1. **如何把一个数组复制到ArrayList里？反过来从ArrayList到数组呢？**

循环

ArrayList的构造函数可以传入一个ICollection对象。

ArrayList的AddRange方法也可以传入一个ICollection对象。

1. **string str=null和string str=””有什么区别**

从内存分配上解释

string str=null定义了变量，但是没有在堆中分配有内存

string str=””定义了变量，而且在堆中分配了内存，值是空字符串。

1. **StringBuilder有什么作用**

在与非托管代码交互时也有作用。

从string类型和不可变性开始解释。

1. **泛型有什么好处？**

减少代码冗余，可以从具体的例子如List<T>类型解释，List<int>、List<string>等都有Add、Remove、Sort等方法，而且方法逻辑都是一样的，如果都要一个一个实现，那么会有很多方法都是冗余的，而泛型正好可以解决这个问题。

1. **拥有一个字符串数组{“2”,”3”,”4”},如何转换为整数数组。**

这个题目考的是数组转换问题，解决方法有

循环

Array静态方法ConverAll，顺便说一下效率问题。

1. **什么是序列化，.net中是如何实现序列化的？**

序列化概念：把数据转换成特定的格式，如xml、json等格式。

.net中序列化的实现有几个类型

XmlSerialize、JsonFormmater、SoapFormmater、BinaryFormmater

1. **什么是编码?什么是Unicode**
2. **Base64编码主要应用在什么场合**

公钥、证书、邮件附件、网站图片。

1. **字符串池是如何提高系统性能的。**

从字符串的不可变性开始解释

字符串的不可变性导致字符串是线程安全的，所以即便是几个变量指向同一个字符串，也不会出现线程安全问题。字符串池之所以能提高系统性能，是因为线程池的机制，导致在声明相同的字符串变量时，不用额外分配内存，只需要指向字符串池中已有的字符串。

1. **加密字符串类型和普通字符串类型有什么区别？**

加密字符串类型内存是在非托管堆中分配的，不过加密字符串中实现了Dispose和Finalize机制，所以能被垃圾回收器回收。而不同字符串是在托管堆中分配的内存。

1. **.NET设计小组为什么会设计System.Object这个类型。**

这个是CLS的规定，System.Object是所有类型的根，没有System.Object根的类型不是安全类型，不被公共语言系统兼容，不能跟.net中其他语言开发的模块实现良好互操作。

1. **System.Object类型有基类吗**

没有

1. **同一进程下的线程可以共享以下？(BD)**

A. Stack 栈

B. data section 数据段

C. register set 寄存器组

D. file fd 文件描述符

线程的栈和寄存器集都是私有的，线程在切换时会保存一个线程控制块的结构，里面包含了线程的栈和寄存器集等信息，用来下个时间片运行时恢复数据。

1. **后台线程和一般线程有什么区别？**

new Thread方式创建的线程默认都是前台线程，当所有前台线程都终止的时候，进程才会终止，线程终止时，所有后台线程都会被终止。

1. Lock、Monitor和Mutex的区别

这三个作用是一样的，可以实现线程同步

Lock是使用Monitor来实现的。

Monitor可以使用TryEnter来加锁，这样可以防止线程一直等待，所以可以阻止死锁的发生。而Lock内办法阻止死锁。

Mutex是内核级别的对象，消耗较大的资源（这点有点疑问，需要找资料），由于是内核对象，所以可以跨进程操作，也就是实现跨进程的同步。比如果防止程序打开两个实例。

1. 其他线程同步方式
   1. SpinLock 自旋锁。
   2. AutoResetEvent也可以实现类似锁的功能。
   3. 信号量Semaphore,当信号量阈值为1时，可以作为锁使用。当阈值大于1时，往往用来做流量控制。
2. 多线程安全问题
   1. 锁 lock、monitor、mutex、spinlock、semaphore(阈值为1时)
   2. Cas(compare and swap)比较和交换，先取值，对值进行操作后，在存储的时候进行比较交换，如果内存中的值与取出来的值不一样，说明已经被其他线程变更过了，再自旋重复操作。Cas是一个cpu指令，操作是原子性的，也就是在比较并判断是否交换的过程是一步实现的。.net中的InterLock类中有cas操作。
   3. 原子操作。InterLock类操作都是原子操作，包含了自增和cas等操作。
   4. 线程安全类。这些类型是在System.Collection.Concurrent命名空间下。

