 c# 中的三元运算符是？

- ?：

1.1.79 当整数 a 赋值给一个 object 对象时，整数 a 将会被？

- 装箱。

1.1.80 类成员有 ____

- this.;new Class().Method;

1.1.81 public static const int A = 1; 这段代码有错误么？是什么？

- const 不能用 static 修饰。

1.1.82 float f = -123.567F; int i = (int)f; i 的值现在是 _____?

- -123。

1.1.83 委托声明的关键字是 $_{delegate}$__?

- delegate.

1.1.84 用 sealed 修饰的类有什么特点？

- 密封，不能继承。

1.1.85 在 Asp.net 中所有的自定义用户控件都必须继承自 $_{Control}$__?

- Control。

1.1.86 在.Net 中所有可序列化的类都被标记为 $_{[serializable]}$__?

- [serializable]

1.1.87 在.Net 托管代码中我们不用担心内存漏洞，这是因为有了 $_{GC}$__?

- GC。

1.1.88 当类 T 只声明了私有实例构造函数时，则在 T 的程序文本外部，__ __ or 不可以）从 T 派生出新的类，_ _ or 不可以）直接创建 T 的任何实例。

- 不可以，不可以。

1.1.89 下面这段代码有错误么？

```
switch(i){
case(): // case() 条件不能为空
    CaseZero();
    break;
case1:
    CaseOne();
    break;
case2:
default:
```

```
    //dufault; //wrong, 格式不正确
    CaseTwo();
    break;
}
```

1.1.90 在.Net 中，类 System.Web.UI.Page 可以被继承么？

- 可以。

1.1.91 .net 的错误处理机制是什么？

- net 错误处理机制采用 try -> catch -> finally 结构，发生错误时，层层上抛，直到找到匹配的 Catch 为止。

1.1.92 利用 operator 声明且仅声明了 ==，有什么错误么?

- 要同时修改 Equale 和 GetHash() ?
- 重载了"=" 就必须重载"!"

1.1.93 在.net（C# or vb.net）中如何取消一个窗体的关闭。

```
private void Form1_Closing(object sender,
                           System.ComponentModel.CancelEventArgse) {
    e.Cancel = true;
}
```

1.1.94 在.net（C# or vb.net）中，Appplication.Exit 还是 Form.Close 有什么不同？

- 一个是退出整个应用程序，一个是关闭其中一个 form。

1.1.95 某一密码仅使用 K、L、M、N、O 共 5 个字母，密码中的单词从左向右排列，密码单词必须遵循如下规则：

- （1）密码单词的最小长度是两个字母，可以相同，也可以不同
- （2）K 不可能是单词的第一个字母
- （3）如果 L 出现，则出现次数不止一次
- （4）M 不能是最后一个, 也不能是倒数第二个字母
- （5）K 出现，则 N 就一定出现
- （6）O 如果是最后一个字母，则 L 一定出现
- 问题一：下列哪一个字母可以放在 LO 中的 O 后面，形成一个 3 个字母的密码单词？
  - A)K B)L C) M D) N
  - 答案:B
- 问题二：如果能得到的字母是 K、L、M，那么能够形成的两个字母长的密码单词的总数是多少？
  - A)1 个 B)3 个 C)6 个 D）9 个
  - 答案:A

- 问题三：下列哪一个是单词密码？
  - A)KLLN B) LOML C) MLLO D)NMKO
  - 答案:C

1.1.96 62-63=1 等式不成立，请移动一个数字（不可以移动减号和等于号），使得等式成立，如何移动？

- 62 移动成 2 的 6 次方 2^6 - 63 = 1

1.1.97 C# 中 property 与 attribute 的区别，他们各有什么用处，这种机制的好处在哪里？

- attribute: 自定义属性的基类;
- property: 类中的属性

1.1.98 在 C # 中，string str = null 与 string str = "" 请尽量使用文字或图象说明其中的区别。

- null 是没有空间引用的；
- "" 是空间为 0 的字符串；

1.1.99 abstract class 和 interface 有什么区别?

- 声明方法的存在而不去实现它的类被叫做抽像类（abstract class），它用于要创建一个体现某些基本行为的类，并为该类声明方法，但不能在该类中实现该类的情况。
  - 不能创建 abstract 类的实例。然而可以创建一个变量，其类型是一个抽像类，并让它指向具体子类的一个实例。
  - 不能有抽像构造函数或抽像静态方法。
  - Abstract 类的子类为它们父类中的所有抽像方法提供实现，否则它们也是抽像类。取而代之，在子类中实现该方法。知道其行为的其它类可以在类中实现这些方法。
- 接口（interface）是抽像类的变体。
  - 在接口中，所有方法都是抽像的。多继承性可通过实现这样的接口而获得。接口中的所有方法都是抽像的，没有一个有程序体。
  - 接口只可以定义 static final 成员变量。
  - 接口的实现与子类相似，除了该实现类不能从接口定义中继承行为。当类实现特殊接口时，它定义（即将程序体给予）所有这种接口的方法。然后，它可以在实现了该接口的类的任何对像上调用接口的方法。
  - 由于有抽像类，它允许使用接口名作为引用变量的类型。通常的动态联编将生效。引用可以转换到接口类型或从接口类型转换，instanceof 运算符可以用来决定某对象的类是否实现了接口。

1.1.100 <%# %> 和 <% %> 有什么区别？

- <%# %> 表示绑定的数据源
- <%%> 是服务器端代码块

1.1.101 重载 overloading 与覆盖 overriding 的区别？

- 1、方法的覆盖是子类和父类之间的关系，是垂直关系；方法的重载是同一个类中方法之间的关系，是水平关系
- 2、覆盖只能由一个方法，或只能由一对方法产生关系；方法的重载是多个方法之间的关系。

1.1.102 Overloaded 的方法是否可以改变返回值的类型?

- Overloaded 的方法是可以改变返回值的类型。

1.1.103 C# 可否对内存进行直接的操作？

- 在.net 下，.net 引用了垃圾回收（GC）功能，它替代了程序员.
- 不过在 C# 中，不能直接实现 Finalize 方法，而是在析构函数中调用基类的 Finalize() 方法

1.1.104 根据线程安全的相关知识，分析以下代码，当调用 test 方法时 i>10 时是否会引起死锁? 并简要说明理由。

```
  public void test(int i) {
      lock (this) {
          if (i > 10) {
              i--;
              test(i);
          }
      }
  }
```

- 不会发生死锁，（但有一点 int 是按值传递的，所以每次改变的都只是一个副本，因此不会出现死锁。但如果把 int 换做一个 object，那么死锁就会发生）

1.1.105 给定以下 XML 文件,完成算法流程图。< DriverC > 请画出遍历所有文件名（FileName）的流程图 (请使用递归算法)。

```
void FindFile (Directory d) {
    FileOrFolders = d.GetFileOrFolders();
    foreach (FileOrFolder fof in FileOrFolders) {
        if( fof is File )
            You Found a file;
        else if (fof is Directory )
            FindFile (fof);
    }
}
```

1.1.106 产生一个 int 数组，长度为 100，并向其中随机插入 1-100，并且不能重复。

```
using System.Collections;
int[] intArr = new int[100];
ArrayList myList = new ArrayList();
Random rnd = new Random();
while (myList.Count < 100) {
    int num = rnd.Next(1,101);
    if (!myList.Contains(num))
        myList.Add(num);
}
for (int i = 0; i < 100; i++) {
    intArr[i] = (int)myList[i];
    System.Console.WriteLine("intArr[i]: " + intArr[i]);
}
```

1.1.107 下面的代码中有什么错误吗？_____

```
using System;
class A {
    public virtual void F(){
        Console.WriteLine("A.F");
    }
}
abstract class B: A {
    public abstract override void F(); // abstract override 是不可以一起修饰.
} // new public abstract void F(); 什么意思
```

