# C#.NET 笔试题 基础篇

C#.NET 笔试题 基础篇

1.C#中堆和栈的区别？
2.C#中的委托是什么？事件是不是一种委托？
3.C#静态构造函数特点是什么？
4.CTS、CLS、CLR 分别作何解释
5.C#中什么是值类型与引用类型？
6.请详述在 C#中类(class)与结构(struct)的异同？
7.new 关键字的作用
8.int?和 int 有什么区别
9.C#中值传递与引用传递的区别是什么？
10.C#中参数传递 ref 与 out 的区别？
11.C#中什么是装箱和拆箱？
12.C#实现多态的过程中 overload 重载 与 override 重写的区别？
13.C# 中 static 关键字的作用？
14.C# 成员变量和成员函数前加 static 的作用？
15.C#中索引器的实现过程，是否只能根据数字进行索引，请描述一下
16.C#中 abstract class 和 interface 有什么区别?
17.C#中用 sealed 修饰的类有什么特点？
18.字符串中 string str=null 和 string str=""和 string str=string.Empty 的区别
19.byte b = 'a'; byte c = 1; byte d = 'ab'; byte e = '啊'; byte g = 256; 这些变量有些错误是错在哪里?
20.string 和 StringBuilder 的区别,两者性能的比较
21.什么是扩展方法？
22. 特性是什么？如何使用？
23.什么叫应用程序域(AppDomain)
24.byte a =255;a+=5;a 的值是多少？
25.const 和 readonly 有什么区别？
26.分析下面代码，a、b 的值是多少？
27.Strings = new String(“xyz”);创建了几个 String Object?
28.c#可否对内存直接操作
29.什么是强类型，什么是弱类型？哪种更好些？为什么?
30. Math.Round(11.5)等于? Math.Round(-11.5)等于?Math.Round(-11.3)呢？
31.&和&&的区别
32.i++和++i 有什么区别？
33.as 和 is 的区别
34.谈谈 final、finally 的区别。
35.简述 C#成员修饰符
36.什么是匿名类，有什么好处？
37.说说什么是逐字字符串

## 1.C#中堆和栈的区别?

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。

存放在栈中时要管存储顺序，保持着先进后出的原则，他是一片连续的内存域，有系统自动分配和维护；

堆：是无序的，他是一片不连续的内存域，有用户自己来控制和释放，如果用户自己不释放的话，当内存达到一定的特定值时，通过垃圾回收器(GC)来回收。

栈内存无需我们管理，也不受 GC 管理。当栈顶元素使用完毕，立马释放。而堆则需要 GC 清理。

使用引用类型的时候，一般是对指针进行的操作而非引用类型对象本身。但是值类型则操作其本身

---->详解

## 2.C#中的委托是什么？事件是不是一种委托？

委托的本质是一个类，委托是将一种方法作为参数代入到另一种方法。

事件是委托的实例，事件是一种特殊的委托。//比如：onclick 事件中的参数就是一种方法。

---->详解

## 3.C#静态构造函数特点是什么？

最先被执行的构造函数，且在一个类里只允许有一个无参的静态构造函数执行顺序：静态变量>静态构造函数>实例变量>实例构造函数

## 4.CTS、CLS、CLR 分别作何解释

CTS：通用语言系统。CLS：通用语言规范。CLR：公共语言运行库。

CTS：Common Type System 通用类型系统。Int32、Int16→int、String→string、Boolean→bool。

每种语言都定义了自己的类型，.NET 通过 CTS 提供了公共的类型，然后翻译生成对应的.NET 类型。

CLS：Common Language Specification 通用语言规范。不同语言语法的不同。每种语言都有自己的语法，.NET 通过 CLS 提供了公共的语法，然后不同语言翻译生成对应的.NET 语法。

CLR：Common Language Runtime 公共语言运行时，就是 GC、JIT 等这些。有不同的 CLR，比如服务器 CLR、Linux CLR（Mono）、Silverlight CLR(CoreCLR)。相当于一个发动机，负责执行 IL。

---->详解

### 5.C#中什么是值类型与引用类型?

值类型：struct、enum、int、float、char、bool、decimal

引用类型：class、delegate、interface、array、object、string

---->详解

### 6.请详述在 C#中类(class)与结构(struct)的异同?

class 可以被实例化,属于引用类型,

class 可以实现接口和单继承其他类,还可以作为基类型,是分配在内存的堆上的 struct 属于值类型,不能作为基类型,但是可以实现接口,是分配在内存的栈上的.

