Язык C#

Сборки, потоки и домены приложений

Лекция #5

Common Object Model

- Повторное использование кода
- Создание COM-серверов и регистрация их в реестре (HREY_CLASSES_ROOT)
- Проблема версий COM
- Запись в реестре и сам двоичный файл COM не связаны друг с другом

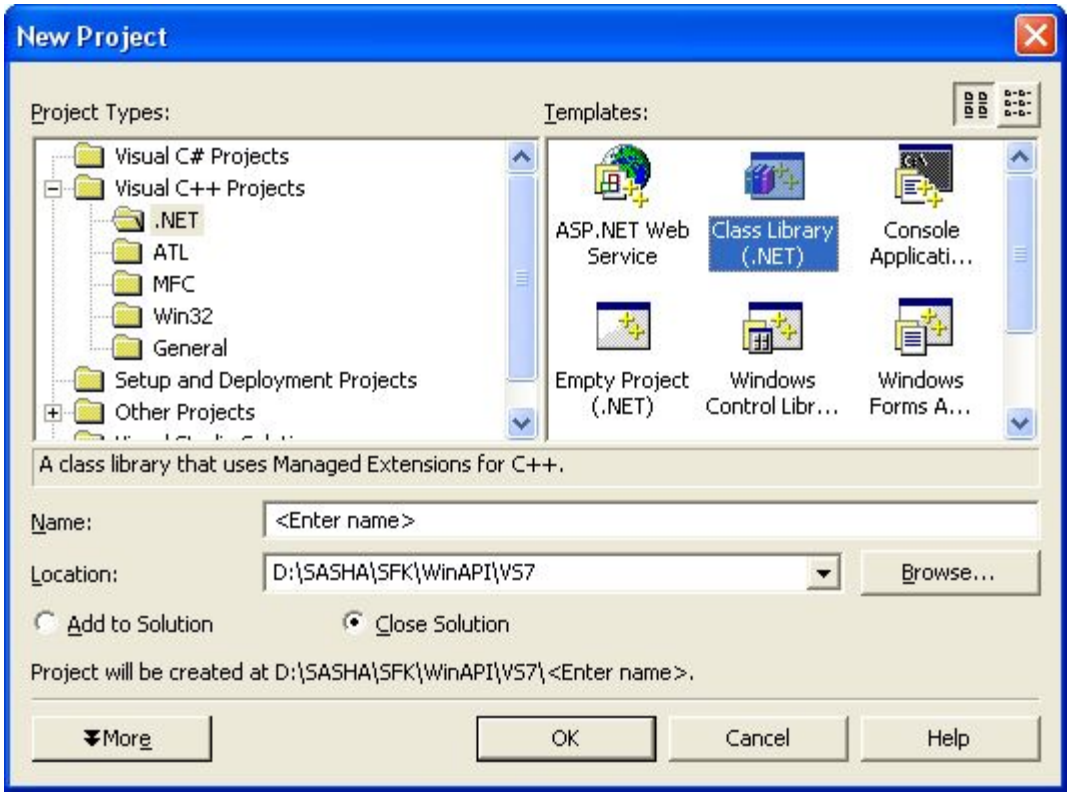
Обзор сборок .NET

- Приложение .NET – объединение любого количества сборок
- Сборка – это двоичный файл (DLL или EXE), который содержит номер версии, метаданные, а также типы и дополнительные ресурсы
- Манифест – набор метаданных о самой сборке