1.1.108 对于这样的一个枚举类型：

```
enum Color: byte {
    Red,
    Green,
    Blue,
    orange
}
```

```
string[] ss = Enum.GetNames(typeof(Color));
byte[] bb = Enum.GetValues(typeof(Color));
```

1.1.109 写一个 **HTML** 页面，实现以下功能，左键点击页面时显示“您好”，右键点击时显示“禁止右键”。并在 **2** 分钟后自动关闭页面。

```
setTimeout(`window.close();`, 3000);
function show() {
    if (window.event.button == 1) {
        alert(" 左");
    } else if (window.event.button == 2) {
        alert(" 右");
    }
}
```

1.1.110 大概描述一下 **ASP**。**NET** 服务器控件的生命周期

- 初始化加载视图状态处理回发数据加载发送回发更改通知处理回发事件预呈现保存状态呈现处置卸载

1.1.111 **Anonymous Inner Class (**匿名内部类**)** 是否可以 **extends(**继承**)** 其它类，是否可以 **implements(**实现**)interface(**接口**)?**

- 不能，可以实现接口

1.1.112 **Static Nested Class** 和 **Inner Class** 的不同，说得越多越好

- Static Nested Class 是被声明为静态（static）的内部类，它可以不依赖于外部类实例被实例化。
- 而通常的内部类需要在外部类实例化后才能实例化。

1.1.113 **&** 和 **&&** 的区别。

- & 是位运算符，表示按位与运算，
- && 是逻辑运算符，表示逻辑与（and）.

1.1.114 HashMap 和 Hashtable 的区别。

- HashMap 是 Hashtable 的轻量级实现（非线程安全的实现），他们都完成了 Map 接口，
- 主要区别在于 HashMap 允许空（null）键值（key），由于非线程安全，效率上可能高于 Hashtable.

1.1.115 error 和 exception 有什么区别?

- error 表示恢复不是不可能但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况。
- exception 表示一种设计或实现问题。也就是说，它表示如果程序运行正常，从不会发生的情况。

1.1.116 你觉得 ASP.NET 2.0（VS2005）和你以前使用的开发工具（.Net 1.0 或其他）有什么最大的区别？你在以前的平台上使用的哪些开发思想（pattern / architecture）可以移植到 ASP.NET 2.0 上 (或者已经内嵌在 ASP.NET 2.0 中)

- 1 ASP.NET 2.0 把一些代码进行了封装打包, 所以相比 1.0 相同功能减少了很多代码.
- 2 同时支持代码分离和页面嵌入服务器端代码两种模式, 以前 1.0 版本,.NET 提示帮助只有在分离的代码文件, 无法在页面嵌入服务器端代码获得帮助提示,
- 3 代码和设计界面切换的时候,2.0 支持光标定位. 这个我比较喜欢
- 4 在绑定数据, 做表的分页.Update,Delete, 等操作都可以可视化操作, 方便了初学者
- 5 在 ASP.NET 中增加了 40 多个新的控件, 减少了工作量

1.1.117 描述一下 C# 中索引器的实现过程，是否只能根据数字进行索引？

- 不是。可以用任意类型。

1.1.118 分析以下代码。

```
public static void test(string ConnectString) {
    System.Data.OleDb.OleDbConnection conn = new System.Data.OleDb.OleDbConnection();
    conn.ConnectionString = ConnectString;
    try {
        conn.Open();
        //…….
    } catch(Exception Ex) {
        MessageBox.Show(Ex.ToString());
    } finally {
        if (!conn.State.Equals(ConnectionState.Closed))
            conn.Close();
    }
}
```

- 请问 1) 以上代码可以正确使用连接池吗？
- 回如果传入的 connectionString 是一模一样的话，可以正确使用连接池。不过一模一样的意思是，连字符的空格数，顺序完全一致。

1.1.119 什么是 WSE？目前最新的版本是多少？

- WSE (Web Service Extension) 包来提供最新的 WEB 服务安全保证，目前最新版本 2.0。

1.1.120 下面的例子中

```
class A {
    public static int X;
    static A() {
        //System.Console.WriteLine("A constructor: ");
        X = B.Y + 1;
        //System.Console.WriteLine("X: " + X);
    }
}
class B {
    public static int Y = A.X + 1;
    static B() {
        //System.Console.WriteLine("B constructor: ");
        //System.Console.WriteLine("Y: " + Y);

    }
    static void Main() {
        Console.WriteLine("X = {0}, Y = {1}", A.X, B.Y);
    }
}
```

- 产生的输出结果是什么?
- x = 1,y = 2

```
A constructor:
X: 1
B constructor:
Y: 2
X = 1, Y = 2
```

1.2 125/135 个基本的 C# 面试问答

- http://blog.jobbole.com/74515/

1.2.1 什么是值类型和引用类型?

- 值类型存储在堆栈中。
  - 例如: bool, byte, chat, decimal, double, enum, float, int, long, sbyte, short, strut, uint, ulong, ushort。
- 引用类型存储在托管堆中。
  - 例如: class, delegate, interface, object, string。

1.2.2 有哪些缓存的种类?

- 有三种类型的缓存:
  - 输出缓存 (Output Caching): 存储 asp.net 页面的应答信息。
  - 片段缓存 (Fragment Caching): 仅缓存/存储部分页面内容 (用户控制)。
  - 数据缓存 (Data Caching): 为了性能通过编程的方式来缓存对象。

1.2.3 自定义控件和用户控件之间的区别是什么？

- 自定义控件是编译后的代码 (Dlls)，容易使用，创建困难，可以放在工具箱。拖拉使用的控件。
- 属性可以直观地在设计时指定。可以被多个应用程序使用 (如果共享 Dlls)，即使是私有的也能拷贝到 web 应用程序的 bin 目录，添加引用和使用。通常是用来为独立的消费应用程序提供公用功能。
- 用户控件和 ASP 的 include 文件，轻松创建，不能放置在箱中来拖拉放置它。用户控件在单个应用程序文件之间共享。

1.2.4 什么是方法？

- 方法是由对象或者类执行来实现计算或者操作的成员。静态方法通过类访问。实例方法通过类的实例来访问。

1.2.5 什么是域？

- 域是类或者类的实例相关的变量。

1.2.6 什么是事件？

- 事件是使一个类或对象能够提供通知的成员。事件声明像域声明一样，除了声明包含 event 关键字并且类型必须为委托类型。

1.2.7 什么是文本和它们的类型？

- 文本是程序分配给变量的值常量。C# 支持的几种文本类型是
  - 整数 (Integer literals)
  - 实数 (Real literals)
  - 布尔值 (Boolean literals)
  - 单字符 (Single character literals)
  - 字符串 (String literals)
  - 反斜线 (Backslash character literals)

1.2.8 C# 有什么特性？

- C# 是一个简单而强大的编程语言，用于编写企业版的应用程序。
- 它是 C++ 和 VB 的混合体。它保留了许多 C++ 特性，如语句，表达式和运算符并结合了 VB 的生产力。
- C# 帮助开发者轻易地构建网络服务，能够通过任何语言，任何平台来访问 Internet。
- C# 帮助开发者用更少的代码完成开发，从而在代码中错误更少。
- C# 引入了相当大的改进和创新，如类型安全，版本控制，事件和垃圾收集这些领域。

1.2.9 错误的类型是什么？

- 语法错误 (Syntax error)
- 逻辑错误 (Logic error)
- 运行时错误 (Runtime error)

1.2.10 break 和 continue 语句之间有什么区别？

- break 语句是用来终止当前封闭循环或者它所在的条件语句的。我们已经使用 break 语句跳出 switch 语句。
- continue 语句是用来改变执行顺序的。和 break 语句那样跳出循环相反，continue 语句停止当前迭代并且只将控制返回到循环顶部。

