ENS IV – Tasca 05

Autor: Sergio Blanco Cuaresma

Enunciat

1. Se trata de convertir el controller/model del Chat en un shared assembly registrado en la GAC, de forma que la vista lo utilice como referencia (GAC).

2. Mostrar un ejemplo de conexión y utilización desde C# de un componente COM localizado en el registro de windows. El Explorer os puede servir.

3. Comentad las diferencias, ventajas y desventajas del modelo de assemblies y GAC frente a componentes COM y el registro de Windows. ¿Qué pretenden solucionar los assemblies que no incorpora ya el modelo COM?

GAC

El proyecto consta de 3 elementos que pueden ser introducidos en el GAC:

- Módulo controlador
- Módulo modelo
- Módulo remoting

Para esto necesitaremos crear 3 claves criptográficas asimétricas que corresponderán a los Strong Names de cada uno de los assemblies anteriores:

```
cd “C:\Documents and Settings\Marble\Mis documentos\Visual Studio Projects\Tasca05\”
md keys
cd keys
"c:\Archivos de programa\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin\sn.exe" -k
ControllerKey.snk
"c:\Archivos de programa\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin\sn.exe" -k
ModelKey.snk
"c:\Archivos de programa\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin\sn.exe" -k
RemotingKey.snk
```

A continuación ligamos cada archivo con el módulo determinado cambiando el fichero AssemblyInfo.cs de cada módulo:

[assembly: AssemblyKeyFile("../../../keys/ControllerKey.snk")]

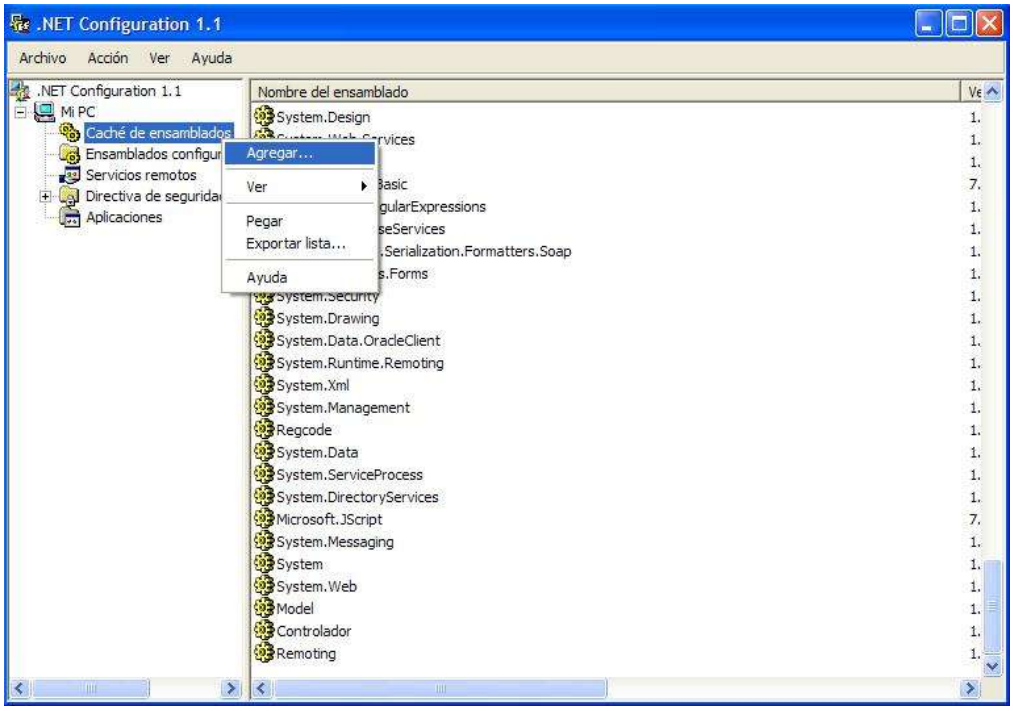
[assembly: AssemblyKeyFile("../../../keys/ModelKey.snk")]

[assembly: AssemblyKeyFile("../../../keys/RemotingKey.snk")]

