# 常见面试题简答---刘振华

# 值类型与引用类型

## 简单回答

C#中，一般情况下值类型存在它申明的地方，引用类型存在托管堆上。值类型转换引用类型叫装箱，引用类型转换值类型叫拆箱。

## 高级回答

一般.Net的程序在程序运行的时候会主要开辟下面几大块内存空间：

全局数据区：存放全局变量，静态数据，常量

代码区：存放所有的程序代码

线程栈区：存放为运行而分配的局部变量，参数，返回数据，返回地址等，

托管堆区：即自由存储区

### **值类型**

    对于值类型的实例，CLR在运行时有两种分配方式：(1) 如果该值类型的实例作为类型中的方法(Method)中的局部变量，则该实例被创建在线程栈上；(2) 如果该值类型的实例作为类型的成员，则该实例作为引用类型(引用类型在GC堆或者LOH上创建)的实例的一部分，被创建在GC堆上

### 引用类型     对于引用类型的实例，CLR在运行时也有两种分配方式：(1) 如果该引用类型的实例的Size<85000Byte，则该实例被创建在GC(Garbage Collection)堆上(当CLR在分配和回收对象时，GC可能会对GC堆进行压缩)；(2) 如果该引用类型的实例的Size>=85000byte，则该实例被创建在LOH(Large Object Heap)上(LOH不会被压缩)。

### 更深的扩展请查看

<https://kb.cnblogs.com/page/42318/>

<http://www.cnblogs.com/happyhippy/archive/2007/04/12/710929.html>

# .Net下页面之间传值方式

## 简单回答

Session、Cookie、Application、Server.Transfer、window.localStorage（基于Html5）、Cache、数据库、GET、POST、PUT、DELETE。

## 高级回答

#### GET，POST，PUT，DELETE

Http定义了与服务器交互的不同方法，最基本的方法有4种，分别是GET，POST，PUT，DELETE。URL全称是资源描述符，我们可以这样认为：一个URL地址，它用于描述一个网络上的资源，而HTTP中的GET，POST，PUT，DELETE就对应着对这个资源的查，改，增，删4个操作。到这里，大家应该有个大概的了解了，GET一般用于获取/查询资源信息，而POST一般用于更新资源信息，早期的系统由于不支持DELETE，因此PUT和DELETE用的较少。

Get与Post的区别请查看：<http://www.techweb.com.cn/network/system/2016-10-11/2407736.shtml>

#### Session

在WEB开发中，服务器可以为每个用户浏览器创建一个会话对象（session对象），注意：一个浏览器独占一个session对象(默认情况下)。因此，在需要保存用户数据时，服务器程序可以把用户数据写到用户浏览器独占的session中，当用户使用浏览器访问其它页面，可以从用户的session中取出该用户的数据，为用户服务。

服务器创建session出来后，会把session的id号发给客户端,客户端写入cookie，这样，只要客户机的浏览器不关，再去访问服务器时，都会带着session的id号去，服务器发现客户机浏览器带session id过来了，就会使用内存中与之对应的session为之服务。

Sssion的详细解析参看：<http://justsee.iteye.com/blog/1570652>

#### Cookie

Cookie 是在 HTTP 协议下，服务器或脚本可以维护客户工作站上信息的一种方式。Cookie 是由 Web 服务器保存在用户浏览器（客户端）上的小文本文件，它可以包含有关用户的信息。无论何时用户链接到服务器，Web 站点都可以访问 Cookie 信息 。

目前有些 Cookie 是临时的，有些则是持续的。临时的 Cookie 只在浏览器上保存一段规定的时间，一旦超过规定的时间，该 Cookie 就会被系统清除 。

持续的 Cookie 则保存在用户的 Cookie 文件中，下一次用户返回时，仍然可以对它进行调用。在 Cookie 文件中保存 Cookie，有些用户担心 Cookie 中的用户信息被一些别有用心的人窃取，而造成一定的损害。其实，网站以外的用户无法跨过网站来获得 Cookie 信息。如果因为这种担心而屏蔽 Cookie，肯定会因此拒绝访问许多站点页面。因为，当今有许多 Web 站点开发人员使用 Cookie 技术，例如 Session 对象的使用就离不开 Cookie 的支持 。

Cookie深入了解：<https://baike.baidu.com/item/cookie/1119?fr=aladdin>

#### Application

Application对象的作用范围是整个全局，也就是说对所有用户都有效。它在整个应用程序生命周期中都是有效的，类似于使用全局变量一样，所以可以在不同页面中对它进行存取。它和Session变量的区别在于，前者是所有的用户共用的全局变量，后者是各个用户独有的全局变量。