---->详解

### 7.new 关键字的作用

运算符：创建对象实例

修饰符：在派生类定义一个重名的方法，隐藏掉基类方法

约束：泛型约束定义，约束可使用的泛型类型

---->详解

### 8.int?和 int 有什么区别

int? 为可空类型，默认值可以是 null

int 默认值是 0

int?是通过 int 装箱为引用类型实现

---->详解

### 9.C#中值传递与引用传递的区别是什么?

值传递时，系统首先为被调用方法的形参分配内存空间，并将实参的值按位置一一对应地复制给形参，此后，被调用方法中形参值得任何改变都不会影响到相应的实参；引用传递时，系统不是将实参本身的值复制后传递给形参，而是将其引用值（即地址值）传递给形参，因此，形参所引用的该地址上的变量与传递的实参相同，方法体内相应形参值得任何改变都将影响到作为引用传递的实参。

简而言之，按值传递不是值参数是值类型，而是指形参变量会复制实参变量，也就是会在栈上多创建一个相同的变量。而按引用传递则不会。

可以通过 ref 和 out 来决定参数是否按照引用传递。

---->详解

### 10.C#中参数传递 ref 与 out 的区别?

（1）ref 指定的参数在函数调用时必须先初始化，而 out 不用

（2）out 指定的参数在进入函数时会清空自己，因此必须在函数内部进行初始化赋值操作，而 ref 不用

总结：ref 可以把值传到方法里，也可以把值传到方法外；out 只可以把值传到方法外

注意：string 作为特殊的引用类型，其操作是与值类型看齐的，若要将方法内对形参赋值后的结果传递出来，需要加上 ref 或 out 关键字。

---->详解

### 11.C#中什么是装箱和拆箱?

装箱：把值类型转换成引用类型

拆箱：把引用类型转换成值类型

装箱：对值类型在堆中分配一个对象实例，并将该值复制到新的对象中。

（1）第一步：新分配托管堆内存(大小为值类型实例大小加上一个方法表指针。

（2）第二步：将值类型的实例字段拷贝到新分配的内存中。

（3）第三步：返回托管堆中新分配对象的地址。这个地址就是一个指向对象的引用了。

拆箱：检查对象实例，确保它是给定值类型的一个装箱值。将该值从实例复制到值类型变量中。

在装箱时是不需要显式的类型转换的，不过拆箱需要显式的类型转换。

```
int i=0;
System.Object obj=i; //这个过程就是装箱！就是将 i 装箱！
int j=(int)obj;//这个过程 obj 拆箱！
```

---->详解

### 12.C#实现多态的过程中 overload 重载 与 override 重写的区别?

override 重写与 overload 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。实现多态。

重载：是方法的名称相同，参数或参数类型不同；重载是面向过程的概念。

重写：是对基类中的虚方法进行重写。重写是面向对象的概念。

---->详解

### 13.C# 中 static 关键字的作用?

对类有意义的字段和方法使用 static 关键字修饰，称为静态成员，通过类名加访问操作符“.”进行访问；对类的实例有意义的字段和方法不加 static 关键字，称为非静态成员或实例成员。

注：静态字段在内存中只有一个拷贝，非静态字段则是在每个实例对象中拥有一个拷贝。而方法无论是否为静态，在内存中只会有一份拷贝，区别只是通过类名来访问还是通过实例名来访问。

### 14.C# 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。不用 new

**15.C#中索引器的实现过程，是否只能根据数字进行索引，请描述一下**

C#通过提供索引器，可以象处理数组一样处理对象。特别是属性，每一个元素都以一个 get 或 set 方法暴露。索引器不单能索引数字（数组下标），还能索引一些 HASHMAP 的字符串，所以，通常来说，C#中类的索引器通常只有一个，就是 THIS，但也可以有无数个，只要你的参数列表不同就可以了索引器和返回值无关，索引器最大的好处是使代码看上去更自然，更符合实际的思考模式。