1.2.11 定义命名空间？

- 命名空间被称为容器，用来组织分层的.NET 类。

1.2.12 什么是代码组？

- 代码组是一组共享安全上下文的套件。

1.2.13 C# 中什么是密封类？

- sealed 修饰符用来阻止从一个类派生。如果一个密封类被指定为另一个类的基类时会发生编译时错误。

1.2.14 静态方法和实例方法的区别是什么？

- 以 static 修饰符声明的方法是静态方法。静态方法不操作具体的实例，并且只能被静态成员访问。
- 没有以 static 修饰符声明的方法是实例方法。实例方法操作一个具体的实例并且可以被静态和实例成员访问。在其上调用实例方法的实例可以像这样显示访问。在静态方法中这么调用是错误的。

1.2.15 C# 中有哪些变量的类型？

- C# 中不同的变量类型是：
  - 静态变量 (static variables)
  - 实例变量 (instance variable)
  - 值参数 (value parameters)
  - 引用参数 (reference parameters)
  - 数组元素 (array elements)
  - 输出参数 (output parameters)
  - 局部变量 (local variables)

1.2.16 C# 中有什么特殊的运算符？

- C# 支持一下特殊运算符。
- is (关系运算符)
- as (关系运算符)
- typeof (类型运算符)
- sizeof (大小运算符，用于获取非托管类的大小)
- new (对象运算符)

1.2.17 dot (成员访问运算符)

- checked (溢出检查)
- unchecked?(防止溢出检查)

1.2.18 C# 中运算符的含义是什么？

- 运算符是界定了对类实例应用特定的运算表达式内涵的成员。可以定义三种类型的运算符：一元运算符，二元运算符和转换运算符。所有的运算符必须声明为 public 和 static 的。

1.2.19 什么是类型参数化？

- 类型参数化是一个类型在另一个值或者类型之上参数化。

1.2.20 抽象类的特性是什么？

- 抽象类不能被实力化，在抽象类上使用 new 操作符是错误的。
- 抽象类允许 (但不必要) 包含抽象方法和入口。
- 抽象类不能用 scaled 修饰符。

1.2.21 控制台应用程序和窗口应用程序有什么区别？

- 控制台应用程序，设计用来在没有用户界面的命令行中运行。
- 窗口应用程序，设计用来通过用户界面在用户桌面执行。

1.2.22 C# 有 throws 子句吗？

- 不，不像 Java，C# 不需要开发者指定方法可以抛出的异常。

1.2.23 C# 支持可变数目的参数吗？

- 是的，使用 params 关键字。该参数指定为特定类型的参数列表。

1.2.24 可以重写私有虚方法吗？

- 不可以，私有方法不能在类外访问。

1.2.25 什么是多播委托？

- 每个委托对象保持对一个单独方法的引用。但是，一个委托对象保持对多个方法的引用并调用它们是可能的。这样的委托对象成为多播委托或者组合委托。

1.2.26 什么是 C# 独有的特性？

- XML 文档.

1.2.27 在 C# 中使用异常是推荐的吗？

- 是的，在.NET 框架中异常是推荐的错误处理机制。

1.2.28 什么是智能导航？

- 因为服务端验证和页面被刷新导致页面刷新时，光标位置保持不变。

1.2.29 CONST 和 READONLY 的区别是什么？

- 都是为了定义常量值。const 字段只能在声明这个域的时候初始化。readonly 字段可以在声明时或者构造函数中定义。

1.2.30 C# 有 throws 子句吗？

- 不，不像 Java，C# 不需要 (甚至不允许) 开发者指定方法可以抛出的异常。

1.2.31 方法可以重载的不同方式是什么？

- 不同的参数类型，不同的参数个数，不同的参数顺序。

1.2.32 事件有返回值吗？

- 没有，事件没有返回类型。

1.2.33 事件是什么？

- 事件是一个基于另一个程序方法执行的动作。
- 事件是被对象或者类使用来通知其他对象发生的事件的委托类型类成员。
- 事件可以通过 event 关键字来声明。

1.2.34 C# 和 C++ 的区别是什么？

- C# 不支持 #include 语句。它只用 using 语句。
- C# 中，类定义在最后不使用分号。
- C# 不支持多重继承。
- 数据类型的显示转换在 C# 中比 C++ 中安全很多。
- C# 中 switch 也可用于字符串值。
- 命令行参数数组的行为在 C# 中和 C++ 中不一样。

1.2.35 C# 提供拷贝构造函数吗？

- 不，C# 不提供拷贝构造函数。

1.2.36 类或者结构可以有多个构造函数吗？

- 可以，类或者结构可以有多个构造函数。C# 中构造函数可以被重载。

1.2.37 可以创建接口的实例吗？

- 不可以，你不可以创建接口的实例。

1.2.38 接口可以包含字段吗？

- 不可以，接口不能包含字段。

1.2.39 类可以有静态构造函数吗？

- 是的，类可以有静态构造函数。静态构造函数在任何静态字段被访问之前被立即自动调用，并且通常用来初始化静态类成员。它在第一个实例被创建或者任何静态成员被引用之前自动调用。静态构造函数在实例构造函数之前调用。一个例子如下所示。

1.2.40 C# 中委托的主要作用是什么？

- 委托主要用于定义回调方法。

1.2.41 投影 (Shadowing) 和重写 (overriding) 的区别是什么？

- 重写仅仅重定义实现而投影重定义整个元素。
- 重写派生类可以通过“ME”关键字引用父类元素，但投影中你可以通过”MYBASE”访问父类元素。

1.2.42 事件可以用访问修饰符吗？

- 可以，你可以在事件中用访问修饰符。你可以对事件使用 portected 关键字，这样只允许继承类访问。你可以使用 private 修饰事件，仅可以供当前类对象访问。

1.2.43 什么是构造函数和析构函数？

- 构造函数和析构函数是特殊的方法。
- 构造函数和析构函数是每个类的特殊方法。
- 每个类都有自己的构造函数和析构函数，并且在类实例被创建或者销毁时自动调用。
- 每当你访问类的时候，构造函数就初始化所有类成员。当对象不想再需要的时候，析构函数就销毁它们。

1.2.44 我们怎么抑制 finalize 方法？

- GC.SuppressFinalize()。

1.2.45 C# 支持可变数目的参数吗？

- 是的，使用 params 关键字。
- 该参数指定为特定类型的参数列表，例如，int。为了最大的灵活性，类型可以是 object。
- 使用这种方法的标准例子是 System.console.writeLine()。

1.2.46 哪个方法用来启动一个线程？

- Start。

1.2.47 什么是泛型？

- 泛型帮助我们创建灵活的强类型集合。
- 泛型基本上从数据类型中分离了逻辑，以保持更好的可重用性，更好的可维护性等等。

1.2.48 有哪些不同种类的多态性？

- 有两种类型的多态，它们是：
- 编译时多态性
- 运行时多态性

1.2.49 编译时多态性和运行时多态性的区别是什么？

- 编译时多态性
- 编译时多态性也被称为方法重载。
- 方法重载是指有两个或更多同名但含有不同签名的方法。
- 运行时多态性
- 运行时多态性也被称为方法重写。
- 方法重写是指有两个或更多的同名方法，含有相同的方法签名但对应不同的实现。

1.2.50 哪一个命名空间使 XML 中多线程编程可行？

- System.Threading。

1.2.51 在 C# 中可以声明一个静态块吗？

- 不可以，因为 C# 不支持静态块，但它支持静态方法。

1.2.52 方法可以声明为密封 (sealed) 吗？

- 在 C# 中方法不可以声明为 sealed。但当我们在派生类重写一个方法的时候，我们可以将重写的方法定义为 sealed。通过其 sealed，我们就可以避免对该方法的进一步重写。