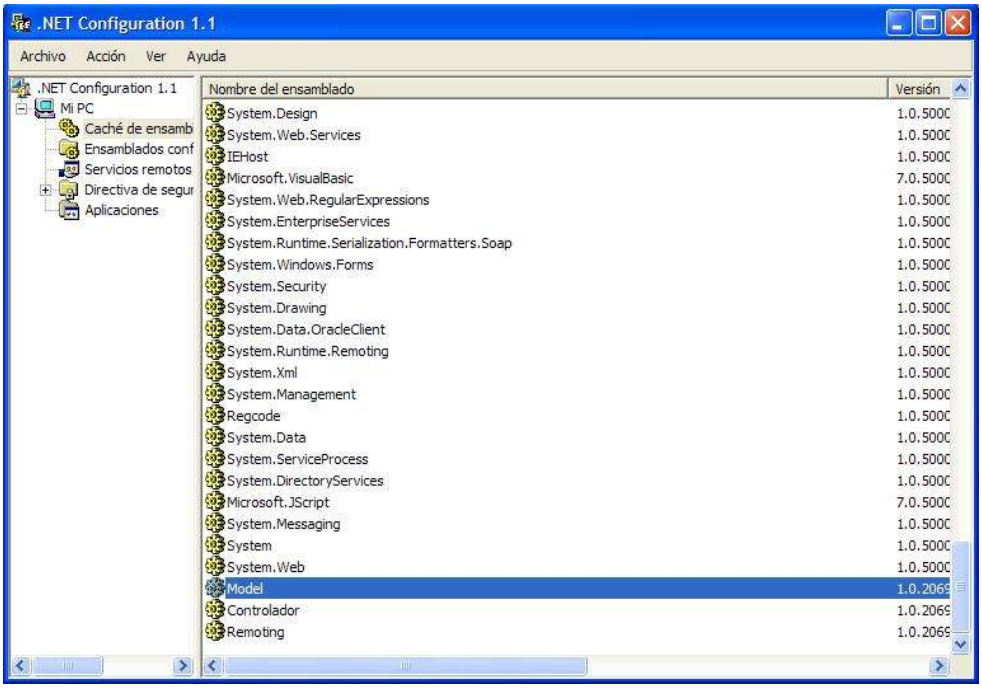
A continuación generamos los proyectos y copiamos todos los asemblies “Model”, “Controlador” y “Remoting” en un único directorio. Acto seguido debemos instalar en el GAC estos assemblies, es possible hacerlo mediante consola con:

```
"c:\Archivos de programa\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin\gacutil.exe" -I
Controlador.dll
```

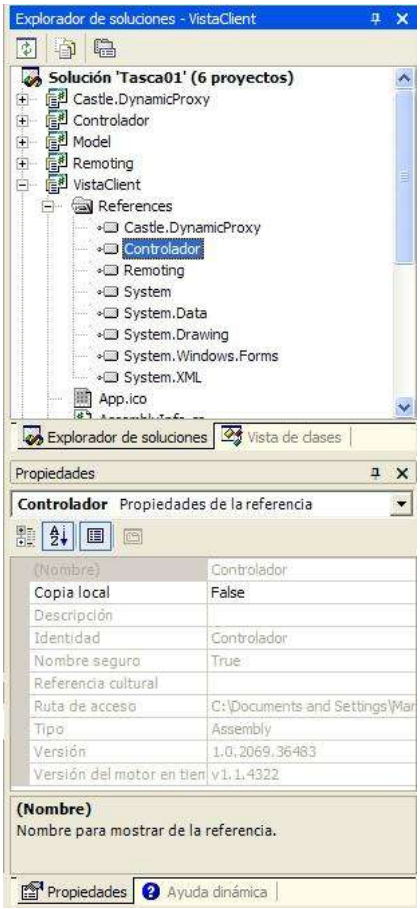
O mediante la interficie gráfica “Panel de Control -> Herramientas de administración -> .NET Configuration”:



Una vez insertados, en esta misma interfície gráfica es posible comprobarlo:



