1. C/S 结构模式与 B/S 结构模式的主要区别是什么?

首先，在系统的性能方面。只要拥有可上网的浏览器，就可以使用 B/S 系统。不过，B/S 结构的客户端只能完成浏览、查询、数据输入等简单功能，绝大部分工作要由服务器承担，这就对服务器提出了很高的要求，无形中增加了用户在这一方面的投入。采用 C/S 结构时，客户端和服务器都承担部分工作，有效利用了客户端和服务器端的资源，使用户不必在硬件上有更多的投入。另外，浏览器页面不便于修改，这为用户定制自己的文件时带来了不便，比如用户想自定义一个报表，用 B/S 结构的系统就比较难完成。

其次，在系统的开发方面，C/S 结构的开发对开发者提出了较高的要求，整个开发过程比较复杂。与 B/S 结构相比，C/S 技术的历史更为“悠久”，从技术成熟度和开发人员普遍掌握的水平来看更为成熟。

第三，系统升级方面。C/S 结构中若有某一模块发生改变，可能要关联到其它模块的变动，使系统升级的成本较大;而 B/S 结构在开发、维护阶段几乎所有的工作都集中在服务器端，只需更新服务器端的软件就可以了。如果系统升级比较频繁，那么 B/S 架构的产品就具有维护工作量少的优势。

第四，安全性方面。在安全性上，B/S 结构则略显不足，毕竟现在网络安全系数不高，只要拥有密码，任何人都可以进入到用户的系统中;而 C/S 结构由于需要特定的客户端软件，并且一般来说都要对客户端加密，甚至可以限定只有某一台计算机可以使用这个客户端，因而对安全性有更多的保障。

2. 可以用 C#编写哪些类型的应用程序?

1) 控制台应用程序。
2) Windows 应用程序。
3) 水晶报表应用程序。
4) ASP.NET Web 应用程序。
5) ASP.NET Web 服务应用程序。
6) ASP.NET 水晶报表 Web 应用程序。
7) 智能设备应用程序。

3. 什么是命名空间?命名空间和类库的关系是什么?

1) 名称空间是对类的一种逻辑上的分组，即将类按照某种关系或联系划分到不同的名称空间下。

2) 名称空间又可以包含其它的名称空间，例如 System.Windows.Forms，是指 System 名称空间下有 Windows 名称空间，Windows 名称空间下有 Forms 名称空间。

4. C#中的数组类型有何特点?

1) 数组一般用于存储同一种类型的数据，包括 Object 类型。
2) 数组是一种引用类型，而不是值类型。
3) C#中除了可以有一维数组、多维数组外，还有交错型数组。

5. C#中不同整型之间进行转换的原则是什么?

在整型之间进行转换时，小范围类型可以隐式转换为大范围类型，但大范围类型转换为小范围类型时需要使用显式转换。

6. 简述装箱和拆箱的过程。

装箱是将值类型隐式地转换为 object 类型或者转换为由该值类型实现了的接口类型。装箱一个数值会为其分配一个对象实例，并把该数值拷贝到新对象中。拆箱是显式地把 object 类型转换成值类型，或者把值类型实现了的接口类型转换成该值类型。

7. C#语言中，值类型和引用类型有何不同?

值类型和引用类型的区别在于，值类型的变量直接存放实际的数据，而引用类型的变量存放的则是数据的地址，即对象的引用。

值类型变量直接把变量的值保存在堆栈中，引用类型的变量把实际数据的地址保存在堆栈中，而实际数据则保存在堆中。注意，堆和堆栈是两个不同的概念，在内存中的存储位置也不相同，堆一般用于存储可变长度的数据，如字符串类型;而堆栈则用于存储固定长度的数据，如整型类型的数据 int(每个 int 变量占用四个字节)。由数据存储的位置可以得知，当把一个值变量赋给另一个值变量时，会在堆栈中保存两个完全相同的值;而把一个引用变量赋给另一个引用变量，则会在堆栈中保存对同一个堆位置的两个引用，即在堆栈中保存的是同一个堆的地址。在进行数据操作时，对于值类型，由于每个变量都有自己的值，因此对一个变量的操作不会影响到其它变量;对于引用类型的变量，对一个变量的数据进行操作就是对这个变量在堆中的数据进行操作，如果两个引用类型的变量引用同一个对象，实际含义就是它们在堆栈中保存的堆的地址相同，因此对一个变量的操作就会影响到引用同一个对象的另一个变量。