　　优点：1.使用简单，消耗较少的服务器资源。

　　　　　2.不仅能传递简单数据，还能传递对象。

　　　　　3.数据量大小是不限制的。

　　缺点：1.作为全局变量容易被误操作。所以单个用户使用的变量一般不能用application。

　　使用方法：1.在源页面的代码中创建你需要传递的名称和值构造Application变量:Application["Nmae"]="Value(Or Object)";

　　　　　　　2.在目的页面的代码使用Application变量取出传递的值。Result = Application["Nmae"]

注意：常用lock和unlock方法用来锁定和解锁，为了防止并发修改。

#### Server.Transfer

　　这个才可以说是面象对象开发所使用的方法，其使用Server.Transfer方法把流程从当前页面引导到另一个页面中，新的页面使用前一个页面的应答流，所以这个方法是完全面象对象的，简洁有效。

　　Server.Transfer是从当前的ASPX页面转到新的ASPX页面，服务器端执行新页并输出,在新页面中通过Context.Handler来获得前一个页面传递的各种数据类型的值、表单数据、QueryString.由于重定向完全在服务器端完成，所以客户端浏览器中的URL地址是不会改变的。调用Server.Transfer时，当前的ASPX页面终止执行，执行流程转入另一个ASPX页面，但新的ASPX页面仍使用前一ASPX页面创建的应答流。

　　ps：比较Server.Transfer和Response.Redirect的区别。  
　　　　(1)Server.Transfer在服务器端完成，所以客户端浏览器中的URL地址是不会改变的；Response.Redirect是客户端完成，向服务器端提出新的页面处理请求，所以客户端浏览器中的URL地址是会改变的。  
　　　　(2)Server.Transfer在服务器端完成,不需要客户端提出请求，减少了客户端对服务器端提出请求。[2]  
　　　　(3)Server.Transfer只能够转跳到本地虚拟目录指定的页面,也就是工程项目中的页面，而Response.Redirect则十分灵活,可以跳转到任何URL地址。  
　　　　(4)Server.Transfer可以将前一个页面的各种类型的值传到新的页面；Response.Redirect则只能借助URL中带参数或是结合上面四种办法把各种类型的值传到新的页面。

　　优点：1.直接在服务器端重定向，使用简单方便，减少了客户端对服务器端提出请求。

　　　　　2.可以传递各种数据类型的值和控件的值。

　　缺点：1.客户端浏览器中的URL地址是不改变，会导致在新的页面可能出现一些意想不到的问题。比如如果源页面和目的页面不在同一个虚拟目录或其子目录下，那么使用相对路径的图片、超链接都会导致错误的指向。

　　使用方法：1.在源页面的代码中，使用Page类的Server.Transfer跳到另一个页面传递页面数据：Server.Transfer("b.aspx","false")。

　　　　　　　2.在目的页面中，使用Context.Handler来接收数据：FormerPage formerPage = (FormerPage)Context.Handler; 然后用formerPage的属性和方法来获取前一个页面的值，或者直接用Context.Items["myParameter "]

#### window.localStorage

在HTML5中，新加入了一个localStorage特性，这个特性主要是用来作为本地存储来使用的，解决了cookie存储空间不足的问题(cookie中每条cookie的存储空间为4k)，localStorage中一般浏览器支持的是5M大小，这个在不同的浏览器中localStorage会有所不同。

#### Cache

Cache传值方式与Application类似，

1、应用程序中的缓存机制用于将需要大量服务器资源来创建的对象存储在内存中，以此大大改进应用程序的性能。这个机制同样可以用来传值。

2、与其他方法不同的是，该方法需要设置缓存项优先级和缓存时间。因为当系统内存缺乏时，缓存机制会自动移除很少使用或优先级较低的项，从而造成传值失败。

3、该方法的优点是传递数据的大小和数量无限制，速度快。缺点是缓存机制的操作相对比较复杂。

### 数据库

可以将要传的值存放在数据库，在需要读取的时候，从数据库进行读取。

# 委托

## 简单回答

委托类似方法指针，可以实现把方法当参数，委托是引用类型。

## 高级回答

委托是一个类，它定义了方法的类型，使得可以将方法当作另一个方法的参数来进行传递，这种将方法动态地赋给参数的做法，可以避免在程序中大量使用If-Else(Switch)语句，同时使得程序具有更好的可扩展性。