BlockingCollection<T>

ConcurrentBag<T>

ConcurrentDictionary<Tkey,Tvalue>

ConcurrentQueue<T>

ConcurrentStack<T>

ConcurrentQueue<T>和ConcurrentStack<T>是使用cas+自旋实现的,.net中就是使用Interlocked.CompareExchange+ SpinWait

ConcurrentDictionary<Tkey,Tvalue>有所不同，是使用锁(Monitor)来实现的，不过他会把hash值的范围分块，当插入某个值时，只是锁定该值所在的块，其他值不影响，这也就是说，当几个值同时插入时，如果hash值是在不同的块中，几个值的插入操作不会阻塞。

ConcurrentBag<T>好像是维护有两个集合。（待研究）

1. 什么时候考虑使用多线程？
   1. 避免主线程阻塞。典型的例子就是窗体程序。当某个操作耗时比较长时，不使用多线程，会造成ui线程长期阻塞，给用户带来不好的体验。
   2. 繁琐耗时的任务。如果任务可以拆分时，可以通过多线程来实现高效处理，当然还要看系统cpu核，单核的话反而会影响效率。
   3. IO等待(网络、磁盘、外部设备等IO输入输出)，可以使用多线程和异步，这样可以利用等待时间去完成其他事。
2. 线程间通讯
   1. AsynchronizationContext
   2. ManualResetEvent、ManualResetEventSlim 和AutoResetEvent
   3. Barrier
   4. Semaphore、SemaphoreSlim
3. Session的存储方式
   1. 服务进程中
   2. Sql server
   3. Redis等缓存数据库中。
4. 常见的http code（200、403、404、500）