a.将一个值类型变量赋给另一个值类型变量时，将复制包含的值。引用类型变量的赋值只复制对对象的引用，而不复制对象本身。

b.值类型不可能派生出新的类型：所有的值类型均隐式派生自 System.ValueType。但与引用类型相同的是，结构也可以实现接口。

c.值类型不可能包含 null 值：然而，可空类型功能允许将 null 赋给值类型。

d.每种值类型均有一个隐式的默认构造函数来初始化该类型的默认值。

8. 结构和类的区别是什么?

1) 结构是一个值类型，保存在栈上，而类是一个引用类型，保存在受管制的堆上。

2) 对结构中的数据进行操作比对类或对象中的数据进行操作速度要快。

3) 一般用结构存储多种类型的数据，当创建一个很多类或对象共用的小型对象时，使用结构效率更高。

9. 错误和异常有什么区别，为什么要进行异常处理，用于异常处理的语句有哪些?

错误是指在执行代码过程中发生的事件，它中断或干扰代码的正常流程并创建异常对象。当错误中断流程时，该程序将尝试寻找异常处理程序(一段告诉程序如何对错误做出响应的代码)，以帮助程序恢复流程。换句话说，错误是一个事件，而异常是该事件创建的对象。

当使用短语“产生异常”时，表示存在问题的方法发生错误，并创建异常对象(包含该错误的信息及发生的时间和位置)来响应该错误。导致出现错误和随后异常的因素包括用户错误、资源失败和编程逻辑失败。这些错误与代码实现特定任务的方法有关，而与该任务的目的无关。

如果不进行异常处理，即不对错误做出响应，程序的健壮性就会大打折扣，甚至无法保证正常运行，所以必须要进行异常处理。

用于异常处理的语句有：try-catch 语句、try-catch-finally 语句、throw 语句。

10.举例说明 new 关键字可用于那些方面?

在 C#中，new 关键字可用作运算符或修饰符。作为运算符用于在堆上创建对象和调用构造函数。作为修饰符用于隐藏基类成员的继承成员。

11. 哪些关键字可以用于版本控制?

override 关键字和 new 关键字均可用于版本控制。

在 C#中，默认情况下方法不是虚拟的。若要使方法成为虚拟方法，必须在基类的方法声明中使用 virtual 修饰符。然后，派生类可以使用 override 关键字重写基类中的虚拟方法，或使用 new 关键字隐藏基类中的虚拟方法。如果 override 关键字和 new 关键字均未指定，编译器将发出警告，并且派生类中的方法将隐藏基类中的方法。

12、重载的关键字 overload,重写的关键字 override,请描述一下他们的区别

重写（Override）的两个函数的函数特征相同，重载（Overload）的两个函数的函数名虽然相同，但函数特征不同。 函数特征包括函数名，参数的类型和个数。Override 是在继承的时候，如果你写的函数与要继承的函数函数特征相同，那么，加上这个关键字，在使用这个子类的这个函数的时候就看不见父类（或超类）的函数了，它被覆盖掉了

13. 使用委托的优点是什么?委托和事件有什么区别和联系?