1.2.53 C# 中什么是密封类？

- sealed 修饰符用来阻止从一个类派生。如果一个密封类被指定为另一个类的基类时会发生编译时错误。

1.2.54 类和接口的区别是什么？

- 抽象类可以实现它的一些成员，但接口不能实现它的任何成员。
- 接口不能有字段，而抽象类可以有字段。
- 接口仅可以从另一个接口继承并且不能继承抽象类，而抽象类可以继承另一个抽象类或另一个接口。
- 类可以同时继承多个接口，而类不能同时继承多个类。
- 抽象类的成员可以定义访问修饰符而接口成员不能定义访问修饰符。

1.2.55 抽象方法和虚方法之间的区别是什么？

- 抽象方法不提供实现，并且强制派生类重写该方法 (除非派生类也是个抽象类)，而虚方法可以有实现并且在派生类中重写与否是可选的。因此虚方法可以实现并提供了派生类重写的选择。抽象方法不能提供实现并且强制派生类重写该方法。

1.2.56 **const** 和 **readonly** 之间有什么区别？

- const 声明的字段不能使用 static 修饰符，而 readonly 可以使用 static 修饰符。
- const 字段只能在声明时初始化，而 readonly 可以在声明时或在构造函数中初始化。
- const 字段的值在设计时就计算出来了，而 readonly 的值在运行时计算。

1.2.57 **C#.NET** 中使用的命名空间有哪些？

- 命名空间是类型的逻辑分组。
- using System;
- using System.Collections.Generic;
- using System.Windows.Forms;

1.2.58 **C#** 有哪些特点？

- C# 有以下特点：
  - 简单
  - 类型安全
  - 灵活
  - 面向对象
  - 兼容
  - 持久化
  - 互操作性
  - 有别于传统

1.2.59 继承有哪些不同的类别？

- 在面向对象编程中继承的四种类型：
  - 单继承：包括一个基类和一个派生类。
  - 多层继承 (Hierarchical inheritance) : 包括一个基类和继承自同一个基类的派生类。
  - 多级继承 (Multilevel inheritance): 包括从一个派生类派生出来的类。
  - 多重继承 (Multiple inheritance): 包括多个基类和一个派生类。

1.2.60 C# 中的修饰符有哪些？

- Abstract
- Sealed
- Virtual
- Const
- Event
- Extern
- Override
- Readonly
- Static
- New

1.2.61 C# 中访问修饰符的种类有哪些？

- C# 中的访问修饰符是：
- public
- protect
- private
- internal
- internal protect

1.2.62 Define destructors? 定义析构函数？

- 当类对象超出作用域或者被明确删除的时候，析构函数被调用。析构函数，顾名思义是用来销毁由构造函数创建的对象的。正如构造函数，析构函数是一个类成员方法，方法名和类名相同，只是由波浪号开头。

1.2.63 枚举数据类型怎么用？

- 枚举类型是另一种用户定义类型，它提供了一种连接名字为数字的方式，从而提高了代码的可理解性。enum 关键字自动地枚举一组词，赋予它们的值为 0,1,2 并以此类推。

1.2.64 定义构造函数？

- 构造函数是和它的类同名的成员方法。每当创建其关联的类的对象时构造函数都会被调用。它之所以被称为构造函数是因为它构造了类的数据成员的值。

1.2.65 什么是交错数组？

- 交错数组是元素为数组的数组。
- 交错数组元素的维度和大小可以不同。
- 交错数组有时称为“数组的数组”。

1.2.66 ref 和 out 参数之间的区别是什么？

- 传递给 ref 参数的参数必须先初始化。与此相比，对 out 参数来说，在参数传递给 out 参数之前不需要显示初始化。

1.2.67 C# 中 using 语句怎么用？

- using 语句通常是获取资源，执行语句，然后释放该资源。

1.2.68 什么是序列化？

- 序列化 (Serialization) 是将对象转换为字节流的过程。
- 反序列化 (De-serialization) 是从字节流创建对象这样相反的过程。
- 序列化/反序列化常用于传递对象。

1.2.69 结构和类之间有什么区别？

- 结构是值类型，类是引用类型。
- 结构不能有构造函数和析构函数。
- 类可以同时有构造函数和析构函数。
- 结构不支持继承，而类支持继承。

1.2.70 类和接口之间有什么区别？

- 类 (Class) 是对象的逻辑表示。它是数据集合和相关子过程的定义。
- 接口 (Interface) 也是一个类，包含没有任何方法体定义的方法。类不支持多重继承，但是接口支持。

1.2.71 什么是委托？

- 委托是类型安全，面向对象的函数指针的实现，并且在许多一个组件需要回调到使用它的组件这样的情况下使用。

1.2.72 什么是认证与授权？

- 认证是识别用户的过程。认证是以证书 (用户名和密码) 来识别/验证用户。
- 授权在认证之后执行。授权是一个基于这些用户身份授予访问权限的过程。
- 授权允许用户对特定资源的访问。

1.2.73 “this”可以在静态方法中用吗？

- 不，‘This’不能在静态方法中使用。仅仅只有静态变量/方法可以在静态方法中使用。

1.2.74 constants、readonly 和 static 之间的区别是什么？

- Constants: 值不能变。
- Read-only: 值在类的构造函数中仅仅初始化一次。
- Static: 值可以被初始化一次。

1.2.75 C# 中支持哪些语句类型？

- C# 支持的几种不同的语句类型是
  - 块语句
  - 声明语句
  - 表达式语句
  - 选择语句
  - 迭代语句
  - 跳转语句
  - 异常处理语句
  - 检查和未检查
  - Lock 语句

1.2.76 C# 是否可以对内存直接进行操作?

- 这个问题比较难回答，也是个很大的问题。但是可以这样问答。C# 是可以对内存进行直接操作的，虽然很少用到指针，但是 C# 是可以使用指针的，在用的时候需要在前边加 unsafe,，在.net 中使用了垃圾回收机制 (GC) 功能，它替代了程序员，不过在 C# 中不可以直接使用 finalize 方法，而是在析构函数中调用基类的 finalize() 方法。

1.2.77 Collection 和 Collections 的区别?

- Collection 是集合类的上级接口，Collections 是针对集合类的一个帮助类，它提供一系列静态方法来实现对各种集合的搜索，排序，线程安全化操作。

1.2.78 New 有种用法?

- 有 3 种，第一种是，实例化如：New Class()
- 第二种是，public new 隐藏基类的方法
- 第三种是，在泛型类申明中的任何类型参数都必须有公共的无参构造函数。
- 17，任何把一个 Array 复制到 Arraylist 中?
- Foreach (object o in Array), ArrayList.Add (0)
- 等有好多中种方法。自己想。

1.2.79 概述反射和序列化?

- 反射：要给发射下一个定义还是比较难的，这里先说说我的理解。反射提供了封装程序集，模块和类型对象，可以用反射动态地创建类型的实例，将类型绑定到现有对象，或者从现有对象类型里获取类型，然后调用类型的方法或访问字段和属性。
- 序列化：将对象转换为另一种媒介传输的格式过程。如，序列化一个对象，用 Http 通过 internet 在客户端和服务器之间传递该对象，在另一端用反序列化从该流中重新得到对象。

1.2.80 UDP 和 TCP 连接有和异同?

- TCP 是传输控制协议，提供的是面向连接的，是可靠的，字节流服务，当用户和服务器彼此进行数据交互的时候，必须在他们数据交互前要进行 TCP 连接之后才能传输数据。TCP 提供超时重拨，检验数据功能。UDP 是用户数据报协议，是一个简单的面向数据报的传输协议，是不可靠的连接。

1.2.81 进程和线程分别该怎么理解?