委托是可以同时赋值多个方法的，即委托对象可以绑定多个方法。当输入参数后，每个方法会按顺序进行迭代处理，并返回最后一个方法的计算结果,委托的这种特性叫多播委托

# 事件

## 简单回答

事件是一种特殊的委托,可以利用事件很方便、安全的实现观察者模式

## 高级回答

因为委托自身可以被重新赋值的特点，在某些情况下会引发安全问题，所以C#封装了一个事件，如果对事件进行反编译，就可以看到事件其实是由一个私有的委托加一个附加委托和一个移出委托的方法组成，所以可以说事件是一种特殊的委托。

# MVC

## 简单回答

MVC是一种模式，M代表Model、V代表视图、C代表控制器，.Net MVC是主要用于实现B/S结构的系统

## 高级回答

MVC全名是Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。

MVC最大优点就是结构清晰、职责明确、[耦合性](https://baike.baidu.com/item/%E8%80%A6%E5%90%88%E6%80%A7)低、重用性高、可维护性高

# MVC路由

## 简单回答

获取URL，通过配置好的正则来解析，获取对应的Controller和Action，并执行对应的Action，得出相应的结果，返回给浏览器

## 高级回答

ASP.NET MVC中，通常地，URL映射并非映射到具体磁盘上的物理文件，这是基于ASP.NET路由特性，主要有两个过程，即URL映射和URL生成（接下来讲解）。在ASP.NET路由中，您可以定义URL模式，该模式包含在处理URL请求时使用的值的占位符。在运行时，运用程序名称后面的URL部分根据您所定义的URL模式分析为离散值。例如，在请求http://server/application/Products/show/beverages时，路由分析器可以将值Products、show和beverages传递给请求的处理程序。相反，在一个不由URL路由管理的请求中，/Products/show/beverages片段将被解释为运用程序中的一个文件的路径。

**路由的定义:**路由是一种被映射到某个处理程序的URL模式。处理程序可能是一个物理文件，如在WebForms运用程序中的aspx文件。处理程序也可能是一个处理请求的类，如MVC应用程序中的控制器。要定义路由，您可以通过指定URL模式、处理程序和路径的名称来创建路由类的实例。

# MVC过滤器

## 简单回答

对请求进行拦截,并做相应的处理, 我们项目做权限验证的时候用到过滤器,来判断是否有权限

## 高级回答

Action执行前或者执行后，通常我们会做一些特殊的操作（比如身份验证，日志，异常，行为截取等）。微软并不想让MVC开发人员去关心和写这部分重复的代码，所以在MVC项目中我们就可以直接使用它提供的Filter的特性帮我们解决。

MVC支持的过滤器类型有四种，分别是：Authorization(授权)、Action（行为）、Result（结果）和Exception（异常）

但是默认实现的只有三种 但是默认实现它们的过滤器只有三种，分别是Authorize（授权），ActionFilter，HandleError（错误处理）；

# 授权过滤器

## 简单回答

主要用于验证用户是否有权限进入控制器或请求, 项目中在权限管理那里用到的授权过滤器，防止用户在没有授权的情况下直接用url访问。

## 高级回答

# 错误过滤器

## 简单回答

捕捉控制器或者请求的错误，我们在项目中用错误过滤器加Log4Net记录所有的未处理的错误。

## 高级回答

# MVC生命周期

## 简单回答

解析请求的URL通过Routing路由配置找到Controller并实例化它,然后调用URL中对应的Action

## 高级回答

# MVC部分视图

## 简单回答

Mvc部分视图可以将其他视图输出到当前视图，如果有多个页面的某个部分都一样，可以考虑用部分视图。实现部分视图的常用方法有：Partial() 、RenderPartial() 、 Action() 、RenderAction() 、 RenderPage()

## 高级回答

# WebApi

## 简单回答

WebApi是微软的一种框架,主要是用于搭建轻量级SOA项目的接口服务。

## 高级回答

Web API最重要的是可以构建面向各种客户端的服务。另外与WCF REST Service不同在于，Web API利用Http协议的各个方面来表达服务(例如 URI/request response header/caching/versioning/content format)，因此就省掉很多配置。

#### WebApi的主要功能

1. 支持基于Http verb (GET, POST, PUT, DELETE)的CRUD (create, retrieve, update, delete)操作

通过不同的http动作表达不同的含义，这样就不需要暴露多个API来支持这些基本操作。

2. 请求的回复通过Http Status Code表达不同含义，并且客户端可以通过Accept header来与服务器协商格式，例如你希望服务器返回JSON格式还是XML格式。

3. 请求的回复格式支持 JSON，XML，并且可以扩展添加其他格式。

4. 原生支持OData。

5. 支持Self-host或者IIS host。

6. 支持大多数MVC功能，例如Routing/Controller/Action Result/Filter/Model Builder/IOC Container/Dependency Injection