微软官方一个示例：

索引器允许类或结构的实例按照与数组相同的方式进行索引。索引器类似于属性，不同之处在于它们的访问器采用参数。在下面的示例中，定义了一个泛型类（class SampleCollection ），并为其提供了简单的 get 和 set 访问器 方法（作为分配和检索值的方法）。Program 类为存储字符串创建了此类的一个实例。

**16.C#中 abstract class 和 interface 有什么区别?**

abstract class abstract 声明抽象类抽象方法，一个类中有抽象方法，那么这个类就是抽象类了。所谓的抽象方法，就是不含主体（不提供实现方法），必须由继承者重写。因此，抽象类不可实例化，只能通

过继承被子类重写。

interface 声明接口，只提供一些方法规约，在 C#8 之前的版本中不提供任何实现，在 C#9 版本也可以支持接口的实现；不能用 public、abstract 等修饰，无字段、常量，无构造函数

两者区别：

1.interface 中不能有字段，而 abstract class 可以有; 2.interface 中不能有 public 等修饰符，而 abstract

class 可以有。3.interface 可以实现多继承

---->详解

### 17.C#中用 sealed 修饰的类有什么特点?

密封，不能继承。

密封类在声明中使用 sealed 修饰符，这样就可以防止该类被其它类继承。如果试图将一个密封类作为其它类的基类，C#将提示出错。理所当然，密封类不能同时又是抽象类，因为抽象总是希望被继承的。

### 18.字符串中 string str=null 和 string str=""和 string str=string.Empty 的区别

string.Empty 相当于“”,Empty 是一个静态只读的字段。string str="" ，初始化对象，并分配一个空字符串的内存空间 string str=null,初始化对象，不会分配内存空间

**19.byte b = 'a'; byte c = 1; byte d = 'ab'; byte e = '啊'; byte g = 256; 这些变量有些错误是错在哪里?**

本题考查的是数据类型能承载数据的大小。

1byte =8bit，1 个汉字=2 个 byte，1 个英文=1 个 byte=8bit

所以 bc 是对的，deg 是错的。'a'是 char 类型，a 错误

java byte 取值范围是-128~127，而 C#里一个 byte 是 0~255

## 20.string 和 StringBuilder 的区别,两者性能的比较

都是引用类型，分配再堆上

StringBuilder 默认容量是 16，可以允许扩充它所封装的字符串中字符的数量.每个 StringBuffer 对象都有一定的缓冲区容量，当字符串大小没有超过容量时，不会分配新的容量，当字符串大小超过容量时，会自动增加容量。

对于简单的字符串连接操作，在性能上 stringbuilder 不一定总是优于 strin 因为 stringbulider 对象的创建也消耗大量的性能，在字符串连接比较少的情况下，过度滥用 stringbuilder 会导致性能的浪费而非节约，只有大量无法预知次数的字符串操作才考虑 stringbuilder 的使用。从最后分析可以看出如果是相对较少的字符串拼接根本看不出太大差别。

Stringbulider 的使用，最好制定合适的容量值，否则优于默认值容量不足而频繁的进行内存分

--->详解

### 21.什么是扩展方法?

一句话解释，扩展方法使你能够向现有类型“添加”方法，无需修改类型

条件：按扩展方法必须满足的条件，1.必须要静态类中的静态方法 2.第一个参数的类型是要扩展的类型，并且需要添加 this 关键字以标识其为扩展方法

建议：通常，只在不得已的情况下才实现扩展方法，并谨慎的实现

使用：不能通过类名调用，直接使用类型来调用

---->详解

### 22. 特性是什么？如何使用?

特性与属性是完全不相同的两个概念，只是在名称上比较相近。Attribute 特性就是关联了一个目标对象的一段配置信息，本质上是一个类，其为目标元素提供关联附加信息，这段附加信息存储在 dll 内的元数据，它本身没什么意义。运行期以反射的方式来获取附加信息