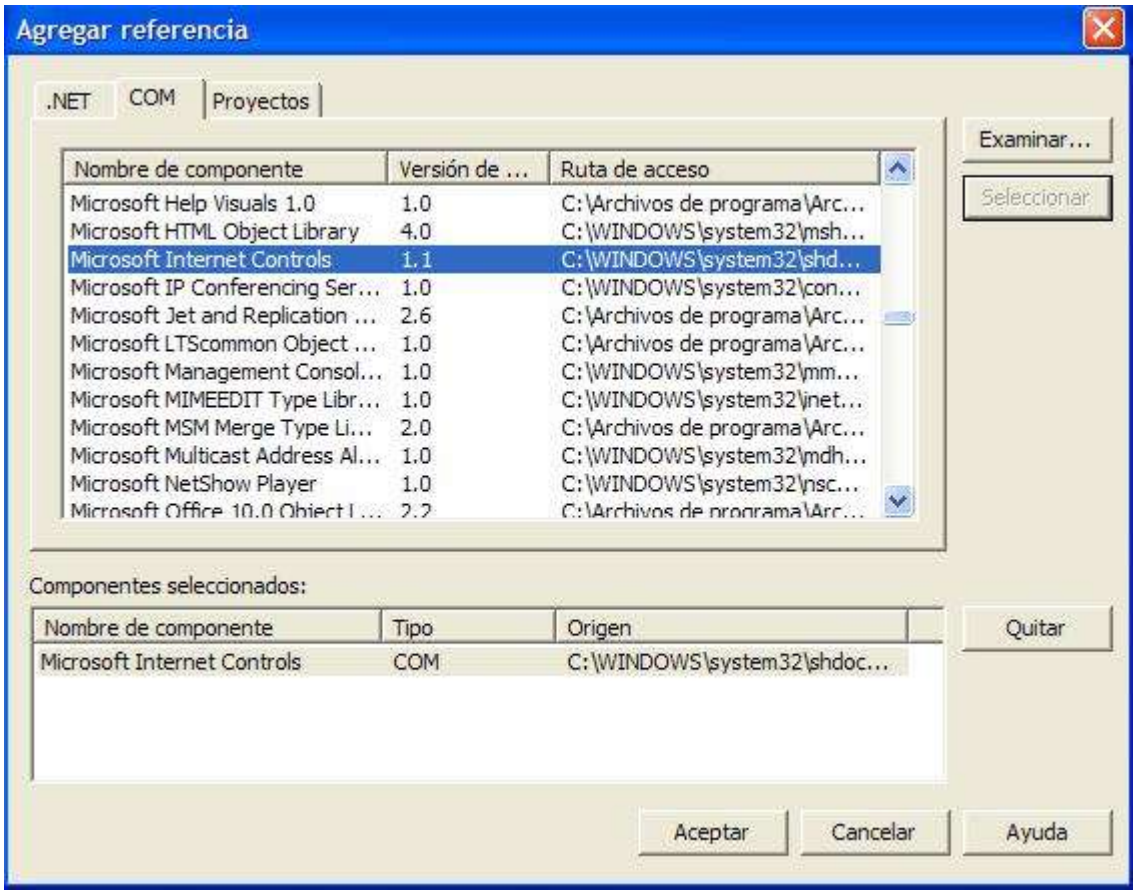
Finalmente, en los proyectos que utilizan estos assemblies (VisualClient y VisualServidor), debemos añadir referencias a los assemblies que habíamos agrupado en un único directorio (Visual Studio .NET 2003 no lista los assemblies del GAC, debemos insertarlos manualmente) y cambiar las propiedades para que no copie el assembly requerido al directorio donde se genera el ejecutable:



A partir de ahora no sera necesario tener los assemblies en el mismo directorio que el ejecutable VisualServer o VisualClient dado que estos seran localizados en el GAC.

COM

Para utilizar un componente COM desde .NET se ha añadido dicha referencia a la interfície gráfica VisualClient:



Y se han creado dos botones nuevos “Obrir explorer” y “Tancar explorer” que se encargan de abrir/cerrar una ventana de Internet Explorer:

```
using SHDocVw;
using System.Runtime.InteropServices;

// ...
private InternetExplorer explorer = null;
// ...

private void buttonObrir_Click(object sender, System.EventArgs e)
{
                if (explorer == null)
```

```
        {
                explorer = new InternetExplorer();
                buttonTancar.Enabled = true;
                buttonObrir.Enabled = false;
        }
}

private void buttonTancar_Click(object sender, System.EventArgs e)
{
        if (explorer == null)
        {
                explorer.Quit();
                explorer = null;
                buttonTancar.Enabled = false;
                buttonObrir.Enabled = true;
        }
}
```

GAC + Assemblies VS. Windows Register + COM

Podriamos decir que un assembly es una agrupación lógica de uno o más módulos (adicionalmente también pueden contener recursos como imágenes) englogabados bajo un nombre común. Un assembly puede ser privado (solo es utilizado por una aplicación concreta) o compartido (utilizado por diversas aplicaciones). Los assemblies compartidos se registran en el GAC (Global Assembler Cache).

En el GAC cada assembly es identificado por su tipo, version, cultura y una clave pública (que debe ser generada manualmente). De esta forma es posible tener diversas versiones de un mismo assembly sin entrar en conflicto, cada una de las aplicaciones que utilizan dicho assembly iran a buscar la versión que necesitan.

Si comparamos el model de assemblies + GAC con el seguido hasta la fecha por el sistema Microsoft Windows (Registro + DLLs), vemos que por fin se deja atrás el conocido “infierno de las DLLs” donde nuevas bibliotecas sobreescribian las viejas, haciendo asi que ciertas aplicaciones dejasen de funcionar correctamente.

En cuanto al desarrollo de componentes COM y al trabajo con assemblies, la cantidad de tiempo destinado a la programación se ve drasticamente mejorado con estos últimos. Con los assemblies ya no es necesarrio hacer pasos especiales para que pueda ser utilizado desde diversos lenguajes o diferentes aplicaciones.