- 进程是比线程大的程序运行单元，都是由操作系统所体会的系统运行单元，一个程序中至少要有一个进程，有一个进程中，至少要有一个线程，线程的划分尺度要比进程要小，进程拥有独立的内存单元，线程是共享内存，从而极大的提高了程序的运行效率同一个进程中的多个线程可以并发执行。

1.2.82 ASP.NET 页面之间传递值的几种方式?

- QueryString,session,cookies,application,server.Transfer,respose.redictor.

1.2.83 什么叫应用程序域? 什么是托管代码? 什么是强类型系统? 什么是装箱和拆箱? 什么是重载?CTS、CLS 和 CLR 分别作何解释?

- 应用程序域：就是为安全性，可靠性，隔离性，和版本控制，及卸载程序提供的隔离边界。它通常由运行库宿主创建，应用程序域提供了一个更安全，用途更广的处理单元。
- 托管代码：使用 CLR 编译语言编辑器开发编写的代码就叫托管代码。
- 装箱和拆箱：是把值类型转换为引用类型的过程，是隐式的，相反的过程就是拆箱，是显式的。
- CTS 是公共类型系统，CLS 是公共语言规范，CLR 公共语言运行库。
- 强类型系统：每个变量和对象都必须具有申明类型。

1.2.84 ASP.net 的身份验证方式有哪些?

- windows,forms,passport

1.2.85 解释一下 UDDI、WSDL 的意义及其作用?

- UDDI 是统一描述集成协议，是一套基于 Web 的，分布式的，为 WEB 服务提供的信息注册的实现标准规范，同时为也是为企业本身提供的 Web 服务注册以让别的企业能够发现并访问的协议标准。提供了基于标准的规范，用于描述和发现服务，还提供了一组基于因特网的实现。
- WSDL 这是一个基于 XML 的描述 WEB 服务的接口。

1.2.86 什么是 SOAP?

- 是简单访问协议。是在分布式环境中，交换信息并实现远程调用的协议。是一个基于 XML 的协议。使用 SOAP，可以不考虑任何传输协议，但通常还是 HTTP 协议，可以允许任何类型的对象或代码，在任何平台上，以任一种语言相互通信。它是一种轻量级协议。

1.2.87 如何部署一个 ASP.net 页面?

- vs2003,vs2005 里边都有发表机制，vs2003 可以发布然后在复制部署。
- Vs2005 可以直接部署到对应的位置。

1.2.88 如何理解.net 中的垃圾回收机制?

- .NET 中的垃圾回收机制是引用程序对内存的回收和释放。当每次用 new 关键字创建一个对象时，运行库都要从托管堆中为其分配内存，因为空间是有限的，最终垃圾回收机制是要回收不用的内存的。已释放内存，重新使用。

1.2.89 C# 是一门托管语言，那么是不是说明只要用 C#，就能保证不会出现内存泄露和其他资源泄漏？如果不是，在哪些情况下可能会出现泄漏？

- C# 不能保证没有资源泄漏。比如如下几种情况可能会造成资源泄漏：
  - （1）调用 Native code，比如用 P/Invoke 或者调用 COM；
  - （2）读写文件时的，没有及时 close stream, 或者 ADO.NET 连数据库时，没有及时关闭连接，也算资源泄漏？
  - （3）注册事件后没有 remove，导致 publisher 和 subscriber 的强依赖，垃圾回收可能会被推迟；
  - （4）.NET 还定义了一些方法直接申请非托管内存，比如 Marshal.AllocHGlobal 和 Marshal.AllocCoTaskMem。通过这种方式得到的内存，如果没有及时释放，也会造成内存泄露。

1.2.90 下面的两段 C# 有哪些不同？

```
static void CatchException1() {
    try {
        Function();
    } catch {
        throw;
    }
}

static void CatchException2() {
    try {
        Function();
    } catch (Exception e) {
        throw e;
    }
}
```

- 两个函数的 catch 都是重新抛出截获的 exception，但抛出的 exception 的 call stack 是不一样的。对于第一种方法，exception 的 call stack 是从最开始的抛出地点开始的。对于第二种方法，exception 的 call stack 是从 CatchException2 开始的，最初抛出的地方相关的信息被隐藏了。

1.2.91 运行下图中的 C++ 代码，打印出的结果是什么？

```
bool Fun1(char* str) {
    printf("%s\n", str);
    return false;
}
bool Fun2(char* str) {
    printf("%s\n", str);
    return true;
}
int _tmain(int argc, _TCHAR* argv[]) {
    bool res1, res2;
    res1 = (Fun1("a") && Fun2("b")) || (Fun1("c") || Fun2("d"));
    res2 = (Fun1("a") && Fun2("b")) && (Fun1("c") || Fun2("d"));
    return res1 || res2;
}
```

- 打印出 4 行，分别是 a、c、d、a。
- 在 C/C++ 中，与、或运算是从左到右的顺序执行的。在计算 rest1 时，先计算 Fun1(“a”) && Func2(“b”)。首先 Func1(“a”) 打印出内容为 a 的一行。由于 Fun1(“a”) 返回的是 false, 无论 Func2(“b”) 的返回值是 true 还是 false，Fun1(“a”) && Func2(“b”) 的结果都是 false。由于 Func2(“b”) 的结果无关重要，因此 Func2(“b”) 会略去而不做计算。接下来计算 Fun1(“c”) || Func2(“d”)，分别打印出内容 c 和 d 的两行。
- 在计算 rest2 时，首先 Func1(“a”) 打印出内容为 a 的一行。由于 Func1(“a”) 返回 false，和前面一样的道理，Func2(“b”) 会略去不做计算。由于 Fun1(“a”) && Func2(“b”) 的结果是 false，不管 Fun1(“c”) && Func2(“d”) 的结果是什么，整个表达式得到的结果都是 false，因此 Fun1(“c”) && Func2(“d”) 都将被忽略。

1.2.92 运行下面的 C# 代码，打印出来的结果是什么？

```
struct Person {
    public string Name;
    public override string ToString() {
        return Name;
    }
}
class Program {
    static void Main(string[] args) {
        ArrayList array = new ArrayList();
        Person jim = new Person() {Name = "Jim"};
        array.Add(jim);
        Person first = (Person)array[0];
        first.Name = "Peter";
        Console.WriteLine(array[0].ToString());
    }
}
```

- Person 的定义是一个 struct,因此是一个值类型。在运行到语句 Person first = (Person)array[1] 的时候，first 是 array[1] 的一个拷贝，first 和 array[1] 不是一个实例。因此修改 first 对 array[1] 没有影响。

1.2.93 运行下面的 C++ 代码，打印的结果是什么？

```
class Base {
    public:
    void print() { doPrint();}
    private:
    virtual void doPrint() {cout << "Base::doPrint" << endl;}
};

class Derived : public Base {
    private:
    virtual void doPrint() {cout << "Derived::doPrint" << endl;}
};
int _tmain(int argc, _TCHAR* argv[]) {
    Base b;
    b.print();
    Derived d;
    d.print();
    return 0;
}
```

- 输出两行，分别是 Base::doPrint 和 Derived::doPrint。在 print 中调用 doPrint 时，doPrint() 的写法和 this->doPrint() 是等价的，因此将根据实际的类型调用对应的 doPrint。所以结果是分别调用的是 Base::doPrint 和 Derived::doPrint2。如果感兴趣，可以查看一下汇编代码，就能看出来调用 doPrint 是从虚函数表中得到函数地址的。

[1] DEFINITION NOT FOUND.