---->详解

### 23.什么叫应用程序域(AppDomain)

一种边界，它由公共语言运行库围绕同一应用程序范围内创建的对象建立（即，从应用程序入口点开始，沿着对象激活的序列的任何位置）。

应用程序域有助于将在一个应用程序中创建的对象与在其他应用程序中创建的对象隔离，以使运行时行为可以预知。

在一个单独的进程中可以存在多个应用程序域。应用程序域可以理解为一种轻量级进程。起到安全的作用。占用资源小。

---->详解

### 24.byte a =255;a+=5;a 的值是多少?

byte 的取值范围是-2 的 8 次方至 2 的 8 次方-1，-256 至 258，a+=1 时，a 的值时 0，a+=5 时，a 的值是 0，所以 a+=5 时，值是 4

### 25.const 和 readonly 有什么区别?

都可以标识一个常量。主要有以下区别：

1)、初始化位置不同。const 必须在声明的同时赋值；readonly 即可以在声明处赋值;

2)、修饰对象不同。const 即可以修饰类的字段，也可以修饰局部变量；readonly 只能修饰类的字段

3)、const 是编译时常量，在编译时确定该值；readonly 是运行时常量，在运行时确定该值。

4)、const 默认是静态的；而 readonly 如果设置成静态需要显示声明

5)、修饰引用类型时不同，const 只能修饰 string 或值为 null 的其他引用类型；readonly 可以是任何类型。

---->详解

**26.分析下面代码，a、b 的值是多少?**

```
string strTmp = "a1某某某";
int a = System.Text.Encoding.Default.GetBytes(strTmp).Length;
int b = strTmp.Length;
```

分析：一个字母、数字占一个 byte，一个中文占占两个 byte，所以 a=8,b=5

**27.String s = new String(“ xyz ”); 创建了几个 String Object?**

两个对象，一个是“xyz”,一个是指向“xyz”的引用对象 s。

**28.c#可否对内存直接操作**

C#在 unsafe 模式下可以使用指针对内存进行操作，但在托管模式下不可以使用指针，C#NET 默认不运行带指针的，需要设置下，选择项目右键->属性->选择生成->“允许不安全代码”打勾->保存

---->详解

**29. 什么是强类型，什么是弱类型？哪种更好些？为什么?**

强类型是在编译的时候就确定类型的数据，在执行时类型不能更改，而弱类型在执行的时候才会确定类型。没有好不好，二者各有好处，强类型安全，因为它事先已经确定好了，而且效率高。一般用于编译

型编程语言，如 c++,java,c#,pascal 等,弱类型相比而言不安全，在运行的时候容易出现错误，但它灵活，多用于解释型编程语言，如 javascript 等

---->详解

**30.Math.Round(11.5) 等于多少？ Math.Round(-11.5) 等于多少?Math.Round(-11.3)呢?**

Math.Round(11.5)=12

Math.Round(-11.5)=-12

Math.Round(11.3)=-11

**31.&和&&的区别**

*相同点*

&和&&都可作逻辑与的运算符，表示逻辑与（and），当运算符两边的表达式的结果都为 true 时，其结果才为 true，否则，只要有一方为 false，则结果为 false。

（ps：当要用到逻辑与的时候&是毫无意义，&本身就不是干这个的）

```
string strTmp = "a1 某某某";
int a = System.Text.Encoding.Default.GetBytes(strTmp).Length;
int b = strTmp.Length;
```

*不同点*

```
if(loginUser!=null&&string.IsnullOrEmpty(loginUser.UserName))
```

&&具有短路的功能，即如果第一个表达式为 false，则不再计算第二个表达式，对于上面的表达式，当 loginUser 为 null 时，后面的表达式不会执行，所以不会出现 NullPointerException 如果将&&改为&，则会抛出 NullPointerException 异常。（ps：所以说当要用到逻辑与的时候&是毫无意义的）& 是用作位运算的。

