（寄宿这一块没搞懂）

寄宿的概念：

寄宿（Hosting）是指在一个进程中运行公共语言运行时（CLR）的过程。寄宿程序可以控制CLR的行为，包括垃圾回收、线程管理、安全性和其他方面。

Ps：

寄宿程序通常是一个应用程序，例如ASP.NET或SQL Server，它们需要在自己的进程中运行CLR来执行托管代码。寄宿程序可以通过实现一组托管接口来与CLR进行交互，并控制CLR的行为。

任何windows应用程序都能寄宿CLR。但不要通过调用CoCreateInstance来创建CLR COM服务器的实例。相反，你的非托管宿主应该调用MetaHost.h文件中声明的CLRCreateInstance函数。

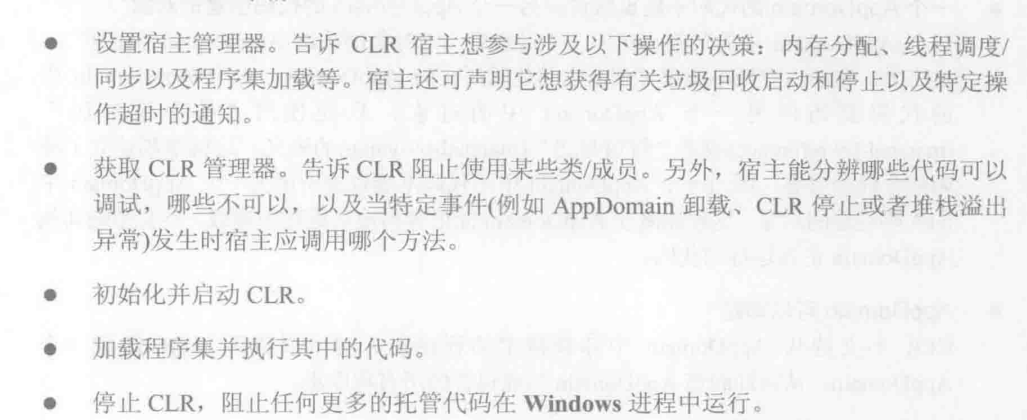
CLR COM服务器是指公共语言运行时（CLR）的COM（组件对象模型）接口。它允许非托管应用程序通过COM接口与CLR进行交互，以便在非托管应用程序中寄宿CLR并执行托管代码。在.NET Framework中，CLR提供了一组COM接口，用于支持非托管应用程序寄宿CLR。这些接口包括ICorRuntimeHost、ICLRRuntimeHost和ICLRControl等。非托管应用程序可以通过调用这些接口来启动和停止CLR、获取应用程序域、执行托管代码等。

例如，非托管应用程序可以通过调用ICorRuntimeHost接口的Start方法来启动CLR，然后调用GetDefaultDomain方法来获取默认应用程序域。接下来，非托管应用程序可以使用默认应用程序域来加载和执行托管程序集。

CLRCreateInstance函数可返回一个ICLRMetaHost接口。宿主应用程序可调用这个接口的GetRuntime函数，指定数组要创建的CLR的版本。然后，垫片将所需版本的CLR加载到宿主的进程中。

默认情况下，当一个托管的可执行文件启动时，垫片会检查可执行文件，提取当初生成和测试应用程序时使用的CLR的版本信息。但应用程序可以在它的xml配置文件中设置requiredRuntime和supportedRuntime来覆盖默认行为

GetRuntime函数返回指定非托管ICLRRuntimeInfo接口的指针。有个这个指针后，就可以利用GetInterface方法获取ICLRRuntimeHost接口。宿主应用程序可调用该接口定义的方法做如下事情



AppDomain（应用程序域）

什么是AppDomain：

AppDomain是一个轻量级的进程，它为运行在其中的应用程序提供了一个隔离的环境。每个应用程序域都有自己的私有内存，可以独立于其他应用程序域加载和卸载程序集。AppDomain是为了提供隔离而设计的。

Ps：（提供隔离）AppDomain是为了隔离运行在同一进程中的多个应用程序而设计的。每个AppDomain都有自己的私有内存，可以独立于其他AppDomain加载和卸载程序集。这意味着在一个AppDomain中发生的错误不会影响其他AppDomain。