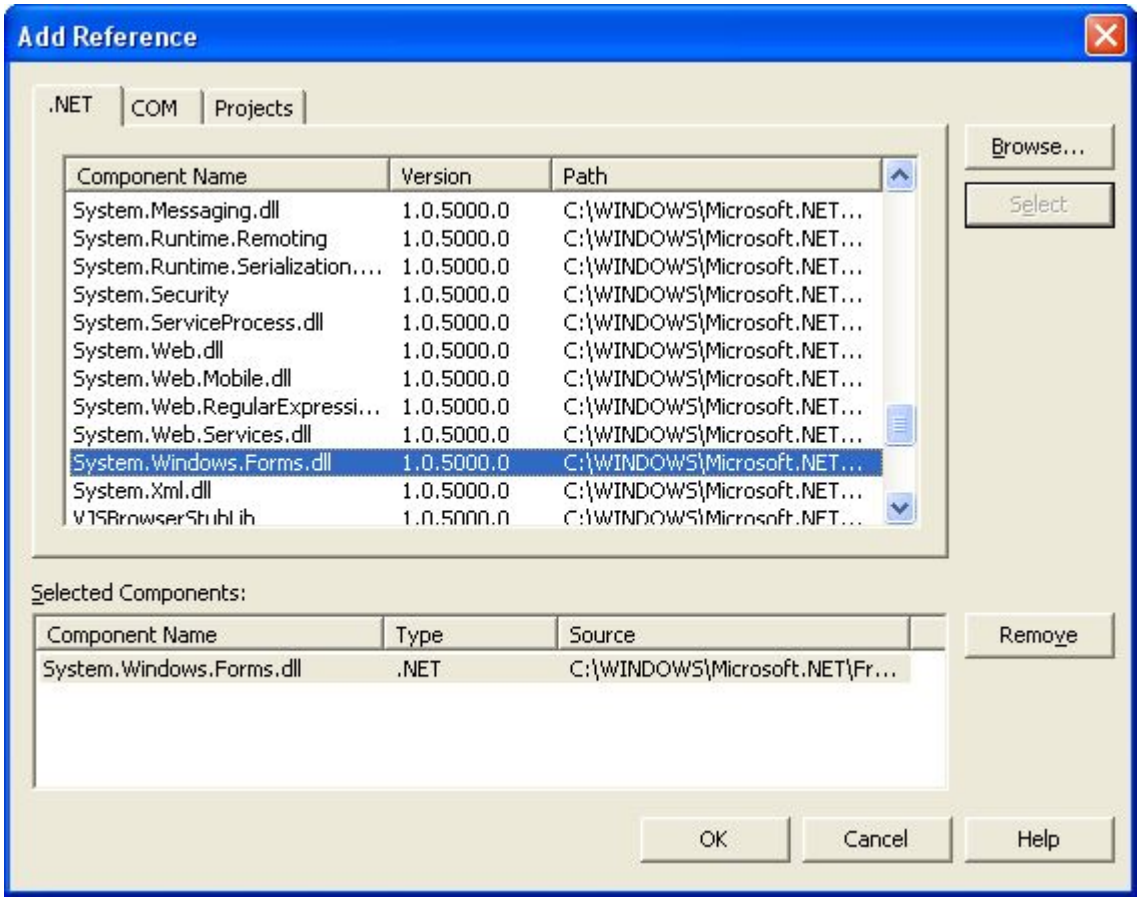
Обзор сборок .NET

- Логическое представление сборки (классы, интерфейсы, ресурсы, делегаты)
- Физическое представление (набор модулей – файлов)
- Сборки обеспечивают повторное использование кода, написанного на любом языке .NET
- Сборки – контейнеры для типов (не будет конфликта имен)
- Разные версии сборок могут выполнятся одновременно