*总结*

&是位运算，返回结果是 int 类型 &&是逻辑运算，返回结果是 bool 类型

### 32.i++和++i 有什么区别?

1).i++是先赋值，然后再自增； ++i 是先自增，后赋值。

2).i=0，i++=0，++i=1；Console.WriteLine(++i==i++); 结果位 true

### 33.as 和 is 的区别

as 在转换的同时判断兼容性，如果无法进行转换，返回位 null（没有产生新的对象），as 转换是否成功判断的依据是是否位 null is 只是做类型兼容性判断，并不执行真正的类型转换，返回 true 或 false，对象为 null 也会返回 false。

as 比 is 效率更高，as 只需要做一次类型兼容检查

--->详解

### 34.谈谈 final、finally 的区别。

final：不能作为父类被继承。一个类不能声明是 final，又声明为 abstract。

finally：用于 try{}catch{}finally{}结构，用于异常处理时执行任何清除操作。

--->详解

### 35.简述 C#成员修饰符

abstract:指示该方法或属性没有实现。

const:指定域或局部变量的值不能被改动。

event:声明一个事件。

extern:指示方法在外部实现。

override:对由基类继承成员的新实现。

readonly:指示一个域只能在声明时以及相同类的内部被赋值。

static:指示一个成员属于类型本身,而不是属于特定的对象。

virtual:指示一个方法或存取器的实现可以在继承类中被覆盖。

--->详解

### 36.什么是匿名类，有什么好处?

不用定义、没有名字的类，使用一次便可丢弃。好处是简单、随意、临时的。

```
var x = new {Name="张三",Age=20,Like="LOL"};
```

--->详解

### 37.说说什么是逐字字符串

在普通字符串中，反斜杠字符是转义字符。而在逐字字符串（Verbatim Strings）中，字符将被编程器按照原义进行解释。使用逐字字符串只需在字符串前面加上 @ 符号。

```
// 逐字字符串：转义符
var filename = @"c:\temp\newfile.txt";
Console.WriteLine(filenaame);
// 逐字字符串：多行文本
var multiLine = @"This is a
multiline paragraph.";
Console.WriteLine(multiLine);
// 非逐字字符串
var escapedFilename = "c:\temp\newfile.txt";
Console.WriteLine(escapedFilename);
```

输出结果：

```
c:\temp\newfile.txt
This is a
multiline paragraph.
c:   emp
ewfile.txt
```

逐字字符串中唯一不被原样解释的字符是双引号。由于双引号是定义字符串的关键字符，所以在逐字字 符串中要表达双引号需要用双引号进行转义。

```
var str = @"""I don't think so"", he said.";
Console.WriteLine(str);
```

输出结果：

```
"I don't think so", he said.
```

在逐字字符串中也可以 $ 符号实现字符串内插值。

```
Testing \n 1 2 3
```

输出结果：

```
Testing \n 1 2 3
```

**38.列举你知道的数字格式化转换**

可使用“0”和“#”占位符进行补位。“0”表示位数不够位数就补充“0”，小数部分如果位数多了则会四舍五入；“#”表示占位，用于辅助“0”进行补位。如下例：

```
// “0”描述：占位符，如果可能，填充位
Console.WriteLine(string.Format("{0:000000}", 1234)); // 结果：001234
 // “#”描述：占位符，如果可能，填充位
Console.WriteLine(string.Format("{0:######}", 1234)); // 结果：1234
Console.WriteLine(string.Format("{0:#0####}", 1234)); // 结果：01234
Console.WriteLine(string.Format("{0:0#0####}", 1234)); // 结果：0001234
// "."描述：小数点
Console.WriteLine(string.Format("{0:000.000}", 1234));  // 结果：1234.000
Console.WriteLine(string.Format("{0:000.000}", 4321.12543)); // 结果：4321.125
// ","描述：千分表示
Console.WriteLine(string.Format("{0:0,0}", 1234567)); //结果：1,234,567
// "%"描述：格式化为百分数
 Console.WriteLine(string.Format("{0:0%}", 1234));  // 结果：123400%
Console.WriteLine(string.Format("{0:#%}", 1234.125));   // 结果：123413%
Console.WriteLine(string.Format("{0:0.00%}", 1234));   // 结果：123400.00%
Console.WriteLine(string.Format("{0:#.00%}", 1234.125)); // 结果：123412.50%
```