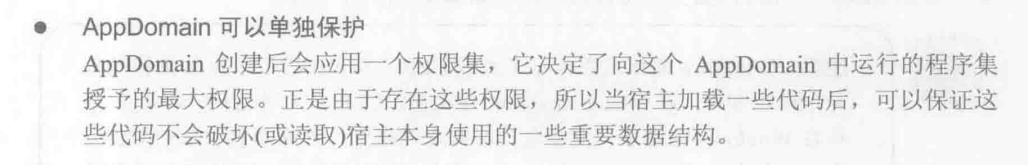
AppDomain的具体功能：

1.一个AppDomain的代码不能直接访问另一个应用程序域的代码。 AppDomain之间有清晰的分隔和边界，正因如此这种隔离使得应用程序域很从进程中卸载时不会影响其他AppDomain正在运行的代码。（隔离性）



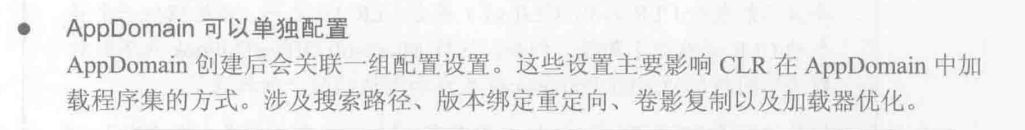
2.AppDomain可以卸载。CLR不支持从AppDomain中卸载特定的程序集，但可以卸载一个AppDomain从而卸载这个程序控制域当前包含的所有程序集。

3.AppDomain可以单独保护。AppDomain创建后会应用一个权限集，它决定了向这个AppDomain中运行的程序集授予的最大权限。（安全性？）



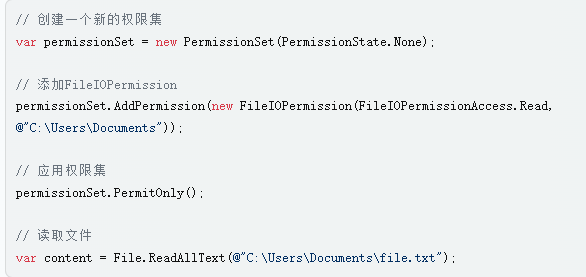
4.AppDomain可以单独配置。

AppDomain创建后会关联一组配置。这些设置主要影响CLR在AppDomain中加载程序集的方式。



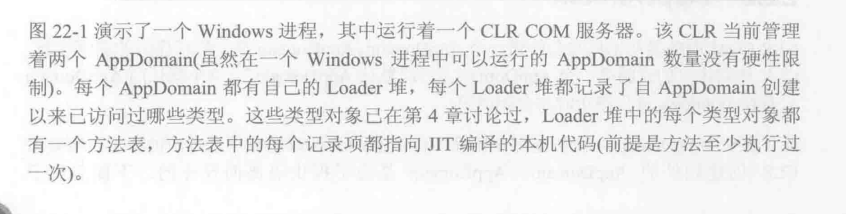
Ps：什么是权限集，权限集（Permission Set）是.NET Framework中的一个概念，它表示一组权限，用于控制代码能够执行的操作。每个权限都表示对系统资源的一种访问类型，例如读取文件、访问网络、调用非托管代码等。权限集可以包含多个权限，以便控制代码能够执行的操作。