Создание однофайловой сборки



Добавление ссылки на внешнюю сборку



Библиотека кода - CarLibary

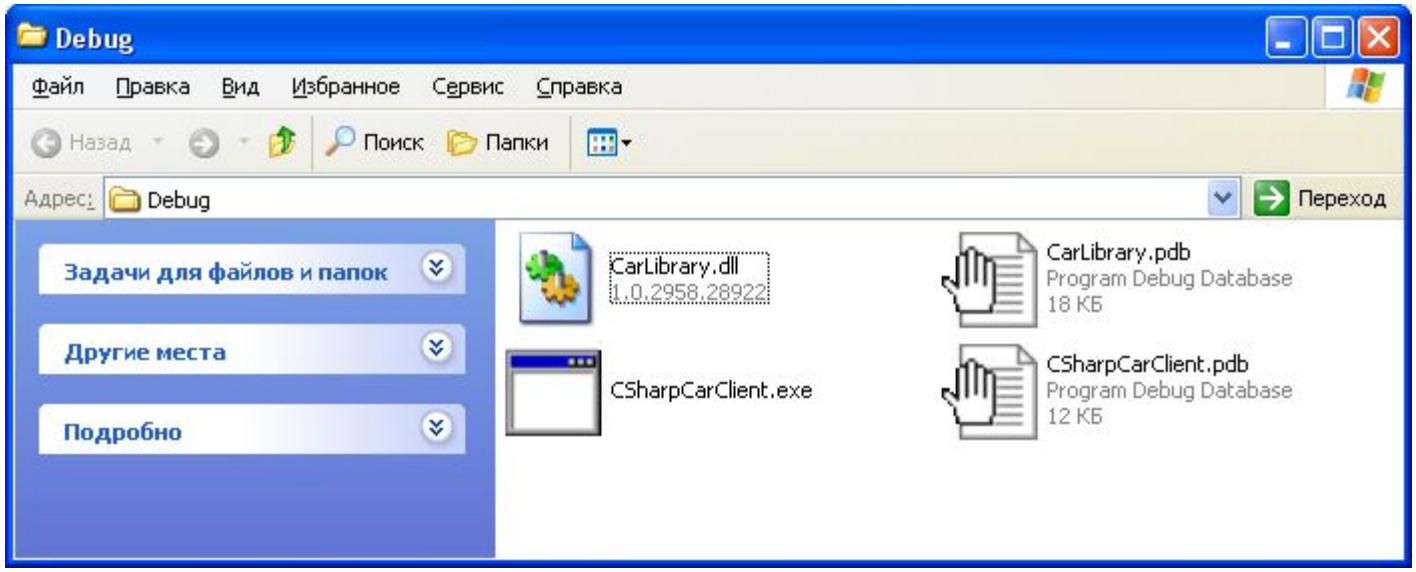
```
namespace CarLibrary
{   using System;

public enum EngineState          // Для двух возможных состояний двигателя
{       engineAlive,
        engineDead
}
public abstract class Car     // Абстрактный класс — базовый в нашей будущей иерархии
{
        // Защищенные данные о состоянии
        protected string petName;
        protected short currSpeed;
        protected short maxSpeed;
        protected EngineState egnState;

        public Car() {egnState = EngineState.engineAlive;}
        public Car(string name, short max, short curr)
        {       egnState = EngineState.EngineAlive;
                petName = name; maxSpeed = max; currSpeed = curr;  }

        public string PetName  {  get { return petName; }    set { petName = value; }  }

        public short CurrSpeed   {  get { return currSpeed; }     set { currSpeed = value; }    }

        public short MaxSpeed       { get {return maxSpeed; } }

        public EngineState EngineState     { get { return egnState; } }

        public abstract void TurboBoost();
}}
```

Реализация конкретных классов SportsCar и MiniVan

```
namespace CarLibrary
{ using System;
  using System.Windows.Forms; // Чтобы можно было использовать MessageBox

public class SportsCar : Car // Определение класса SportsCar
{
        // Конструкторы
        public SportsCar(){}
        public SportsCar(string name, short max, short curr) : base (name, max, curr) {}

        // Специфическая реализация метода TurboBoost()
        public override void TurboBoost()
        {
                MessageBox.Show("Ramming speed!", "Faster is better...");
        }
}

// Определение класса MiniVan
public class Minivan : Car
{
        // Конструкторы
        public MiniVan(){}
        public MiniVan(string name, short max, short curr) : base (name, max, curr){}

        // Реализация метода TurboBoost()
        {
                // Мини-вэны разгоняются неважно
                egnState = EngineState.engineDead;
                MessageBox.Show("Time to call AAA", "Your car is dead");
        }
}
}
```

