

```

using System.Runtime.InteropServices;
using System;
using System.Security;
using System.Security.Permissions;
using System.Collections;
using System.IO;
using System.Text;
using System.Windows.Forms;

namespace sb54_CSAI.MiscFormComponents
{
    #region EnableThemingInScope CLASS
    /// <summary>
    /// This class is intended to use with the C# 'using' statement
    /// to activate an activation context for turning on visual theming at
    /// the beginning of a scope, and have it automatically deactivated
    /// when the scope is exited.
    /// <br></br>
    /// <br></br>
    /// This class was taken directly from the following web site
    /// http://support.microsoft.com/default.aspx?scid=kb;en-us;830033
    /// This article was written by microsoft to provide correct Theming
    /// to coneract the "InteropServices.SEHException: External component
    /// has thrown an exception", that was raised using the
    /// Application.EnableVisualStyles() method call.
    /// </summary>
    [ SuppressUnmanagedCodeSecurity ]
    public class EnableThemingInScope : IDisposable
    {
        #region Instance fields
        //Instance fields
        private uint cookie;
        private static ACTCTX enableThemingActivationContext;
        private static IntPtr hActCtx;
        private static bool contextCreationSucceeded = false;
        #endregion
        #region Public Methods
        /// <summary>
        ///Windows XP makes use of 2 types of common control comctl32.dll, however
        ///only v6 provides theming facilities using the UxTheme.Dll.
        ///This class attempts to correctly switch the Windows XP
        ///common controls from v5.8 (Non themed to v6.0 themed)
        ///So that comctl32 function calls can be properly redirected to comctl32 v6.0.
        /// </summary>
        /// <param name="enable">True to enable Visual Theming</param>
        public EnableThemingInScope(bool enable)
        {
            cookie = 0;
            //If theming is available for this operating system
            if (enable && OSFeature.Feature.IsPresent(OSFeature.Themes))
            {
                //The content must be created 1st
                if (EnsureActivateContextCreated())
                {
                    if (!ActivateActCtx(hActCtx, out cookie))
                    {
                        // Be sure cookie always zero if activation failed
                        cookie = 0;
                    }
                }
            }
        }

        /// <summary>
        /// Deconstrcutor, simply disposes of the EnableThemingInScope object
        /// </summary>
        ~EnableThemingInScope()
        {
            Dispose(false);
        }

        /// <summary>
        /// Provides a Dispose method, to allow the EnableThemingInScope object to be

```

```

disposed correctly
    /// </summary>
    void IDisposable.Dispose()
    {
        Dispose(true);
    }
#endregion
#region Private Methods
    /// <summary>
    /// Provides a mechanism to clean up any held resources
    /// </summary>
    /// <param name="disposing">True if the held resources should be de-allocated</
param>
    private void Dispose(bool disposing)
    {
        if (cookie != 0)
        {
            if (DeactivateActCtx(0, cookie))
            {
                // deactivation succeeded...
                cookie = 0;
            }
        }
    }

    /// <summary>
    /// Attempts to provide active content to the controls that need to be
    /// Themed
    /// </summary>
    /// <returns>true if the active content could be created</returns>
    private bool EnsureActivateContextCreated()
    {
        //lock on this object type, mark this section as a critical section
        lock (typeof(EnableThemingInScope))
        {
            if (!contextCreationSucceeded)
            {
                // Pull manifest from the .NET Framework install
                // directory
                string assemblyLoc = null;

                FileIOPermission fiop = new FileIOPermission(PermissionState.None);
                fiop.AllFiles = FileIOPermissionAccess.PathDiscovery;
                fiop.Assert();
                try
                {
                    assemblyLoc = typeof(Object).Assembly.Location;
                }
                finally
                {
                    CodeAccessPermission.RevertAssert();
                }

                string manifestLoc = null;
                string installDir = null;
                if (assemblyLoc != null)
                {
                    installDir = Path.GetDirectoryName(assemblyLoc);
                    const string manifestName = "XPThemes.manifest";
                    manifestLoc = Path.Combine(installDir, manifestName);
                }

                if (manifestLoc != null && installDir != null)
                {
                    enableThemingActivationContext = new ACTCTX();
                    enableThemingActivationContext.cbSize = Marshal.SizeOf(typeof
(ACTCTX));
                    enableThemingActivationContext.lpSource = manifestLoc;

                    // Set the lpAssemblyDirectory to the install
                    // directory to prevent Win32 Side by Side from
                    // looking for comctl32 in the application
                    // directory, which could cause a bogus dll to be

```

```

        // placed there and open a security hole.
        enableThemingActivationContext.lpAssemblyDirectory = installDir;
        enableThemingActivationContext.dwFlags =
ACTCTX_FLAG_ASSEMBLY_DIRECTORY_VALID;

        // Note this will fail gracefully if file specified
        // by manifestLoc doesn't exist.
        hActCtx = CreateActCtx(ref enableThemingActivationContext);
        contextCreationSucceeded = (hActCtx != new IntPtr(-1));
    }
}

// If we return false, we'll try again on the next call into
// EnsureActivateContextCreated(), which is fine.
return contextCreationSucceeded;
}
}

//Declare all the Dll Imports, these are C++ Dll's
[DllImport("Kernel32.dll")]
private extern static IntPtr CreateActCtx(ref ACTCTX actctx);
[DllImport("Kernel32.dll")]
private extern static bool ActivateActCtx(IntPtr hActCtx, out uint lpCookie);
[DllImport("Kernel32.dll")]
private extern static bool DeactivateActCtx(uint dwFlags, uint lpCookie);

private const int ACTCTX_FLAG_ASSEMBLY_DIRECTORY_VALID = 0x004;

//Create a structure to hold Active Content
private struct ACTCTX
{
    public int        cbSize;
    public uint       dwFlags;
    public string     lpSource;
    public ushort     wProcessorArchitecture;
    public ushort     wLangId;
    public string     lpAssemblyDirectory;
    public string     lpResourceName;
    public string     lpApplicationName;
}
#endregion
}
#endregion
}

```