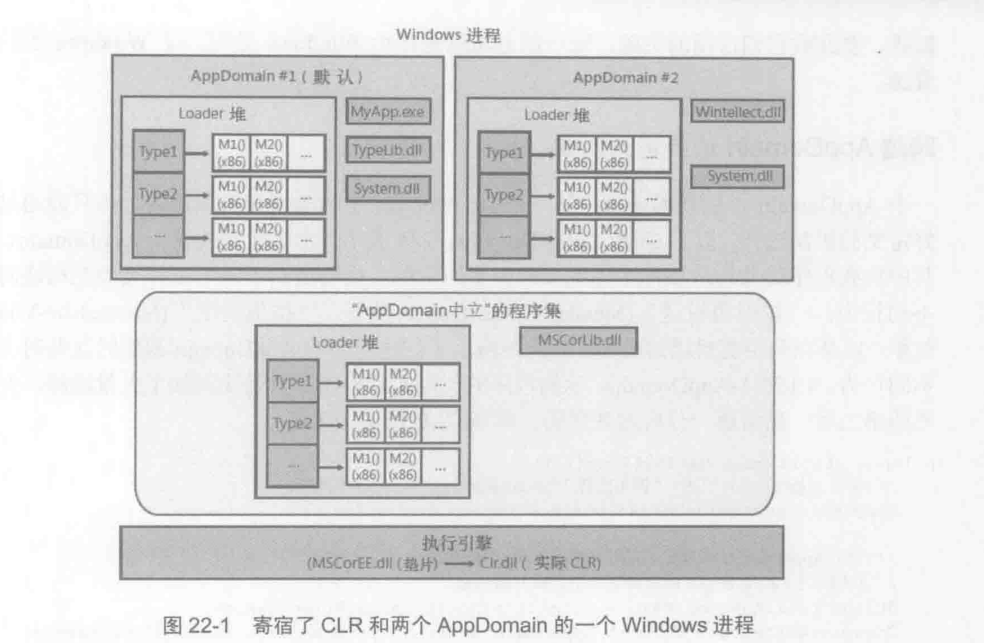
例子：创建一个权限集



读取文件的权限集，PermitOnly意味着只能在当前程序集使用

下面这个例子可以帮助理解AppDomain间的关系，P493：





图中的AppDomain中立的程序集是域中立程序集，是指可以在多个应用程序域之间共享的程序集。这意味着域中立程序集只会被加载一次，然后可以被多个AppDomain共享使用。域中立程序集通常用于包含公共类型和方法的程序集，例如.NET Framework类库。这些程序集通常不会被修改，因此可以在多个AppDomain之间共享，以减少内存占用和提高性能。且以AppDomain中立方式加载的程序集无法卸载，要回收资源只能终止进程让Windows回收。

两个AppDomain如果加载了同一个程序集，且都使用了同一个类型，那么这个在两个程序集的Loader堆的类型对象是完全一样的，但是两个AppDomain的类型对象不共享内存以及JIT编译产生的IL代码。这样不共享看起来会浪费内存，但是符合AppDomain的预期效果——隔离。

跨越AppDomain的边界访问对象：

按照AppDomain通信可以将类型分为三种：按引用封送类型，按值封送类型，完全不能封送的类型。

线程和AppDomain并不是一对一关系，而是多对多关系。但线程可以使用GetDomain方法获取正在哪个AppDomain中执行

什么是“友好名称”，AppDomain在被创建好后就被赋予一个友好名称，是一个用于标识应用程序域的String。可以使用 System.AppDomain命名空间的FriendlyName只读属性获取默认AppDomain的友好名称。

什么是“代理类型”：

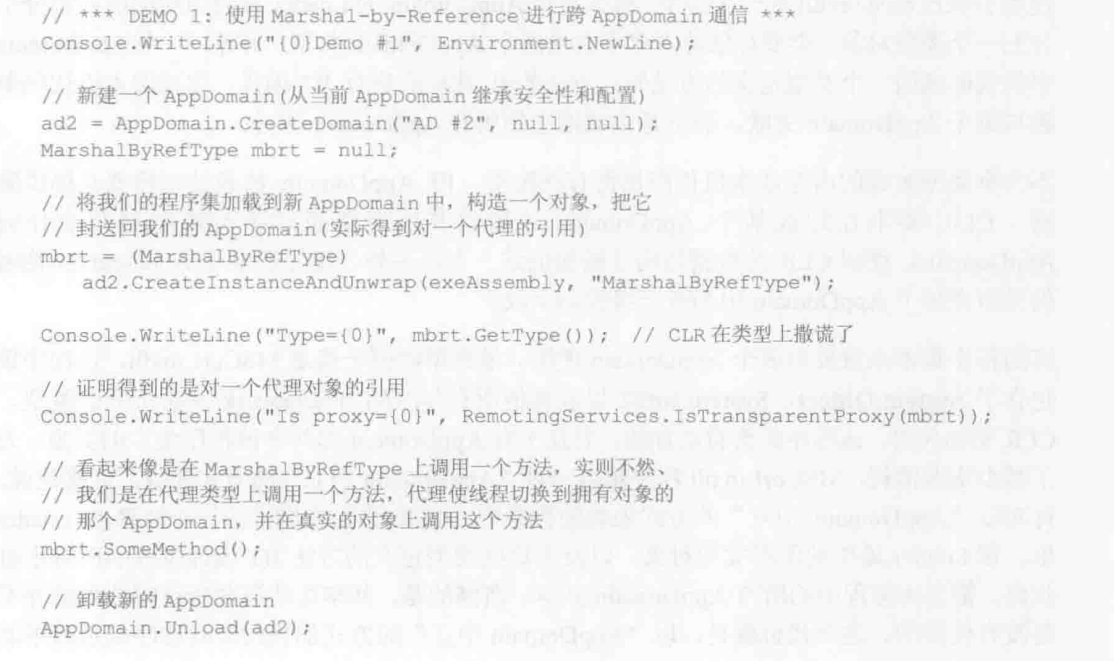
代理类型通常指在跨应用程序域访问对象时使用的代理对象的类型。当您需要在一个应用程序域中访问另一个应用程序域中的对象时，可以使用代理对象来实现。代理对象是一个特殊的对象，它位于当前应用程序域中，但是能够与目标应用程序域中的对象进行通信。