C#中的委托类似于 C 或 C++中的函数指针。使用委托使程序员可以将方法引用封装在委托对象内。然后可以将该委托对象传递给可调用所引用方法的代码，而不必在编译时知道将调用哪个方法。与 C 或 C++中的函数指针不同，委托是面向对象，而且是类型安全的。

C#中的“事件”是当对象发生某些事情时，类向该类的客户提供通知的一种方法。事件最常见的用途是用于图形用户界面;通常，表示界面中的控件的类具有一些事件，当用户对控件进行某些操作(如单击某个按钮)时，将通知这些事件。

使用委托来声明事件。委托对象封装一个方法，以便可以匿名调用该方法。事件是类允许客户为其提供方法(事件发生时应调用这些方法)的委托的一种方法。事件发生时，将调用其客户提供给它的委托。

14. 简要回答文件和流之间的区别和联系。

文件(file)和流(stream)即有区别又有联系。文件是在各种媒质上(可移动磁盘、硬盘、CD 等)永久存储的数据的有序集合。它是一种进行数据读写操作的基本对象。通常情况下，文件按照树状目录进行组织，每个文件都有文件名、文件所在路径、创建时间、访问权限等属性。

流是字节序列的抽象概念，例如文件、输入输出设备、内部进程通信管道或者 TCP/IP 套接字等均可以看成流。流提供一种向后备存储器写入字节和从后备存储器读取字节的方式。

15. 组件与控件的主要区别是什么?

组件是指可重复使用并且可以和其他对象进行交互的对象。组件(component)是靠类实现的。控件是能够提供用户界面接口(UI)功能的组件。换句话说就是，控件是具有用户界面功能的组件。

所有控件肯定都是组件，但并不是每个组件都一定是控件。

16. 控件有几种类型?各有什么特点?

控件分为：复合、扩展和自定义三类。

复合控件是封装在公共容器内的 Windows 窗体控件的集合。这种控件有时称为“用户控件”，包含的控件称为“构成控件”。复合控件包含与每个包含的 Windows 窗体控件相关联的所有固有功能，允许有选择地公开和绑定它们的属性。复合控件还提供了大量的默认键盘处理功能，不需要任

何额外的开发。复合控件从 UserControl 类派生而来。

扩展控件是从任何现有的 Windows 窗体控件或者自定义控件导出的继承控件。它保留 Windows 窗体控件的所有固有功能，然后通过添加自定义属性、方法或其他功能扩展此固有功能。可以使用此选项重写基控件的绘制逻辑，然后更改该控件的外观以扩展其用户界面。

创建控件的另一种方法是通过从 Control 继承从头开始创建一个控件。Control 类提供控件所需的所有基本功能(包括鼠标和键盘处理事件)，但不提供控件特定的功能或图形界面。若要实现自定义控件，必须编写该控件的 OnPaint 事件的代码，以及所需的任何功能特定的代码。

17. 描述线程与进程的区别？

线程(Thread)与进程（Process）二者都定义了某种边界，不同的是进程定义的是应用程序与应用程序之间的边界，不同的进程之间不能共享代码和数据空间，而线程定义的是代码执行堆栈和执行上下文的边界。一个进程可以包括若干个线程，同时创建多个线程来完成某项任务，便是多线程。而同一进程中的不同线程共享代码和数据空间。用一个比喻来说，如果一个家庭代表一个进程，在家庭内部，各个成员就是线程，家庭中的每个成员都有义务对家庭的财富进行积累，同时也有权利对家庭财富进行消费，当面对一个任务的时候，家庭也可以派出几个成员来协同完成，而家庭之外的人则没有办法直接消费不属于自己家庭的财产。

一个程序至少有一个进程,一个进程至少有一个线程.

18.什么是强类型，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,vb 等

19. 阐述面向接口、面向对象、面向方面编程的区别

面向接口更关注的是概念，它的原则是先定义好行为规范，再根据行为规范创建实现，严格的来说，面向接口应该是面向对象中的一部分吧，因为面向对象也强调的是依赖倒置原则，也就是实现依赖于抽象，而抽象不依赖于具体实现，更具比较的应该是面向接口与面向抽象对象，我的体会是面向接口更加灵活，但实现时候，稍微有些代码冗余，而面向抽象可以结合面向接口，先定义接口，再定义抽象类，在抽象类中处理一些公共逻辑，再实现具体实现类。面向对象是对复杂问题的分解。面向方面的编程是一种新概念，它解决了很多面向对象无法解决的问题，比如面向对象技术只能对业务相关的代码模块化，而无法对和业务无关的代码模块化。而面向方面正是解决这一问题的方案，它的关键思想是"将应用程序中的商业逻辑与对其提供支持的通用服务进行分离"。

20. 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

21.从概念上阐述前期绑定（early-binding）和后期绑定（late-binding）的区别?