Клиентское приложение

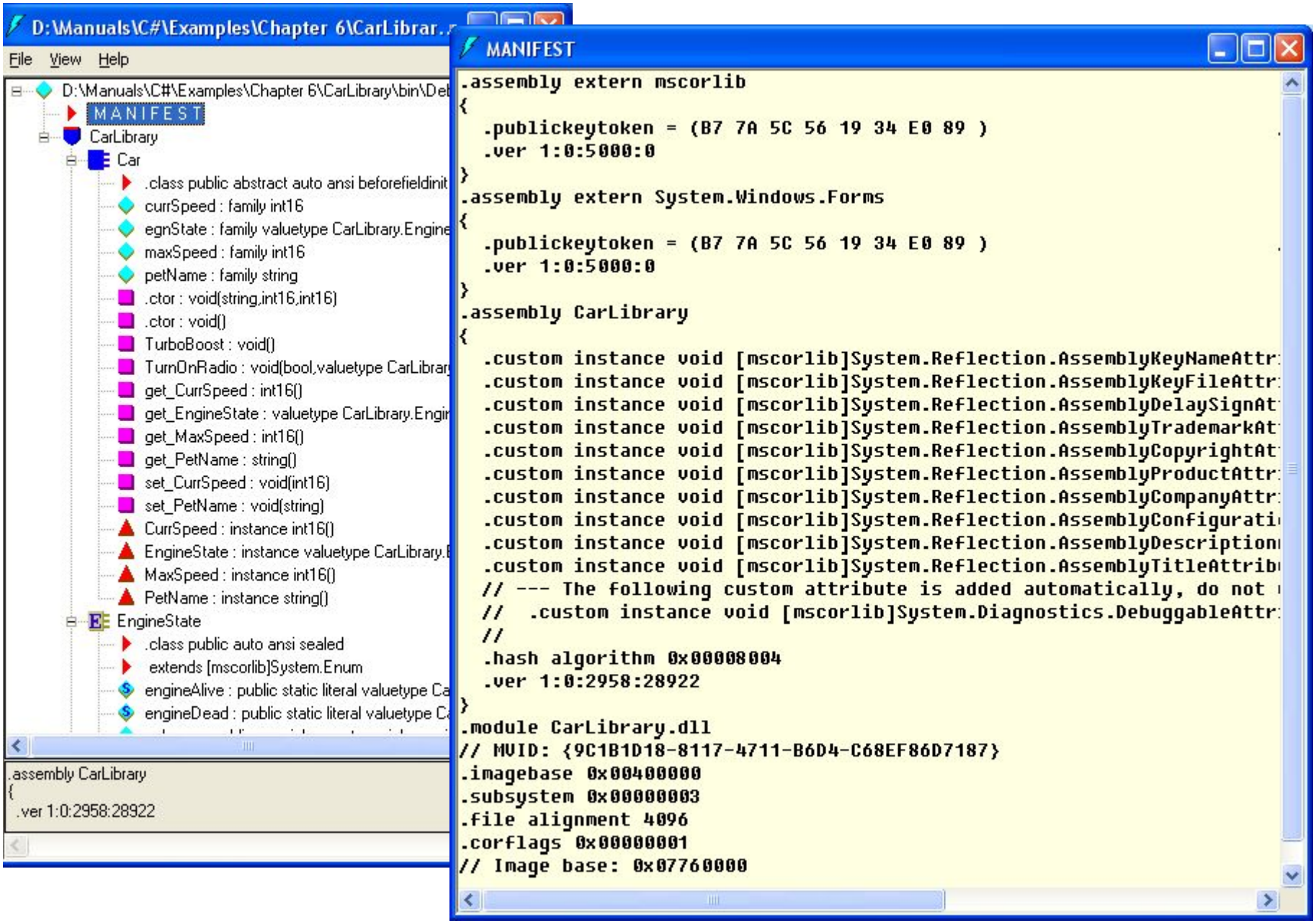
При добавлении в проекте ссылки на сборку
в каталог DEBUG полностью копируется DLL-файл