内置快捷字母格式化用法：

```
// E-科学计数法表示
  Console.WriteLine((25000).ToString("E")); // 结果：2.500000E+004
  // C-货币表示，带有逗号分隔符，默认小数点后保留两位，四舍五入
  Console.WriteLine((2.5).ToString("C")); // 结果：¥2.50
   // D[length]-十进制数
   Console.WriteLine((25).ToString("D5")); // 结果：00025
    // F[precision]-浮点数，保留小数位数(四舍五入)
   Console.WriteLine((25).ToString("F2")); // 结果：25.00
    // G[digits]-常规，保留指定位数的有效数字，四舍五入
   Console.WriteLine((2.52).ToString("G2")); // 结果：2.5
   // N-带有逗号分隔符，默认小数点后保留两位，四舍五入
   Console.WriteLine((2500000).ToString("N")); // 结果：2,500,000.00
   // X-十六进制，非整型将产生格式异常
   Console.WriteLine((255).ToString("X")); // 结果:FF
```

ToString 也可以自定义补零格式化：

```
   Console.WriteLine((15).ToString("000"));                // 结果：015
   Console.WriteLine((15).ToString("value is 0"));         // 结果：value
is 15
   Console.WriteLine((10.456).ToString("0.00"));          // 结果：10.46
    Console.WriteLine((10.456).ToString("00"));            // 结果：10
   Console.WriteLine((10.456).ToString("value is 0.0")); // 结果：value
is 10.5

```

## 39.说说字符串拼接、字符串内插法

将数组中的字符串拼接成一个字符串：

```
var parts = new[] { "Foo", "Bar", "Fizz", "Buzz"};
 var joined = string.Join(", ", parts);
 // joined = "Foo, Bar, Fizz, Buzz"
```

以下四种方式都可以达到相同的字符串拼接的目的：

```
string first = "Hello";
 stringsecond = "World";
 string foo = first + " " + second;
 string foo = string.Concat(first, " ", second);
 string foo = string.Format("{0} {1}", firstname, lastname);
 string foo = $"{firstname} {lastname}";
```

字符串内插法简单用法：

```
var name = "World";
         var str =$"Hello, {name}!";
 // str = "Hello, World!"
```

带日期格式化

```
var date = DateTime.Now;
 var str = $"Today is {date:yyyy-MM-dd}! ";
```

补齐格式化（Padding）：

```
  var number = 42;
            // 向左补齐
            var str = $"The answer to life, the universe and everything is
{number,5}.";
            // str = "The answer to life, the universe and everything is ___42
('_'表示空格)
            // 向右补齐
            var str = $"The answer to life, the universe and everything is
${number,-5}.";
            // str = "The answer to life, the universe and everything is 42___
```

结合内置快捷字母格式化：

```
var amount = 2.5;
var str = $"It costs {amount:C}";
// str = "¥2.50"
var number = 42;
 var str = $"The answer to life, the universe and everything is
{number,5:f1}.";
 // str = "The answer to life, the universe and everything is
___42.1"
```

--->详解

## 40.什么是虚函数？什么是抽象函数?

虚函数：没有实现的，可以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。

--->详解

## 41.Webservice 与 Webapi 的区别?

**Webservice**

它是基于 SOAP 协议的，数据格式是 XML (SOAP )

只支持 HTTP 协议

不是开源的，但可以被任意一个了解 XML 的人使用

它只能部署在 IIS 上

**Webapi**

Web API 是一个开源的、理想的、构建 REST-ful 服务的技术

它也支持 MVC 的特征，像路由、控制器、action、filter 等

它可以部署在应用程序和 IIS 上

Response 可以被 Web API 的 MediaTypeFormatter 转换成 Json、XML 或者任何你想转换的格式。

--->详解

## 42.ADO.NET 常用对象有哪些?

**Connection:** 主要是开启程序和数据库之间的连接。没有利用连接对象将数据库打开，是无法从数据库中取得数据的。Close 和 Dispose 的区别，Close 以后还可以 Open，Dispose 以后则不能再用。

**Command**：主要可以用来对数据库发出一些指令，例如可以对数据库下达查询、新增、修改、删除数

据等指令，以及调用存在数据库中的存储过程等。这个对象是架构在 Connection 对象上，也就是 Command 对象是通过在 Connection 对象连接到数据源。

**DataAdapter**：主要是在数据源以及 DataSet 之间执行数据传输的工作，它可以透过 Command 对象下达命令后，并将取得的数据放入 DataSet 对象中。这个对象是架构在 Command 对象上，并提供了许多配合 DataSet 使用的功能。