根据对象封送方式的不同，代理类型也有所不同。例如，当对象按引用封送时，会在当前应用程序域中创建一个透明代理对象。透明代理对象是一个特殊的代理类型，它能够将方法调用转发到目标应用程序域中的实际对象上。这样，您就可以在当前应用程序域中像操作本地对象一样操作透明代理对象，而实际上是在目标应用程序域中操作实际对象。

按引用封送：

实际上是AppDomain比如B传给另一个AppDomain比如A一个引用对象b，当执行b的函数时，实际上是线程又切回了B，调用完后再切回A。

当对象按引用封送时，会在目标应用程序域中创建一个代理对象，该代理对象与源应用程序域中的原始对象通信。当在目标应用程序域中调用代理对象的方法时，实际上是在源应用程序域中调用原始对象的方法。按引用封送通常用于跨应用程序域访问对象，而不需要复制整个对象。



图片包含 表格

描述已自动生成

CreateInstanceAndUnwrap为什么导致切换线程： 方法用于在指定的应用程序域中创建一个新的对象实例，并返回对该对象的引用。这个过程可能会导致线程切换，因为在创建新的对象实例时，需要在目标应用程序域中执行代码。这意味着，如果目标应用程序域正在由另一个线程执行，则可能会发生线程切换，以便在目标应用程序域中执行代码。

如果CreateInstanceAndUnwrap封送的对象类型派生自MarshalByRefObject就会跨应用程序域边界封送对象

文本

描述已自动生成

跨AppDomian边界的方法调用是同步执行的

手机截图图有时间和文字

描述已自动生成

报纸上的文字

描述已自动生成

文本

描述已自动生成

按值封送：

实际上是将一个AppDomain中的对象序列化然后放到另一个AppDomain进行反序列化中。

文本

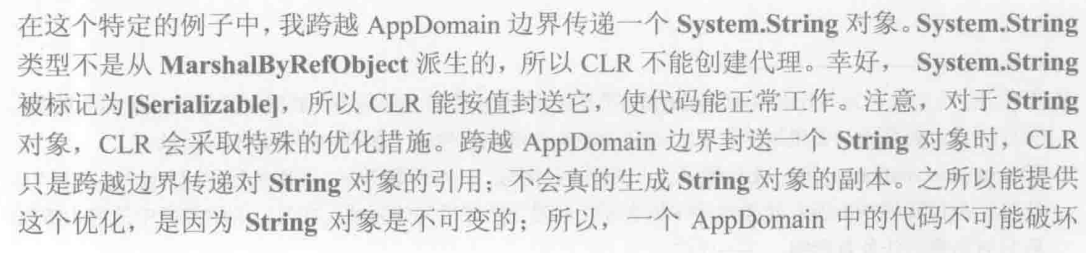
描述已自动生成

不可封送类型：

文本

描述已自动生成

传递的不是MarshalByRefObject的派生类对象



卸载AppDomain：

如何卸载AppDomain：

调用静态方法AppDomain.Unload

结果：

CLR卸载AppDomain中全部程序集，释放Loader堆

步骤：

1.CLR挂起进程中执行过托管代码的所有线程

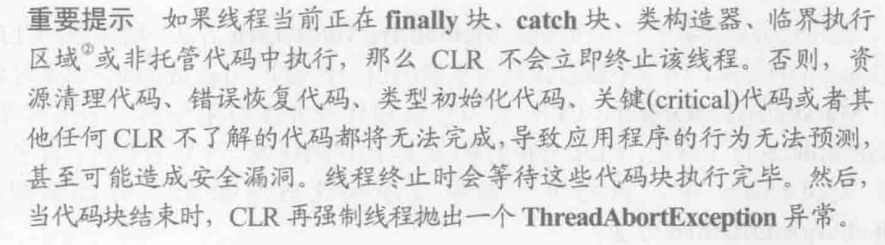
2.查看那些线程正在执行要卸载的AppDomain中的代码，有就强迫对应的线程抛出一个ThreadAbortException(线程中止异常)并恢复执行线程，这将导致执行遇到的所有的finally块以清理资源。如果没有代码捕捉到这个异常，那么finally就不会执行，但是线程会终止，但进程可继续运行。（其它所有未处理的异常我们在前面讲过会终止进程，而这里只会终止这个线程.