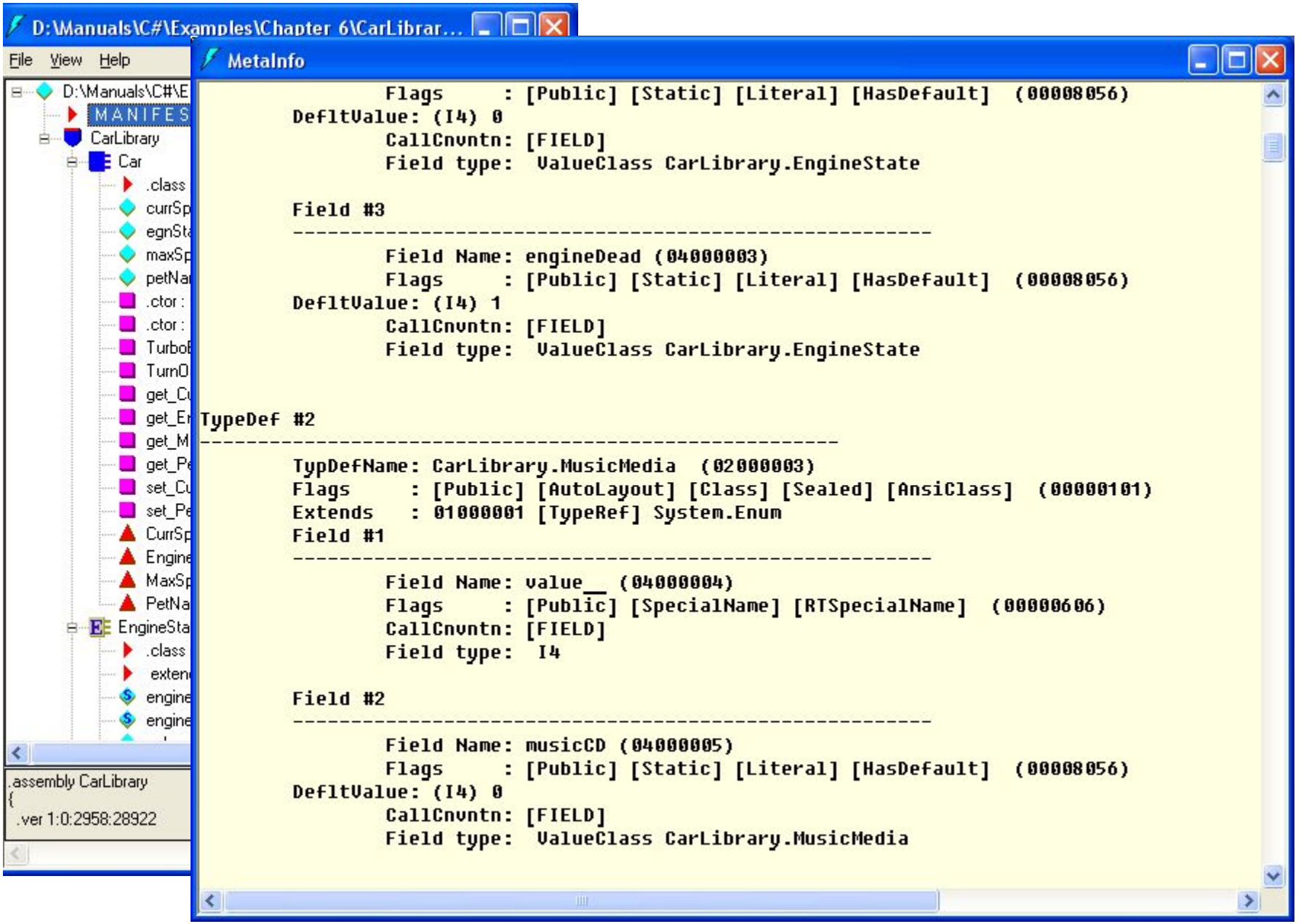
Клиентское приложение

```
// Первый опыт использования собственной библиотеки кода
namespace CSharpCarClient
{
using System;

// Используем типы из CarLibrary
using CarLibrary;

public class CarClient
{
	public static int Main(string[] args)
	{
		// Создаем автомобиль спортивной модели
		SportsCar viper = new SportsCar("Viper", 240, 40);
		viper.TurboBoost();

		// Создаем мини-вэн
		MiniVan mv = new MiniVan();
		mv.TurboBoost();

		return 0;
	}
}
}
```