2开头 （请求成功）表示成功处理了请求的状态代码。

200 （成功） 服务器已成功处理了请求。 通常，这表示服务器提供了请求的网页。

201 （已创建） 请求成功并且服务器创建了新的资源。

202 （已接受） 服务器已接受请求，但尚未处理。

203 （非授权信息） 服务器已成功处理了请求，但返回的信息可能来自另一来源。

204 （无内容） 服务器成功处理了请求，但没有返回任何内容。

205 （重置内容） 服务器成功处理了请求，但没有返回任何内容。

206 （部分内容） 服务器成功处理了部分 GET 请求。

3开头 （请求被重定向）表示要完成请求，需要进一步操作。 通常，这些状态代码用来重定向。

300 （多种选择） 针对请求，服务器可执行多种操作。 服务器可根据请求者 (user agent) 选择一项操作，或提供操作列表供请求者选择。

301 （永久移动） 请求的网页已永久移动到新位置。 服务器返回此响应（对 GET 或 HEAD 请求的响应）时，会自动将请求者转到新位置。

302 （临时移动） 服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求。

303 （查看其他位置） 请求者应当对不同的位置使用单独的 GET 请求来检索响应时，服务器返回此代码。

304 （未修改） 自从上次请求后，请求的网页未修改过。 服务器返回此响应时，不会返回网页内容。

305 （使用代理） 请求者只能使用代理访问请求的网页。 如果服务器返回此响应，还表示请求者应使用代理。

307 （临时重定向） 服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求。

4开头 （请求错误）这些状态代码表示请求可能出错，妨碍了服务器的处理。

400 （错误请求） 服务器不理解请求的语法。

401 （未授权） 请求要求身份验证。 对于需要登录的网页，服务器可能返回此响应。

403 （禁止） 服务器拒绝请求。

404 （未找到） 服务器找不到请求的网页。

405 （方法禁用） 禁用请求中指定的方法。

406 （不接受） 无法使用请求的内容特性响应请求的网页。

407 （需要代理授权） 此状态代码与 401（未授权）类似，但指定请求者应当授权使用代理。

408 （请求超时） 服务器等候请求时发生超时。

409 （冲突） 服务器在完成请求时发生冲突。 服务器必须在响应中包含有关冲突的信息。

410 （已删除） 如果请求的资源已永久删除，服务器就会返回此响应。

411 （需要有效长度） 服务器不接受不含有效内容长度标头字段的请求。

412 （未满足前提条件） 服务器未满足请求者在请求中设置的其中一个前提条件。

413 （请求实体过大） 服务器无法处理请求，因为请求实体过大，超出服务器的处理能力。

414 （请求的 URI 过长） 请求的 URI（通常为网址）过长，服务器无法处理。

415 （不支持的媒体类型） 请求的格式不受请求页面的支持。

416 （请求范围不符合要求） 如果页面无法提供请求的范围，则服务器会返回此状态代码。

417 （未满足期望值） 服务器未满足”期望”请求标头字段的要求。

5开头（服务器错误）这些状态代码表示服务器在尝试处理请求时发生内部错误。 这些错误可能是服务器本身的错误，而不是请求出错。

500 （服务器内部错误） 服务器遇到错误，无法完成请求。

501 （尚未实施） 服务器不具备完成请求的功能。 例如，服务器无法识别请求方法时可能会返回此代码。

502 （错误网关） 服务器作为网关或代理，从上游服务器收到无效响应。

503 （服务不可用） 服务器目前无法使用（由于超载或停机维护）。 通常，这只是暂时状态。

504 （网关超时） 服务器作为网关或代理，但是没有及时从上游服务器收到请求。

505 （HTTP 版本不受支持） 服务器不支持请求中所用的 HTTP 协议版本。

1. 什么是多层架构，什么是mvc
2. Cookie和session的实现原理。
3. Cookies

* 过期时间

Cookie可以设置过期时间

* 存储位置

存储在客户端

* 域名

Cookie是不可跨域名的，但是可以通过设置域名来让一级域名和二级域名共享cookies

[www.helloweenvsfei.com和images.helloweenvsfei.com](http://www.helloweenvsfei.com和images.helloweenvsfei.com)共享以下cookies

cookie.setDomain(".helloweenvsfei.com")

* 路径

cookie.setPath("/session/")

浏览器会将domain和path都相同的cookie保存在一个文件里，cookie间用\*隔开

* 安全性

setSecure(true)，设置之后cookies只会在安全协议https和ssl中传输

1. Session

* 存储位置

存储在服务端，session有个sessionid，这个id是存储在cookies中的，服务器根据cookies中的sessionid获取session

1. Sql注入
2. 跳转和重定向

跳转只能在项目内，重定向可以定向到其他项目

1. 怎么提高反射效率

对于经常使用到的类信息、字段信息和方法信息，使用字典存储起来。在存储的时候，只存储句柄，使用的时候再从句柄获取信息。

句柄有RuntimeTypeHandle、RuntimeFieldHanlde、RuntimeMethodHandle、RuntimeArgumentHandle

在获取的时候使用Type.GetTypeFromHandle()来通过句柄获取对应信息，FieldInfo、MethodInfo也是这样获取信息

1. 委托链MuticastDelegate

委托链是在MuticastDelegate中维护一个委托数组\_InvocationList;当进行Combin操作的时候，把Combin对象的\_InvocationList放入当前MuticastDelegate对象的\_InvocationList中。

1. 事件和委托

事件就是一个委托类型，一般使用event EventHandle<T>来实现事件，EventHandle<T>就是一个泛型的委托对象，其内部使用委托链来实现多个事件的订阅。事件一般有object和TEventArgs两个参数，这样比较符合事件的规范，当然也可以使用自定义的委托来定义事件。event CustomDelegate().不过事件和委托有一个比较大的区别，就是事件具有更良好的封装性，事件的Invoke方法只能在定义事件的类中执行，即便是其子类也不能调用事件的Invoke方法。

1. 元数据存储什么信息

类型信息（包括类型的field信息）、引用程序集信息、字符串字面量