3.执行完第二步，所有的线程都会离开此AppDomain，CLR进入前面垃圾回收类似的标记阶段，标记垃圾。

4.CLR强制执行垃圾回收，这些对象的Finalize方法被调用来正确清理占用的资源。

5.CLR恢复所有线程的执行。

Ps：任何时候都只会有一个线程来调用Unload方法，不会有多个线程同时调用。



监视AppDomain

可以使用 AppDomain.MonitoringIsEnabled 属性来启用 AppDomain 监视。当此属性设置为 true 时，可以使用

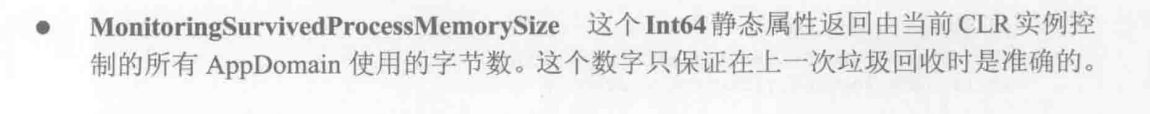
AppDomain.MonitoringTotalProcessorTime 属性

AppDomain.MonitoringTotalAllocatedMemorySize属性

AppDomain.MonitoringSurvivedMemorySize 属性

AppDomain.MonitoringSurvivedProcessMemorySize属性

来检索有关 AppDomain 的性能数据。



文本

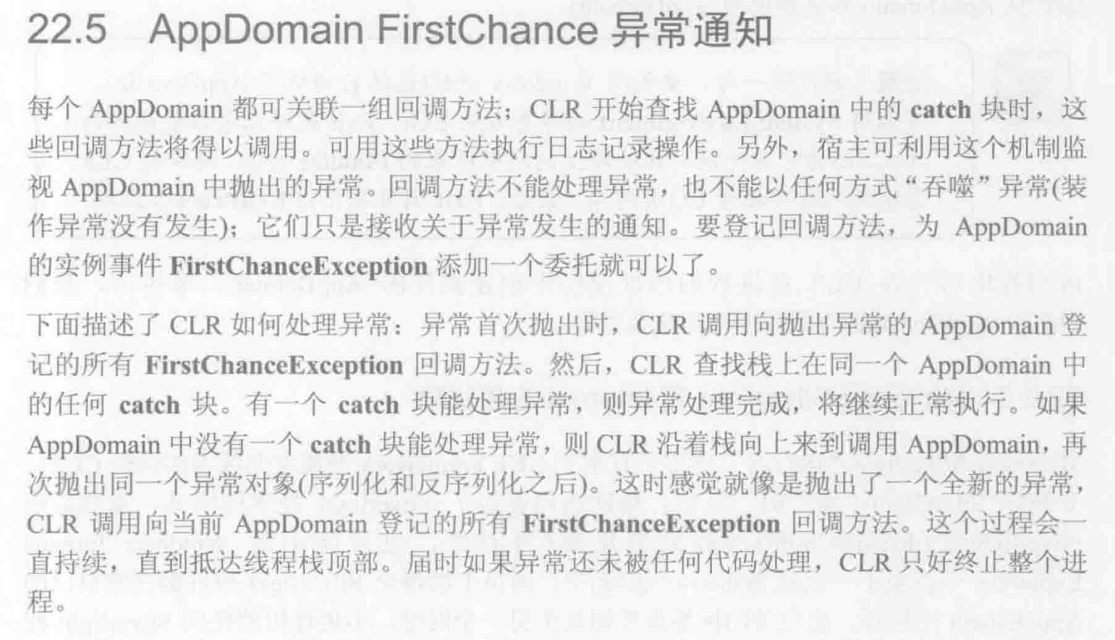
中度可信度描述已自动生成

下面是书上P506的使用这些属性的例子：

文本

描述已自动生成

没看懂：



宿主如何使用AppDomain：

控制台UI应用程序，NT Service应用程序，Windows窗体应用程序和WPF应用程序都是自寄宿（即自己容纳CLR）的应用程序。