1.2.94 维护数据库的完整性、一致性、你喜欢用触发器还是自写业务逻辑？为什么**?**

- 尽可能用约束（包括 CHECK、主键、唯一键、外键、非空字段）实现，这种方式的效率最好；其次用触发器，这种方式可以保证无论何种业务系统访问数据库都能维持数据库的完整性、一致性；最后再考虑用自写业务逻辑实现，但这种方式效率最低、编程最复杂，当为下下之策。

1.2.95 什么是事务？什么是锁？

- 事务是指一个工作单元，它包含了一组数据操作命令，并且所有的命令作为一个整体一起向系统提交或撤消请求操作，即这组命令要么都执行，要么都不执行。
- 锁是在多用户环境中对数据的访问的限制。SqlServer 自动锁定特定记录、字段或文件，防止用户访问，以维护数据安全或防止并发数据操作问题，锁可以保证事务的完整性和并发性。

1.2.96 什么是索引，有什么优点？

- 索引象书的目录类似，索引使数据库程序无需扫描整个表，就可以在其中找到所需要的数据，索引包含了一个表中包含值的列表，其中包含了各个值的行所存储的位置，索引可以是单个或一组列，索引提供的表中数据的逻辑位置，合理划分索引能够大大提高数据库性能。

1.2.97 视图是什么？游标是什么？

- 视图是一种虚拟表，虚拟表具有和物理表相同的功能，可以对虚拟表进行增该查操作;
- 视图通常是一个或多个表的行或列的子集;
- 视图的结果更容易理解（修改视图对基表不影响），获取数据更容易（相比多表查询更方便），限制数据检索（比如需要隐藏某些行或列），维护更方便。
- 游标对查询出来的结果集作为一个单元来有效的处理，游标可以定位在结果集的特定行、从结果集的当前位置检索一行或多行、可以对结果集中当前位置进行修改、

1.2.98 什么是存储过程？有什么优点？

- 存储过程是一组予编译的 SQL 语句
- 它的优点：
  - 1. 允许模块化程序设计，就是说只需要创建一次过程，以后在程序中就可以调用该过程任意次。
  - 2. 允许更快执行，如果某操作需要执行大量 SQL 语句或重复执行，存储过程比 SQL 语句执行的要快。
  - 3. 减少网络流量，例如一个需要数百行的 SQL 代码的操作有一条执行语句完成，不需要在网络中发送数百行代码。
  - 4. 更好的安全机制，对于没有权限执行存储过程的用户，也可授权他们执行存储过程。

1.2.99 什么是触发器？

- 触发器是一种特殊类型的存储过程，出发器主要通过事件触发而被执行的，
- 触发器的优点：
  - 1. 强化约束，触发器能够提供比 CHECK 约束;
  - 2. 跟踪变化，触发器可以跟踪数据库内的操作，从而不允许未经允许许可的更新和变化;

- 3. 联级运算，比如某个表上的触发器中包含对另一个表的数据操作，而该操作又导致该表上的触发器被触发

1.2.100 简单介绍下http://ADO.NETADO?

- ADO 以 Recordset 存储, 而http://ADO.NETDataSethttp://ADO.NET。

1.2.101 http://ASP.NETASP?

- http://ASP.NETASP。

1.2.102 C# 中的委托是什么？事件是不是一种委托？

- 委托本质上是一种“方法接口”，它相当于 C/C++ 中的函数指针，当然它比函数指针安全，在 C# 中通常用于事件处理。事件不是委托，不过由于事件的性质决定了处理它的程序逻辑能访问的参数，因此，在 C# 中处理事件的逻辑都包装为委托。

1.2.103 列举http://ADO.NET

- Connection 连接对象,Command 执行命令和存储过程,DataReader 向前只读的数据流,DataAdapter 适配器，支持增删查询，DataSet 数据级对象，相当与内存里的一张或多张表。

1.2.104 概述三层结构体系

- 表示层 (UI), 业务逻辑层 (BLL), 数据访问层 (DAL)

1.2.105 什么是装箱和拆箱？什么是重载？

- 装箱就是把值类型转成引用类型，拆箱相反把引用转换成值类型。
- 重载就是指一个方法名相同, 参数个数不相同, 返回值可以相同的方法。

1.2.106 简述 WebService

- WebService 服务可以描述为可以在 web 上部署并可以被任何应用程序或其他服务调用的功能。所谓服务就是系统提供一组接口，并通过接口使用系统提供的功能，WebService 服务可以提供任何企业到客户，企业到企业，点对点或部门对部门通讯所需的服务，比如一个公司可以通过网络连接到另一个公司的服务，从而直接传递订购单。

1.2.107 DataGrid 的 Datasouse 可以连接什么数据源

- DataTabe，DataView，DataSet，DataViewManager，任何实现 IListSource 接口的组件，任何实现 IList 接口的组件

1.2.108 概述反射和序列化

- 反射：公共语言运行库加载器管理应用程序域。这种管理包括将每个程序集加载到相应的应用程序域以及控制每个程序集中类型层次结构的内存布局。程序集包含模块，而模块包含类型，类型又包含成员。反射则提供了封装程序集、模块和类型的对象。您可以使用反射动态地创建类型的实例，将类型绑定到现有对象，或从现有对象中获取类型。然后，可以调用类型的方法或访问其字段和属性。
- 序列化：序列化是将对象状态转换为可保持或传输的格式的过程。与序列化相对的是反序列化，它将流转换为对象。这两个过程结合起来，可以轻松地存储和传输数据。

1.2.109 概述 O/R Mapping 的原理

- 利用反射，配置将对象和数据库表映射。

1.2.110 列举http://ADO.NET

- 共享类：DataSet，DataTable，DataRow，DataColumn，DataRealtion，Constraint，DataColumnMapping，DataTableMapping
- 特定类:(x)Connection,(x)Command,(x)CommandBuilder,(x)DataAdapter,(x)DataReader,(x)Parameter，(x)Transaction

1.2.111 详述.NET 里 class 和 struct 的异同

- 结构与类共享几乎所有相同的语法，但结构比类受到的限制更多：尽管结构的静态字段可以初始化，结构实例字段声明还是不能使用初始值设定项。
- 结构不能声明默认构造函数（没有参数的构造函数）或析构函数。
- 结构的副本由编译器自动创建和销毁，因此不需要使用默认构造函数和析构函数。实际上，编译器通过为所有字段赋予默认值（参见默认值表）来实现默认构造函数。
- 结构不能从类或其他结构继承。
- 结构是值类型 – 如果从结构创建一个对象并将该对象赋给某个变量，变量则包含结构的全部值。复制包含结构的变量时，将复制所有数据，对新副本所做的任何修改都不会改变旧副本的数据。
- 由于结构不使用引用，因此结构没有标识 – 具有相同数据的两个值类型实例是无法区分的。C# 中的所有值类型本质上都继承自 ValueType，后者继承自 Object。编译器可以在一个称为装箱的过程中将值类型转换为引用类型。
- 结构具有以下特点：
  - 结构是值类型，而类是引用类型。
  - 向方法传递结构时，结构是通过传值方式传递的，而不是作为引用传递的。
  - 与类不同，结构的实例化可以不使用 new 运算符。
  - 结构可以声明构造函数，但它们必须带参数。
  - 一个结构不能从另一个结构或类继承，而且不能作为一个类的基。所有结构都直接继承自 System.ValueType，后者继承自 System.Object。
  - 结构可以实现接口。
  - 在结构中初始化实例字段是错误的。

1.2.112 什么叫应用程序域？什么是托管代码？什么是强类型系统？什么是装箱和拆箱？什么是重载？CTS、CLS 和 CLR 分别作何解释？

- 应用程序域：应用程序域为安全性、可靠性、版本控制以及卸载程序集提供了隔离边界。应用程序域通常由运行库宿主创建，运行库宿主负责在运行应用程序之前引导公共语言运行库。应用程序域提供了一个更安全、用途更广的处理单元，公共语言运行库可使用该单元提供应用程序之间的隔离。
- 托管代码：使用基于公共语言运行库的语言编译器开发的代码称为托管代码；托管代码具有许多优点，例如：跨语言集成、跨语言异常处理、增强的安全性、版本控制和部署支持、简化的组件交互模型、调试和分析服务等。