**DataSet**：这个对象可以视为一个暂存区（Cache），可以把从数据库中所查询到的数据保留起来，甚至可以将整个数据库显示出来，DataSet 是放在内存中的。DataSet 的能力不只是可以储存多个 Table 而已，还可以透过 DataAdapter 对象取得一些例如主键等的数据表结构，并可以记录数据表间的关联。

DataSet 对象可以说是 ADO.NET 中重量级的对象，这个对象架构在 DataAdapter 对象上，本身不具备和数据源沟通的能力；也就是说我们是将

DataAdapter 对象当做 DataSet 对象以及数据源间传输数据的桥梁。DataSet 包含若干 DataTable、DataTableTable 包含若干 DataRow。

**DataReader**：当我们只需要循序的读取数据而不需要其它操作时，可以使用 DataReader 对象。

DataReader 对象只是一次一次向下循序的读取数据源中的数据，这些数据是存在数据库服务器中的，而不是一次性加载到程序的内存中的，只能（通过游标）读取当前行的数据，而且这些数据是只读的，并不允许作其它的操作。因为 DataReader 在读取数据的时候限制了每次只读取一条，而且只能只读，所以使用起来不但节省资源而且效率很好。

--->详解

## 43.什么是托管和非托管?

**托管代码**：是由公共语言运行库（CLR）执行的代码，而不是由操作系统直接执行。有关内存管理（内存申请，内存释放，垃圾回收之类的）全部都是.net 的 CLR 来管理。

**非托管代码**：直接编译成目标计算机码，由操作系统直接执行的代码，内存回收要继承 IDisposable 接口手动回收。

--->详解

## 44.在.NET 托管代码总我们不必担心内存泄漏，这是因为有了?

GC 垃圾收集器。

### 45.什么是 MVC 模式

MVC(Model View Controller)模型 - 视图 - 控制器

在 MVC 项目里 cshtml 就是 View 视图；Model：DataSet、Reader、对象；Controller：CS 逻辑代码。

MVC 是典型的平行关系，没有说谁在上谁在下的关系，模型负责业务领域的事情，视图负责显示的事情，控制器把数据读取出来填充模型后把模型交给视图去处理。而各种验证什么的应该是在模型里处理了。它强制性的使应用程序的输入、处理和输出分开。MVC 最大的好处是将逻辑和页面分离。

--->详解

### 46.能用 foreach 遍历访问的对象的要求

需要实现 IEnumerable 接口或声明 GetEnumerator 方法的类型。

--->详解

### 47.什么是反射?

程序集包含模块，而模块又包括类型，类型下有成员，反射就是管理程序集，模块，类型的对象，它能够动态的创建类型的实例，设置现有对象的类型或者获取现有对象的类型，能调用类型的方法和访问类型的字段属性。它是在运行时创建和使用类型实例。

--->详解

### 48.ORM 中的延迟加载与直接加载有什么异同?

延迟加载（Lazy Loading）只在真正需要进行数据操作的时候再进行加载数据，可以减少不必要的开销。

--->详解

### 49.简述 Func 与 Action 的区别?

Func 是有返回值的委托，Action 是没有返回值的委托。

--->详解

### 50.23 种设计模式分别叫什么名称，如何分类?

分三类：

创建型，行为型，结构型；

**创建型包含**：

1）.单例模式，2）.工厂模式 3）.建造者模式 4）.原型模式 5）.工厂方法模式

**行为型包含**：

1）.策略模式

2）.模板方法模式

3）.观察者模式

4）.迭代子模式

5）.责任链模式

6）.命令模式

7）.备忘录模式

8）.状态模式

9）.访问者模式

10）.中介者模式

11）.解释器模式

**结构型设计模式包含：**

1）.适配器模式

2）.装饰器模式

3）.代理模式

4）.外观模式

5）.桥接模式

6）.组合模式

7）.享元模式

更多初中高面试题可以扫码关注公众号↓ ↓ ↓ ↓ ↓ ↓ ↓

北京群 群1：219690210，群2：377501688，群3：262827065，成都群：209844460
上海群 ：376029918 杭州群：376029918 广州群：344744167 深圳群：542733289
西安群：542733289 高级群：165150112
微信公众号:DotNet开发跳槽


欢迎关注