# WebApi与Mvc的不同

## 简单回答

MVC主要用来构建网站，既关心数据也关心页面展示，而Web API只关注数据

## 高级回答

1. MVC主要用来构建网站，既关心数据也关心页面展示，而Web API只关注数据
2. Web API支持格式协商，客户端可以通过Accept header通知服务器期望的格式
3. Web API支持Self Host，MVC目前不支持
4. Web API通过不同的http verb表达不同的动作(CRUD)，MVC则通过Action名字表达动作
5. Web API内建于ASP.NET System.Web.Http命名空间下，MVC位于System.Web.Mvc命名空间下，因此model binding/filter/routing等功能有所不同
6. 最后，Web API非常适合构建移动客户端服务

# WebService

## 简单回答

WebService基于SOAP协议，主要用于搭建SOA项目的服务端，.Net下引用WebService直接在添加引用那里引用就好。

## 高级回答

Web service是一个平台独立的，低耦合的，自包含的、基于可编程的web的应用程序，可使用开放的XML（标准通用标记语言下的一个子集）标准来描述、发布、发现、协调和配置这些应用程序，用于开发分布式的互操作的应用程序。

Web Service技术， 能使得运行在不同机器上的不同应用无须借助附加的、专门的第三方软件或硬件， 就可相互交换数据或集成。依据Web Service规范实施的应用之间， 无论它们所使用的语言、 平台或内部协议是什么， 都可以相互交换数据。Web Service是自描述、 自包含的可用网络模块， 可以执行具体的业务功能。Web Service也很容易部署， 因为它们基于一些常规的产业标准以及已有的一些技术，诸如标准通用标记语言下的子集XML、HTTP。Web Service减少了应用接口的花费。Web Service为整个企业甚至多个组织之间的业务流程的集成提供了一个通用机制。

# WCF

## 简单回答

WCF是微软提供的主要用于基于SOA开发的框架，可以实现消息队列、WebService、Socket等功能的服务。

## 高级回答

整合了原有的windows通讯的 .net Remoting，WebService，Socket的机制，并融合有HTTP和FTP的相关技术。

是Windows平台上开发分布式应用最佳的实践方式。

简单的归结为四大部分

1>.网络服务的协议，即用什么网络协议开放客户端接入。

2>.业务服务的协议，即声明服务提供哪些业务。

3>.数据类型声明，即对客户端与服务器端通信的数据部分进行一致化。

4>.传输安全性相关的定义。

它是.NET框架的一部分，由 .NET Framework3.0 开始引入，与Windows Presentation Foundation及Windows Workflow Foundation并行为新一代 Windows操作系统以及 WinFX 的三个重大应用程序开发类库。在 .NET Framework2.0 以及前版本中，微软发展了 Web Service(SOAP with HTTP communication)，.NET Remoting (TCP/HTTP/Pipeline communication) 以及基础的 Winsock 等通信支持。由于各个通信方法的设计方法不同，而且彼此之间也有相互的重叠性，对于开发人员来说，不同的选择会有不同的程序设计模型，而且必须要重新学习，让开发人员在使用中有许多不便。同时，面向服务架构(Service-Oriented Architecture) 也开始盛行于软件工业中，因此微软重新查看了这些通信方法，并设计了一个统一的程序开发模型，对于数据通信提供了最基本最有弹性的支持，这就是 Windows Communication Foundation。

# Ado.Net

## 简单回答

.Net项目用来与数据库交互的com组件,主要有五大对象Connection、Command、DataReader、DataSet、DataAdapter

## 高级回答

ADO.NET的名称起源于ADO（ActiveX Data Objects），是一个COM组件库，用于在以往的Microsoft技术中访问数据。之所以使用ADO.NET名称，是因为Microsoft希望表明，这是在NET编程环境中优先使用的数据访问接口。

ado.net可让开发人员以一致的方式存取资料来源（例如 SQL Server 与 XML），以及透过 OLE DB 和 ODBC 所公开的资料来源。资料共用的消费者应用程序可使用ado.net 来连接至这些资料来源，并且撷取、处理及更新其中所含的资料。

ado.net可将资料管理的资料存取分成不连续的元件，这些元件可分开使用，也可串联使用ado.net也包含 .NET Framework 资料提供者，以用于连接资料库、执行命令和撷取结果。这些结果会直接处理、放入ado.net DataSet 物件中以便利用机器操作 （Ad Hoc）的方式公开给使用者、与多个来源的资料结合，或在各层之间进行传递。DataSet 物件也可以与.NET Framework 资料提供者分开使用，以便管理应用程序本机的资料或来自 XML 的资料。

ado.net类别 （Class） 位于 System.Data.dll 中，而且会与 System.Xml.dll 中的XML 类别整合。

ado.net可为撰写 Managed 程式码的开发人员提供类似于ActiveX Data Objects （ADO）提供给原生元件物件模型 （Component Object Model，COM）开发人员的功能。建议使用ado.net而非ADO来存取.NET 应用程序中的资料。