Манифест


D:\Manuals\C#\Examples\Chapter 6\CarLibrar..
File View Help
D:\Manuals\C#\Examples\Chapter 6\CarLibrary\bin\Del
MANIFEST
CarLibrary
Car
.class public abstract auto ansi beforefieldinit
currSpeed : family int16
egnState : family valuetype CarLibrary.Engine
maxSpeed : family int16
petName : family string
.ctor : void(string,int16,int16)
.ctor : void()
TurboBoost : void()
TurnOnRadio : void(bool,valuetype CarLibrar
get_CurrSpeed : int16()
get_EngineState : valuetype CarLibrary.Engi
get_MaxSpeed : int16()
get_PetName : string()
set_CurrSpeed : void(int16)
set_PetName : void(string)
CurrSpeed : instance int16()
EngineState : instance valuetype CarLibrary.
MaxSpeed : instance int16()
PetName : instance string()
EngineState
.class public auto ansi sealed
extends [mscorlib]System.Enum
engineAlive : public static literal valuetype Ca
engineDead : public static literal valuetype C
.assembly CarLibrary
{
.ver 1:0:2958:28922
MANIFEST
.assembly extern mscorlib
{
.publickeytoken = (B7 7A 5C 56 19 34 E0 89)
.ver 1:0:5000:0
}
.assembly extern System.Windows.Forms
{
.publickeytoken = (B7 7A 5C 56 19 34 E0 89)
.ver 1:0:5000:0
}
.assembly CarLibrary
{
.custom instance void [mscorlib]System.Reflection.AssemblyKeyNameAttr
.custom instance void [mscorlib]System.Reflection.AssemblyKeyFileAttr
.custom instance void [mscorlib]System.Reflection.AssemblyDelaySignAt
.custom instance void [mscorlib]System.Reflection.AssemblyTrademarkAt
.custom instance void [mscorlib]System.Reflection.AssemblyCopyrightAt
.custom instance void [mscorlib]System.Reflection.AssemblyProductAttr
.custom instance void [mscorlib]System.Reflection.AssemblyCompanyAttr
.custom instance void [mscorlib]System.Reflection.AssemblyConfigurati
.custom instance void [mscorlib]System.Reflection.AssemblyDescriptionI
.custom instance void [mscorlib]System.Reflection.AssemblyTitleAttrib
// --- The following custom attribute is added automatically, do not
// .custom instance void [mscorlib]System.Diagnostics.DebuggableAttr
//
.hash algorithm 0x00008004
.ver 1:0:2958:28922
}
.module CarLibrary.dll
// MVID: {9C1B1D18-8117-4711-B6D4-C68EF86D7187}
.imagebase 0x00400000
.subsystem 0x00000003
.file alignment 4096
.corflags 0x00000001
// Image base: 0x07760000

Метаданные типов



Частные сборки

- private или shared
- Находятся в каталоге приложения или в подкаталогах
- Можно переносить каталог с приложением
- Можно просто все удалить