这个就像是强弱类型的比较相似，前期绑定是在编译的时候就确定了要绑定的数据，而后期绑定是在运行的时候才填充数据。所以前期绑定如果失败，会在编译时报编译错误，而后期绑定失败只有在运行时的时候才发生

22.为什么不提倡 catch(Exception)？

原因可能有两点：1）try..catch 在出现异常的时候影响性能 2）应该捕获更具体得异常，比如 IOExeception,OutOfMemoryException 等

23.a.Equals(b)和 a == b 一样吗？

不一样。多数情况下，a.Equals(b)表示 a 与 b 一致， a==b 表示 a 与 b 的值相等 。也可以有具体重载

24.在对象比较中，对象一致和对象相等分别是指什么？

对象一致是指两个对象是同一个对象，引用相同。而对象相等是指两个对象的值相同，但引用不一定相同

25.什么叫装箱?

装箱（boxing）是将值类型的数据转化成引用类型，int i=3; object o = i ;便是装箱过程，而拆箱(unboxing)是将饮用类型数据转换值类型,比如 int j = (int)o；属于拆箱

26. 如何设计数据库

存储信息的大小，每次扩容的大小，冗余

27. 几十上百万行，如何快速查询出表数据

用分页存储过程

28. Asp.net 如何连接数据库

Connection	连接数据库
Command	执行数据库 SQL 或存储过错命令
DataAdapter	连接数据库，执行数据库 SQL 或存储过错命令，填充 DataSet

29. 什么是事务？

数据库事务是指作为单个逻辑工作单元执行的一系列操作。

数据库事务的 ACID 属性

事务处理可以确保除非事务性单元内的所有操作都成功完成，否则不会永久更新面向数据的资源。通过将一组相关操作组合为一个要么全部成功要么全部失败的单元，可以简化错误恢复并使应用程序更加可靠。一个逻辑工作单元要成为事务，必须满足所谓的 ACID(原子性、一致性、隔离性和持久性)属性：

- 原子性

事务必须是原子工作单元；对于其数据修改，要么全都执行，要么全都不执行。通常，与某个事务关联的操作具有共同的目标，并且是相互依赖的。如果系统只执行这些操作的一个子集，则可能会破坏事务的总体目标。原子性消除了系统处理操作子集的可能性。

- 一致性

事务在完成时，必须使所有的数据都保持一致状态。在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性。事务结束时，所有的内部数据结构（如 B 树索引或双向链表）都必须是正确的。某些维护一致性的责任由应用程序开发人员承担，他们必须确保应用程序已强制

所有已知的完整性约束。例如，当开发用于转帐的应用程序时，应避免在转帐过程中任意移动小数点。

- 隔离性

由并发事务所作的修改必须与任何其它并发事务所作的修改隔离。事务查看数据时数据所处的状态，要么是另一并发事务修改它之前的状态，要么是另一事务修改它之后的状态，事务不会查看中间状态的数据。这称为可串行性，因为它能够重新装载起始数据，并且重播一系列事务，以使数据结束时的状态与原始事务执行的状态相同。当事务可序列化时将获得最高的隔离级别。在此级别上，从一组可并行执行的事务获得的结果与通过连续运行每个事务所获得的结果相同。由于高度隔离会限制可并行执行的事务数，所以一些应用程序降低隔离级别以换取更大的吞吐量。