ADO .NET会提供最直接的方法，让开发人员在 .NET Framework 中进行资料存取。

隐私权声明：

System.Data.dll、System.Data.Design.dll、System.Data.OracleClient.dll、System.Data.SqlXml.dll、System.Data.Linq.dll、System.Data.SqlServerCe.dll 和 System.Data.DataSetExtensions.dll 组件无法区分使用者的私用资料与非私用资料。这些组件不会收集、储存或传输任何使用者的私用资料。不过，协力厂商应用程序可能会使用这些组件来收集、存储或传输使用者的私用资料。

# ORM框架

## 简单回答

对象关系映射，用于实现面向对象编程语言里不同类型系统的数据之间的转换

## 高级回答

因面向对象编程天生与关系型数据库不契合，所以经过

# EF

## 简单回答

EF是微软推出的一种ORM框架，可以用面向对象的方式来操作数据库。有DBFirst、ModelFirst、CodeFirst三种模式。

## 高级回答

# Dapper

## 简单回答

开源的轻量级ORM框架，最大的优点就是执行效率快，因为它是直接对ADO.NET的Connection连接对象进行了扩展

## 高级回答

# 索引

## 简单回答

有点类似我们书籍的目录,可以提升数据库的查询效率,但是要注意的是, 索引不能用于重复特别多和长度特别大的字段,索引的原理是二叉树、B树。

## 高级回答

**全表扫描**：**在数据库中，对无索引的表进行查询一般称为全表扫描。全表扫描是数据库服务器用来搜寻表的每一条记录的过程，直到所有符合给定条件的记录返回为止。**引自：<http://baike.baidu.com/view/2010124.htm?fr=ala0_1_1>

# In 和or是否会全表扫描

## 简单回答

如果使用得当的话会命中索引,如果in或or的数据量特别大就不会命中索引

## 高级回答

or“使用不当”才会引起全表扫描，那么使用得当的话，显然是可以避免全表扫描的。文章的例子也说的很明确。<http://www.zbitedu.com/?action-viewthread-tid-39219>

in和or确实是等效的，数据库会把in转换成or的形式。

# 存储过程

## 简单回答

存储过程就是将一组Sql存储起来，使其可以重用。存储过程经过第一次编译以后，再次调用就不需要再编译，所以执行效率比较快。

## 高级回答

# 事物

## 简单回答

执行一系列Sql，这些Sql要不一起执行，要不就全部执行。事物有四个属性原子性、一致性、隔离性和持久性

## 高级回答

# 事物锁

## 简单回答

在数据库中有两种基本的锁类型：排它锁（Exclusive Locks，即X锁）和共享锁（Share Locks，即S锁）。当数据对象被加上排它锁时，其他的事务不能对它读取和修改。加了共享锁的数据对象可以被其他事务读取，但不能修改。数据库利用这两 种基本的锁类型来对数据库的事务进行并发控制。

## 高级回答

# Jquery选择器

## 简单回答

jQuery 元素选择器和属性选择器允许您通过标签名、属性名或内容对 HTML 元素进行选择。主要有：元素选择器、属性选择器、CSS 选择器

## 高级回答

# 单例模式

## 简单回答

C#中单例模式主要用于实现一个类只有一个实例，有懒汉和饿汉两种实现方式。

懒汉实现方式是当引用实例的时候初始化，需要用两个if判断和加锁，来避免多线程的情况产生多个实例，优点是占用内存少，缺点是可能引发性能问题

饿汉模式方式是直接实例化，优点是提前初始化，性能快，缺点是占用内存多

## 高级回答

# 工厂模式

## 简单回答

工厂模式主要是为了解决new对象产生的耦合问题。

## 高级回答

# 高并发怎么处理

如果是服务器压力大可以多台服务器负载均衡

如果是数据库压力大,而且对数据实时性要求不高可以考虑引入redis等缓存服务器

如果数据量特别大的话,可以考虑给合适的字段加索引, 在适合的情况下分表,分库.

如果是字段特别大,可以考虑引入全文检索。

总之解决高并发的主要手段就是分流并提升处理能力。

# 数据库性能优化