- 强类型：C# 是强类型语言；因此每个变量和对象都必须具有声明类型。
- 装箱和拆箱：装箱和拆箱使值类型能够被视为对象。对值类型装箱将把该值类型打包到 Object 引用类型的一个实例中。这使得值类型可以存储于垃圾回收堆中。拆箱将从对象中提取值类型。
- 重载：每个类型成员都有一个唯一的签名。方法签名由方法名称和一个参数列表（方法的参数的顺序和类型）组成。只要签名不同，就可以在一种类型内定义具有相同名称的多种方法。当定义两种或多种具有相同名称的方法时，就称作重载。
- CTS 通用类型系统 (common type system) ：一种确定公共语言运行库如何定义、使用和管理类型的规范。
- CLR 公共语言运行库：.NET Framework 提供了一个称为公共语言运行库的运行时环境，它运行代码并提供使开发过程更轻松的服务。
- CLS 公共语言规范：要和其他对象完全交互，而不管这些对象是以何种语言实现的，对象必须只向调用方公开那些它们必须与之互用的所有语言的通用功能。为此定义了公共语言规范 (CLS)，它是许多应用程序所需的一套基本语言功能。

1.2.113 如何理解委托

- 委托类似于 C++ 函数指针，但它是类型安全的。委托允许将方法作为参数进行传递。委托可用于定义回调方法。委托可以链接在一起；例如，可以对一个事件调用多个方法。方法不需要与委托签名精确匹配。有关更多信息，请参见协变和逆变。C# 2.0 版引入了匿名方法的概念，此类方法允许将代码块作为参数传递，以代替单独定义的方法。

1.2.114 http://26.ASP.net?

- Windows 身份验证提供程序：提供有关如何将 Windows 身份验证与 Microsoft Internet 信息服务 (IIS) 身份验证结合使用来确保 The Official Microsoft ASP.NET Site 应用程序安全的信息。
- Forms 身份验证提供程序：提供有关如何使用您自己的代码创建应用程序特定的登录窗体并执行身份验证的信息。使用 Forms 身份验证的一种简便方法是使用 The Official Microsoft ASP.NET Site 成员资格和 The Official Microsoft ASP.NET Site 登录控件，它们一起提供了一种只需少量或无需代码就可以收集、验证和管理用户凭据的方法。
- Passport 身份验证提供程序：提供有关由 Microsoft 提供的集中身份验证服务的信息，该服务为成员站点提供单一登录和核心配置

1.2.115 活动目录的作用

- Active Directory 存储了有关网络对象的信息，并且让管理员和用户能够轻松地查找和使用这些信息。
- Active Directory 使用了一种结构化的数据存储方式，并以此作为基础对目录信息进行合乎逻辑的分层组织。

1.2.116 解释一下 UDDI、WSDL 的意义及其作用

- UDDI：统一描述、发现和集成协议 (UDDI, Universa Description, Discovery and Integration) 是一套基于 Web 的、分布式的、为 Web 服务提供的信息注册中心的实现标准规范，同时也包含一组使企业能将自身提供的 Web 服务注册以使得别的企业能够发现的访问协议的实现标准。UDDI 提供了一组基于标准的规范用于描述和发现服务，还提供了一组基于因特网的实现。

- WSDL：WSDL 描述 Web 服务的公共接口。这是一个基于 XML 的关于如何与 Web 服务通讯和使用的服务描述；
- 作用：服务 URL 和命名空间：网络服务的类型（可能还包括 SOAP 的函数调用，正像我所说过的，WSDL 足够自如地去描述网络服务的广泛内容）。有效函数列表，每个函数的参数，每个参数的类型，每个函数的返回值及其数据类型。

1.2.117 什么是 SOAP?

- SOAP（Simpe Object Access Protoco ）简单对象访问协议是在分散或分布式的环境中交换信息并执行远程过程调用的协议，是一个基于 XML 的协议。使用 SOAP，不用考虑任何特定的传输协议（最常用的还是 HTTP 协议），可以允许任何类型的对象或代码，在任何平台上，以任何一直语言相互通信。
- SOAP 是一种轻量级协议，用于在分散型、分布式环境中交换结构化信息。SOAP 利用 XML 技术定义一种可扩展的消息处理框架，它提供了一种可通过多种底层协议进行交换的消息结构。这种框架的设计思想是要独立于任何一种特定的编程模型和其他特定实现的语义。
- SOAP 定义了一种方法以便将 XML 消息从 A 点传送到 B 点。为此，它提供了一种基于 XML 且具有以下特性的消息处理框架：1) 可扩展，2) 可通过多种底层网络协议使用，3) 独立于编程模型。

1.2.118 如何部署一个http://ASP.net?

- VS 2005 和 VS 2003 都有发布机制。2003 可以发布然后再复制部署。VS2005 基本上可以直接部署到对应位置。

1.2.119 如何理解.net 中的垃圾回收机制?

- .NETFramework 的垃圾回收器管理应用程序的内存分配和释放。每次您使用 new 运算符创建对象时，运行库都从托管堆为该对象分配内存。只要托管堆中有地址空间可用，运行库就会继续为新对象分配空间。但是，内存不是无限大的。最终，垃圾回收器必须执行回收以释放一些内存。垃圾回收器优化引擎根据正在进行的分配情况确定执行回收的最佳时间。当垃圾回收器执行回收时，它检查托管堆中不再被应用程序使用的对象并执行必要的操作来回收它们占用的内存。

1.2.120 概述.NET 中的 GC 机制。

- GC 的全称是 garbage collection，中文名称垃圾回收，是.NET 中对内存管理的一种功能。垃圾回收器跟踪并回收托管内存中分配的对象，定期执行垃圾回收以回收分配给没有有效引用的对象的内存。当使用可用内存不能满足内存请求时，GC 会自动进行。
- 在进行垃圾回收时，垃圾回收器会首先搜索内存中的托管对象，然后从托管代码中搜索被引用的对象并标记为有效，接着释放没有被标记为有效的对象并收回内存，最后整理内存将有效对象挪动到一起
- http://33.ASP.NET?
- Response、Request、Server、Session、Application、Cookie

1.2.121 死锁的必要条件？怎么克服？

- 系统的资源不足，进程的推进的顺序不合适，资源分配不当，一个资源每次只能被一个进程使用，一个资源请求资源时，而此时这个资源已阻塞，对已获得资源不放，进程获得资源时，未使用完前，不能强行剥夺。

1.2.122 简要谈您对微软.NET 构架下 remoting 和 webservice 两项技术的理解以及实际中的应用。

- WS 主要是可利用 HTTP，穿透防火墙。而 Remoting 可以利用 TCP/IP，二进制传送提高效率。
- remoting 是.net 中用来跨越 machine,process, appdomain 进行方法调用的技术, 对于三成结构的程序，就可以使用 remoting 技术来构建. 它是分布应用的基础技术. 相当于以前的 DCOM。
- Web Service 是一种构建应用程序的普通模型，并能在所有支持 internet 网通讯的操作系统上实施。Web Service 令基于组件的开发和 web 的结合达到最佳，基于组件的对象模型。

1.2.123 公司要求开发一个继承 System.Windows.Forms.ListView 类的组件，要求达到以下的特殊功能：点击 ListView 各列列头时，能按照点击列的每行值进行重排视图中的所有行 (排序的方式如 DataGrid 相似)。根据您的知识，请简要谈一下您的思路

- 根据点击的列头, 包该列的 ID 取出, 按照该 ID 排序后, 在给绑定到 ListView 中。

1.2.124 给定以下 XML 文件，完成算法流程图。

```
<FileSystem>
   < DriverC >
       <Dir DirName=" MSDOS622" >
              <File FileName =" Redirecting to Command Brand" ></File>
          </Dir>
          <File FileName =" MSDOS.SYS" ></File>
          <File FileName =" IO.SYS" ></File>
       </DriverC>
   </FileSystem>
```

- 请画出遍历所有文件名（FileName）的流程图 (请使用递归算法)。

```
void FindFile(Directory d) {
    FileOrFolders = d.GetFileOrFolders();
    foreach ( FileOrFolder fof in FileOrFolders) {
        if (fof  is  File) {
            // You Found a file;
        } else if (fof is Directory)
            FindFile(fof);
        }
    }
}
```

1.2.125 http://109.ADO.NETADO?