- 持久性

事务完成之后，它对于系统的影响是永久性的。该修改即使出现致命的系统故障也将一直保持。

30. string = null 和 string ＝ “”的区别

前者没有分配内存空间，后着分配了

31. 存储过程和 sql 语句的优缺点

存储过程的优缺点：

优点：

1.由于应用程序随着时间推移会不断更改，增删功能，T－SQL 过程代码会变得更复杂，StoredProcedure 为封装此代码提供了一个替换位置。

2.执行计划（存储过程在首次运行时将被编译，这将产生一个执行计划－－ 实际上是 Microsoft SQL Server 为在存储过程中获取由 T-SQL 指定的结果而必须采取的步骤的记录。）缓存改善性能。

但 sql server 新版本，执行计划已针对所有 T-SQL 批处理进行了缓存，而不管它们是否在存储过程中，所以没比较优势了。

3.存储过程可以用于降低网络流量，存储过程代码直接存储于数据库中，所以不会产生大量 T-sql 语句的代码流量。

4.使用存储过程使您能够增强对执行计划的重复使用，由此可以通过使用远程过程调用 (RPC) 处理服务器上的存储过程而提高性能。RPC 封装参数和调用服务器端过程的方式使引擎能够轻松地找到匹配的执行计划，并只需插入更新的参数值。

5.可维护性高，更新存储过程通常比更改、测试以及重新部署程序集需要较少的时间和精力。

6.代码精简一致，一个存储过程可以用于应用程序代码的不同位置。

7.更好的版本控制，通过使用 Microsoft Visual SourceSafe 或某个其他源代码控制工具，您可以轻松地恢复到或引用旧版本的存储过程。

8.增强安全性：

a、通过向用户授予对存储过程（而不是基于表）的访问权限，它们可以提供对特定数据的访问；

b、提高代码安全，防止 SQL 注入（但未彻底解决，例如，将数据操作语言－－DML，附加到输入参数）；

c、SqlParameter 类指定存储过程参数的数据类型，作为深层次防御性策略的一部分，可以验证用户提供的值类型（但也不是万无一失，还是应该传递至数据库前得到附加验证）。

缺点：

1.如果更改范围大到需要对输入存储过程的参数进行更改，或者要更改由其返回的数据，则您仍需要更新程序集中的代码以添加参数、更新 GetValue() 调用，等等，这时候估计比较繁琐了。

2.可移植性差

由于存储过程将应用程序绑定到 SQL Server，因此使用存储过程封装业务逻辑将限制应用程序的可移植性。如果应用程序的可移植性在您的环境中非常重要，则将业务逻辑封装在不特定于 RDBMS 的中间层中可能是一个更佳的选择。

Sql 语句灵活，可移植性强，查询速度比存储过程慢些

32. 说明一下软件开发流程？

分析（需要，概要，详细），开发（编程，单元测试），测试（集成测试），维护

33. 如果做到编码规范？

方法，类，变量尽量写有意义的单词。注释有写清楚，但不要罗唆

34. try catch finally 中 catch 和 finally 的作用

catch 扑获异常，finally 不管代码是否出现异常都执行

35. HashMap 和 Hashtable 的区别。

答：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.

36.DataReader 和 DataSet 的异同

DataReader 使用时始终占用 SqlConnection,在线操作数据库..任何对 SqlConnection 的操作都会引发 DataReader 的异常..因为 DataReader 每次只在内存中加载一条数据,所以占用的内存是很小的..因为 DataReader 的特殊性和高性能.所以 DataReader 是只进的..你读了第一条后就不能再去读取第一条了..

DataSet 则是将数据一次性加载在内存中.抛弃数据库连接..读取完毕即放弃数据库连接..因为 DataSet 将数据全部加载在内存中.所以比较消耗内存...但是确比 DataReader 要灵活..可以动态的添加行,列,数据.对数据库进行回传更新操作...

37. .net 中读写数据库需要用到哪些类？他们的作用

这个类自己可以写的啊,基类是 configuration,connection,command 等都要用到.

