```
namespace Practices.Services.AppDomainManagers
{
   public class AppDomainManager:ServiceBase
   {
      #region Private Members
      private readonly Dictionary<string,AppDomain> m_AppDomains = new Dictionary<string, AppDomain>();
      private readonly Dictionary<string, AppDomainEventNotify> m_AppDomainEventNotifys = new
Dictionary<string, AppDomainEventNotify>();
      private readonly Dictionary<string, AppDomainEventListener> m_AppDomainEventListeners = new
Dictionary<string, AppDomainEventListener>();
      #endregion

      #region Public Members
      public AppDomain CreateAppDomain(string appDomainName, string baseDirectory)
      {
         if (!m_AppDomains.ContainsKey(appDomainName))
         {
            AppDomainSetup appDomainSetup = new AppDomainSetup
                                    {
                                       ApplicationName = appDomainName,
                                       ApplicationBase = baseDirectory,
                                       ShadowCopyFiles = "true",
                                       ShadowCopyDirectories = baseDirectory,
                                       ConfigurationFile =
                                          AppDomain.CurrentDomain.SetupInformation.ConfigurationFile
                                    };
            AppDomain workerDomain = AppDomain.CreateDomain(appDomainName,null,appDomainSetup);

            const string typeString =
"Practices.Service.AppDomainManager.AppDomainEventNotify,Practices.Service.AppDomainManager";
            AppDomainEventNotify appDomainEventNotify =
               TypeResolver.CreateInstance("", typeString, workerDomain, null) as AppDomainEventNotify;

            AppDomainEventListener appDomainEventListener = new AppDomainEventListener(this,
appDomainEventNotify);

            m_AppDomains.Add(appDomainName, workerDomain);
            m_AppDomainEventNotifys.Add(appDomainName, appDomainEventNotify);
            m_AppDomainEventListeners.Add(appDomainName, appDomainEventListener);
         }
         return m_AppDomains[appDomainName];
      }
```

```
        public void UnloadAppDomain(string appDomainName)
        {
            if (m_AppDomains.ContainsKey(appDomainName))
            {
                AppDomain.Unload(m_AppDomains[appDomainName]);
                m_AppDomains.Remove(appDomainName);
                m_AppDomainEventNotifys.Remove(appDomainName);
                m_AppDomainEventListeners.Remove(appDomainName);
            }
        }
        #endregion

        #region Internal Methods
        internal void OnDomainUnload(object sender, EventArgs e)
        {
            AppDomain appDomain = sender as AppDomain;
            if (appDomain != null)
                Logger.WriteLog("Test1", "DomainUnload in " + appDomain.Id + ", " + appDomain.FriendlyName);
        }

        internal void OnAssemblyLoad(object sender, AssemblyLoadEventArgs args)
        {
            AppDomain appDomain = sender as AppDomain;
            if (appDomain != null)
                Logger.WriteLog("Test1", "AssemblyLoad in " + appDomain.Id + ", " + appDomain.FriendlyName);
        }

        internal void OnUnhandledException(object sender, UnhandledExceptionEventArgs e)
        {
            AppDomain appDomain = sender as AppDomain;
            if (appDomain != null)
                Logger.WriteLog("Test1", "UnhandledException in " + appDomain.Id + ", " +
appDomain.FriendlyName);
        }

        #endregion
    }
}
```

Helper:

```
namespace Practices.Services.AppDomainManagers
{
    public class AppDomainEventListener:ServiceBase
    {
        #region Private Members
        private readonly AppDomainManager m_AppDomainManager;
        private readonly AppDomainEventNotify m_AppDomainEventNotify;
        #endregion

        #region Public Constructor
        public AppDomainEventListener(AppDomainManager appDomainManager,AppDomainEventNotify
appDomainEventNotify)
        {
            m_AppDomainManager = appDomainManager;
            m_AppDomainEventNotify = appDomainEventNotify;
            m_AppDomainEventNotify.RegisterEventListener(this);
        }
        #endregion

        #region Public Methods
        public void OnDomainUnload(object sender, EventArgs e)
        {
            m_AppDomainManager.OnDomainUnload(sender,e);
        }

        public void OnAssemblyLoad(object sender, AssemblyLoadEventArgs e)
        {
            m_AppDomainManager.OnAssemblyLoad(sender, e);
        }

        public void OnUnhandledException(object sender, UnhandledExceptionEventArgs e)
        {
            m_AppDomainManager.OnUnhandledException(sender, e);
        }
        #endregion
    }
}

namespace Practices.Services.AppDomainManagers
{
    public class AppDomainEventNotify:ServiceBase
    {
        #region Private Members
        private readonly List<AppDomainEventListener> m_AppDomainEventListeners = new
List<AppDomainEventListener>();
        #endregion
        #region Public Constructor
        public AppDomainEventNotify()
        {
            AppDomain.CurrentDomain.DomainUnload += CurrentDomain_DomainUnload;
            AppDomain.CurrentDomain.AssemblyLoad += CurrentDomain_AssemblyLoad;
            AppDomain.CurrentDomain.UnhandledException += CurrentDomain_UnhandledException;
        }
        #endregion

        #region Public Method
        public void RegisterEventListener(AppDomainEventListener appDomainEventListener)
        {
            m_AppDomainEventListeners.Add(appDomainEventListener);
        }
        #endregion

        #region Private Events
        private void CurrentDomain_DomainUnload(object sender, EventArgs e)
        {
            foreach (AppDomainEventListener appDomainEventListener in m_AppDomainEventListeners)
            {
                appDomainEventListener.OnDomainUnload(sender,e);
            }
        }

        private void CurrentDomain_AssemblyLoad(object sender, AssemblyLoadEventArgs e)
        {
            foreach (AppDomainEventListener appDomainEventListener in m_AppDomainEventListeners)
            {
                appDomainEventListener.OnAssemblyLoad(sender, e);
            }
        }

        private void CurrentDomain_UnhandledException(object sender, UnhandledExceptionEventArgs e)
        {
            foreach (AppDomainEventListener appDomainEventListener in m_AppDomainEventListeners)
            {
                appDomainEventListener.OnUnhandledException(sender, e);
            }
        }

        #endregion
    }
```

```
}
```