- 1:http://ado.netole db 提供程序, 而是使用.net 托管提供的程序,
- 2: 不使用 com
- 3: 不在支持动态游标和服务器端游
- 4:, 可以断开 connection 而保留当前数据集可用
- 5: 强类型转换
- 6:xml 支持

1.2.126 你觉得 **ASP.NET 2.0**（**VS2005**）和你以前使用的开发工具（**.Net 1.0** 或其他）有什么最大的区别？你在以前的平台上使用的哪些开发思想（**pattern/ architecture**）可以移植到 **ASP.NET2.0** 上 **(**或者已经内嵌在 **ASP.NET 2.0** 中**)**

- 1、ASP.NET2.0 把一些代码进行了封装打包, 所以相比 1.0 相同功能减少了很多代码。
- 2、同时支持代码分离和页面嵌入服务器端代码两种模式, 以前 1.0 版本,.NET 提示帮助只有在分离的代码文件, 无法在页面嵌入服务器端代码获得帮助提示。
- 3、代码和设计界面切换的时候,2.0 支持光标定位. 这个我比较喜欢。
- 4、在绑定数据, 做表的分页.UPDATE,DELETE, 等操作都可以可视化操作, 方便了初学者。
- 5、在http://ASP.NET40,。

1.2.127 重载与覆盖的区别？

- 1、方法的覆盖是子类和父类之间的关系，是垂直关系；方法的重载是同一个类中方法之间的关系，是水平关系。
- 2、覆盖只能由一个方法，或只能由一对方法产生关系；方法的重载是多个方法之间的关系。
- 3、覆盖要求参数列表相同；重载要求参数列表不同。
- 4、覆盖关系中，调用那个方法体，是根据对象的类型（对像对应存储空间类型）来决定；重载关系，是根据调用时的实参表与形参表来选择方法体的。

1.2.128 什么是 **WSE**？目前最新的版本是多少？

- WSE (WebService Extension) 包来提供最新的 WEB 服务安全保证，最新版本 2.0。

1.2.129 **a=10,b=15**，在不用第三方变量的前提下，把 **a,b** 的值互换

- a=a+b;b=a-b;a=a-b;

1.2.130 还有变态要求，需要代码最短呢。有两个结果：

- 1) a^=b$^{(b=a^b)}$; // 13 个字节
- 2) a=b+(b=a)*0; // 11 个字节

1.2.131 请简述面向对象的多态的特性及意义！

- 面向对象的编程使用了派生继承以及虚函数机制. 一个本来指向基类的对象指针可以指向其派生类的. 并访问从基类继承而来的成员变量和函数. 而虚函数是专门为这个特性设计的, 这个函数在每个基类的派生类中都是同一个名字, 但函数体却并不一定相同, 派生类往往为实现自己的功能而修改这个虚函数. 这样用一个指针就能够实现对多种不同的派生类的访问, 并实现其派生类的特定功能 (代码)

1.2.132 **session** 喜欢丢值且占内存，**Cookis** 不安全，请问用什么办法代替这两种原始的方法

- 用 ViewState，stateserver

1.2.133 对数据的并发采用什么办法进行处理较好。

- 可以控制连接池的连接数量, 条件好的话可以用负载平衡
- http://123.ADO.NET?
- 开放式并发，没有用到数据库的锁，而依靠 SQL 语句判断数据是否已经变化了。
- 步骤：通常使用 VS 提供的工具生成 SQL 语句，工具生成的结果实在不符合要求，才手工写 SQL 语句。

1.2.134 动态创建的控件 **PostBack** 后是否可以保存下来, 为什么**?**

- http://ASP.NETViewState

1.2.135 要点：**1.** 联动效果，运行代码只要执行 **Cat.Cryed()** 方法。**2.** 对老鼠和主人进行抽象

- 评分标准:
  - <1>. 构造出 Cat、Mouse、Master 三个类，并能使程序运行 (2 分)
  - <2> 从 Mouse 和 Master 中提取抽象（5 分）
  - <3> 联动效应，只要执行 Cat.Cryed() 就可以使老鼠逃跑，主人惊醒。(3 分)

```
public interface Observer {
    void Response();           // 观察者的响应，如是老鼠见到猫的反映
}
public interface Subject {
    void AimAt(Observer obs);  // 针对哪些观察者，这里指猫的要扑捉的对象---老鼠
}
public class Mouse : Observer {
    private string name;
    public Mouse(string name, Subject subj) {
        this.name = name;
        subj.AimAt(this);
    }
    public void Response() {
        Console.WriteLine(name + " attempt to escape!");
    }
}
public class Master : Observer {
    public Master(Subject subj) {
        subj.AimAt(this);
    }
    public void Response() {
        Console.WriteLine("Host waken!");
    }
}
public class Cat : Subject {
    private ArrayList observers;
    public Cat() {
        this.observers = new ArrayList();
    }
    public void AimAt(Observer obs) {
        this.observers.Add(obs);
    }
    public void Cry() {
        Console.WriteLine("Cat cryed!");
        foreach (Observer obs in this.observers) {
            obs.Response();
        }
    }
}
class MainClass {
    static void Main(string[] args) {
        Cat cat = new Cat();
        Mouse mouse1 = new Mouse("mouse1", cat);
```

```
        Mouse mouse2 = new Mouse("mouse2", cat);
        Master master = new Master(cat);
        cat.Cry();
    }
}

//-------------------------------------------------------------------------------------------------
// 设计方法二: 使用 event -- delegate 设计..
//-------------------------------------------------------------------------------------------------
public delegate void SubEventHandler();
public abstract class Subject {
    public event SubEventHandler SubEvent;
    protected void FireAway() {
        if (this.SubEvent != null)
            this.SubEvent();
    }
}
public class Cat : Subject {
    public void Cry() {
        Console.WriteLine("cat cryed.");
        this.FireAway();
    }
}
public abstract class Observer {
    public Observer(Subject sub) {
        sub.SubEvent += new SubEventHandler(Response);
    }
    public abstract void Response();
}
public class Mouse : Observer {
    private string name;
    public Mouse(string name, Subject sub) : base(sub) {
        this.name = name;
    }
    public override void Response() {
        Console.WriteLine(name + " attempt to escape!");
    }
}
public class Master : Observer {
    public Master(Subject sub) : base(sub){}
    public override void Response() {
        Console.WriteLine("host waken");
    }
}
class Class1 {
    static void Main(string[] args) {
        Cat cat = new Cat();
        Mouse mouse1 = new Mouse("mouse1", cat);
        Mouse mouse2 = new Mouse("mouse2", cat);
        Master master = new Master(cat);
        cat.Cry();
    }
}
```

1.3 C# 中的 Static, readonly 和 const 之间的比较

- http://www.cnblogs.com/suizhikuo/p/4739388.html

1.4 C# 中的委托和事件 - Part.1 （讲叙得非常透彻 ~!）

- http://www.tracefact.net/tech/009.html

1.5 C# 事件和 Unity3D （讲叙得非常透彻 ~!）

- http://zijan.iteye.com/blog/871207