Connection	打开数据库连接
Command	执行数据库命令
DataAdapter	连接数据，执行数据库命令，填充 DataSet
DataSet	数据在内存中的缓存，数据结构
DataReader	只读向前的读取数据库

38.如何理解.net 中的垃圾回收机制。

.NET Framework 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。

39.CTS，CLR，CLS

CTS：通用类型系统（Commom Type System）；

CLS：通用语言规范(Common Language Specification)；
CLR：公共语言运行库（Common Language Runtime）；

40.Static 和 非 Static 的区别：

（1）、用 Static 声明的方法和变量，不需要实例化该类就调用；
（2）、Static 的，就一定要用实例化的对象来调用，即用 new 来实例化。
举例说：
如果有一个类 People，有一个 Static 的方法 MiaoShu(), 调用方法就是 People.MisoShu()
有一个非 Static 的方法 getName(), 调用方法就是 People p= new People(); p.getName();

41.访问修饰符

private：关键字指示成员在类的外部不可见；
public：关键字指示成员在类的外部可见，并对所有派生类可见（在整个项目中都可见）；
protected：关键字指示成员在类的外部不可见，对所有派生类可见；
internal：关键字指示成员在类的外部可见，并对所有的派生类可见；
protected internal：关键字组合指示成员在类的外部可见，并对所有的派生类可见

42.堆和栈

栈是编译期间就分配好的内存空间，因此你的代码中必须就栈的大小有明确的定义；局部值类型变量、值类型参数等都在栈内存中。

堆是程序运行期间动态分配的内存空间，你可以根据程序的运行情况确定要分配的堆内存的大小

43、关键字：

（1）、this 关键字
用于引用类的当前实例，也包括继承而来的方法，通常可以隐藏 this：
a.显得被相似的名称隐藏的成员；
b.将对象作为参数传递到其它方法；
c.声明索引器；
（2）、base
用于从派生类中访问基类的成员（例如特性和变量），也可以使用 base 关键字范围来自基类的方法，功能：
a.调用基类上已被其他方法重写的方法；
b.制定创建派生类实例时应调用的基类构造函数。
（3）、new
a.实例化如：New Class()
b.public new 隐藏基类的方法
c.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。
（4）、using
在 C#中，使用 using 关键字自动调用 Dispose()方法，如：conn 对象只在 using 块中有效，并且在该块执行之后自动释放。

使用 using 关键字是为了确保执行如下操作的优秀方法：当不再需要某些资源（特别是 COM 对象和非托管代码，CLR 中国的垃圾收集器不会自动卸载这些资源）时适当地释放这些资源。

44.C#中的接口和类有什么异同。

异：

不能直接实例化接口。

接口不包含方法的实现。

接口、类和结构可从多个接口继承。但是 C# 只支持单继承：类只能从一个基类继承实现。

类定义可在不同的源文件之间进行拆分。

同：

接口、类和结构可从多个接口继承。

接口类似于抽象基类：继承接口的任何非抽象类型都必须实现接口的所有成员。

接口可以包含事件、索引器、方法和属性。

都可以实现多个接口。

45.什么是 Interface？它与 Abstract Class 有什么区别？简要回答抽象类和接口的主要区别。

接口(Interface)是用来定义行为规范的，不会有具体实现，而抽象类除定义行为规范外，可以有部分实现，但一个类能实现多个接口，但只能继承一个父类

抽象类和接口的一个主要差别是：类可以实现多个接口，但仅能从一个抽象类或任何其它类型的类继承。

46、请描述一下.Net 架构

Microsoft .NET 框架是生成、部署和运行 Web 服务及应用程序的平台。它提供了一个生产率高且基于标准的多语言环境，用于将现有投资与下一代应用程序和服务集成，同时提供了解决 Internet 规模应用程序的部署和操作难题的灵活性。.NET 框架由三个主要部分组成：公共语言运行库、统一类库的分层集合和称为 ASP.NET 的 Active Server Pages 组件化版本

47. 重载， 覆盖＝＝＝》多态 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，方法的重载是多个方法之间的关系。

