AppDomain concept in ASP.NET

ASP.NETintroduces the concept of an Application Domain which is shortly known as AppDomain. It can be considered as a Lightweight process which is both a container and boundary. The .NET runtime uses an AppDomain as a container for code and data, just like the operating system uses a process as a container for code and data. As the operating system uses a process to isolate misbehaving code, the .NET runtime uses an AppDomain to isolate code inside of a secure boundary.  
  
The CLR can allow the multiple .Net applications to be run in a single AppDomain.  
  
The CLR isolates each application domain from all other application domains and prevents the configuration, security, or stability of a running .NET applications from affecting other applications.An AppDomain can be destroyed without effecting the other Appdomains in the process.  
  
Mulitple Appdomains can exist in Win32 process. As we discussed the main aim of AppDomain is to isolate applications from each other and the process is same as the working of operating system process. This isolation is achieved by making sure than any given unique virtual address space runs exactly one application and scopes the resources for the process or application domain using that address space.  
  
Win32 processes provide isolation by having distinct memory addresses. The .Net runtime enforces AppDomain isolation by keeping control over the use of memory. All memory in the App domain is managed by the run time so the runtime can ensure that AppDomains Do not access each others memory.

**How to create AppDomain**

AppDomains are generally created by Hosts for example Internet Explorer and Asp.net. The following is an example to create instance of an object inside it and then executes one of the objects methods. This is the explicit way of creating AppDomain by .Net Applications  
  
AppDomains are created using the CreateDomain method. AppDomain instances are used to load and execute assemblies (Assembly). When an AppDomain is no longer in use, it can be unloaded.  
  
public class MyAppDomain : MarshalByRefObject  
{  
public string GetInfo()  
{  
return AppDomain.CurrentDomain.FriendlyName;  
}  
}

public class MyApp

{

public static void Main()

{

AppDomain apd = AppDomain.CreateDomain("Rajendrs Domain");

MyAppDomain apdinfo = (MyAppDomain)apd.CreateInstanceAndUnwrap (Assembly.GetCallingAssembly().GetName().Name, "MyAppDomain");

Console.WriteLine("Application Name = " + apdinfo.GetInfo());

}

}

The AppDomain class implements a set of events that enable applications to respond when an assembly is loaded, when an application domain will be unloaded, or when an unhandled exception is thrown.   
  
**Advantages**

A single CLR operating system process can contain multiple application domains. There are advantages to having application domains within a single process.

1. Lower system cost - many application domains can be contained within a single system process.
2. Each application domain can have different security access levels assigned to them, all within a single process.
3. Code in one AppDomain cannot directly access code in another AppDomain.
4. The application in an AppDomain can be stopped without affecting the state of another AppDomain running in the same process.
5. An Exception in on AppDomain will not affect other AppDomains or crash the entire process that hosts the AppDomains.

**Summary**

This article explained about the Appdomain concept in Asp.net.