3、覆盖要求方法签名相同；重载要求方法签名不同。＝＝>方法名、参数列表、返回类型不能构成重载

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。

A overload

【方法的签名】：参数个数＋参数类型＋参数顺序

返回类型不能构成重载

B override, virtual

子类覆盖父类中对应的虚函数

C，override与overload的区别

a. overload在同一个类里，不同的【方法签名】
b. override在不同的类里边，并且这两个类存在继承关系，并且子类的方法要覆盖父类的同签名的方法，
c. overload是多个方法、override是一个方法
d. 继承 的时候，子类首先继承父类的构造方法
Override的时候，首先继承父类的构造方法，如果构造方法调用了虚函数，那么紧接着调用子类的覆盖方法
然后，才进入子类的构造方法

48. GC是什么？为什么要有GC?::Gallery Collection

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()
//Ds.Dispose();
//net机制，隔一定的时间，它会自动释放无用的资源（内存）

49. 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。抽象类必须被继承，而final必须不被继承。。。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；更不更被override】

===finally【try..catch..finally】－再异常处理时提供 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。。

===finalize() 方法是在垃圾收集器删除对像之前对这个对象调用的。

50. 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.getMaxID() select max(id) from TableName

51. Session有什么重大BUG，微软提出了什么方法加以解决？【Session会丢失】

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。

52. 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。

1.Session,Cookie,Application
2.Get: xxx.aspx?id=5&name=张三:::Request.QueryString['id']
3.Post:把整个表单提交过去，:::Request.Params['控件name']
4.Server.Transfer('target.aspx',true);
5.ViewState['name']

6.数据库
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，不安全
－－－》input type='hidden' 简单，可能被伪造：：ViewState['namexxx'] = 'zxxxa'; string strName = ViewState['namexxx'].ToString();
url参数 简单，显示于地址栏，长度有限::最大传递的参数不能超过2k
数据库 稳定，安全，但性能相对弱
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
input ttype="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

53. 软件开发过程一般有几个阶段？每个阶段的作用？

需求分析（分析系统的所有功能），
系统设计：(概要设计(数据结构设计)、详细设计（开发进度、技术难点））
架构设计：设计系统架构，B/S, C/S, 三层、N层
代码编写：只占整个开发的30%左右
QA：quality Assure：：测试
部署：CS,打包，安装；BS：发布网站

54. 什么叫做SQL注入，如何防止？请举例说明。

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中随便输入，在t_pwd中输入' admin and 1=1-- '就可以进入系统了。
利用sql关键字对网站进行攻击。过滤关键字'等

55. XML 与 HTML 的主要区别

1. XML是严格区分大小写字母的，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。

56. .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。

57. ADO.NET相对于ADO等主要有什么改进？

1:ado.net不依赖于ole db提供程序(odbc，数据连接桥),而是使用.net托管提供的程序,
2:不使用com，【中间件】
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

58.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(比如说死循环)资源分配不当，一个资源每次只能被一个进程使用,未使用完前，不能强行剥夺;否则容易造成死锁

59.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。

60.ASP.net的身份验证方式有哪些？

windows,forms,passport
1)windows验证，使用windows帐户进行验证
2）forms验证：
3) passport验证:asp.net自带的验证机制,可以在配置文件里进行安全设置

61.概述反射和序列化

反射: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:序列化是将对象转换为容易传输的格式的过程。
例如，可以序列化一个对象，然后使用 HTTP 通过 Internet 在客户端和服务器之间传输该对象。在另一端，反序列化将从该流重新构造对象。

62.什么叫做.net?请说说你对.net的理解？？

第一、.net是一种思想：为了下一代互联网跨平台更加方便，操作更加简单而开发的一系列工具。
第二、是一个大框架、包括三大部分：CLR(Common Language Runtime),一套功能极其强大的基础类库(System.)
集成了多种开发语言（C#,VB.Net, C++.Net, J#）
ASP.NET,ADO.NET, 一系列方便的可视化组件

63.什么是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。