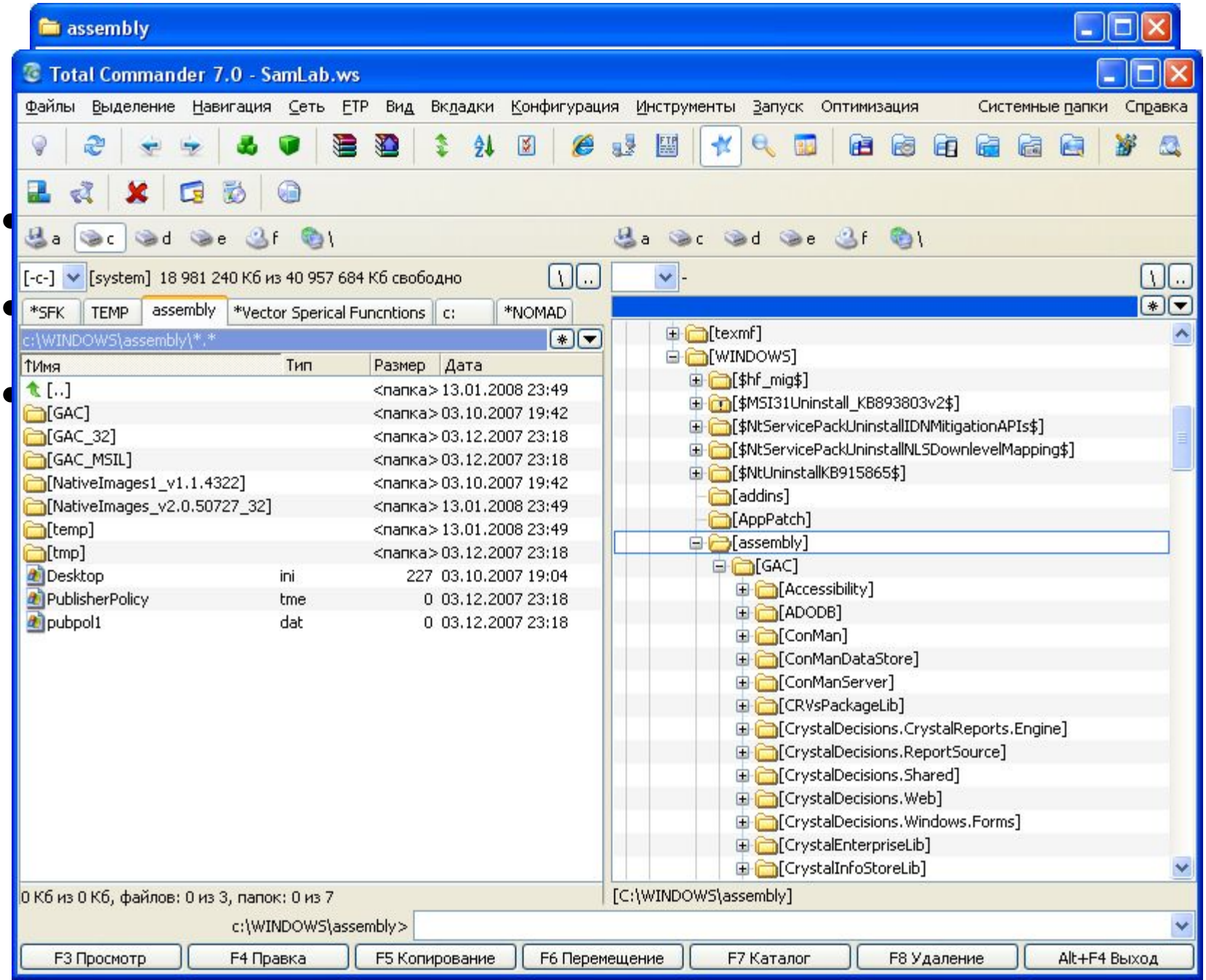
Алгоритм поиска

- В каталоге приложения dll
- В каталоге приложения exe
- Конфигурационный файл

Конфигурационный файл

```
<configuration>
    <runtime>
        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

            <probing privatePath="foo\bar"/>

        </assemblyBinding>
    </runtime>
</configuration>
```

CSSharpClient.exe → CSSharpClient.exe**.config**

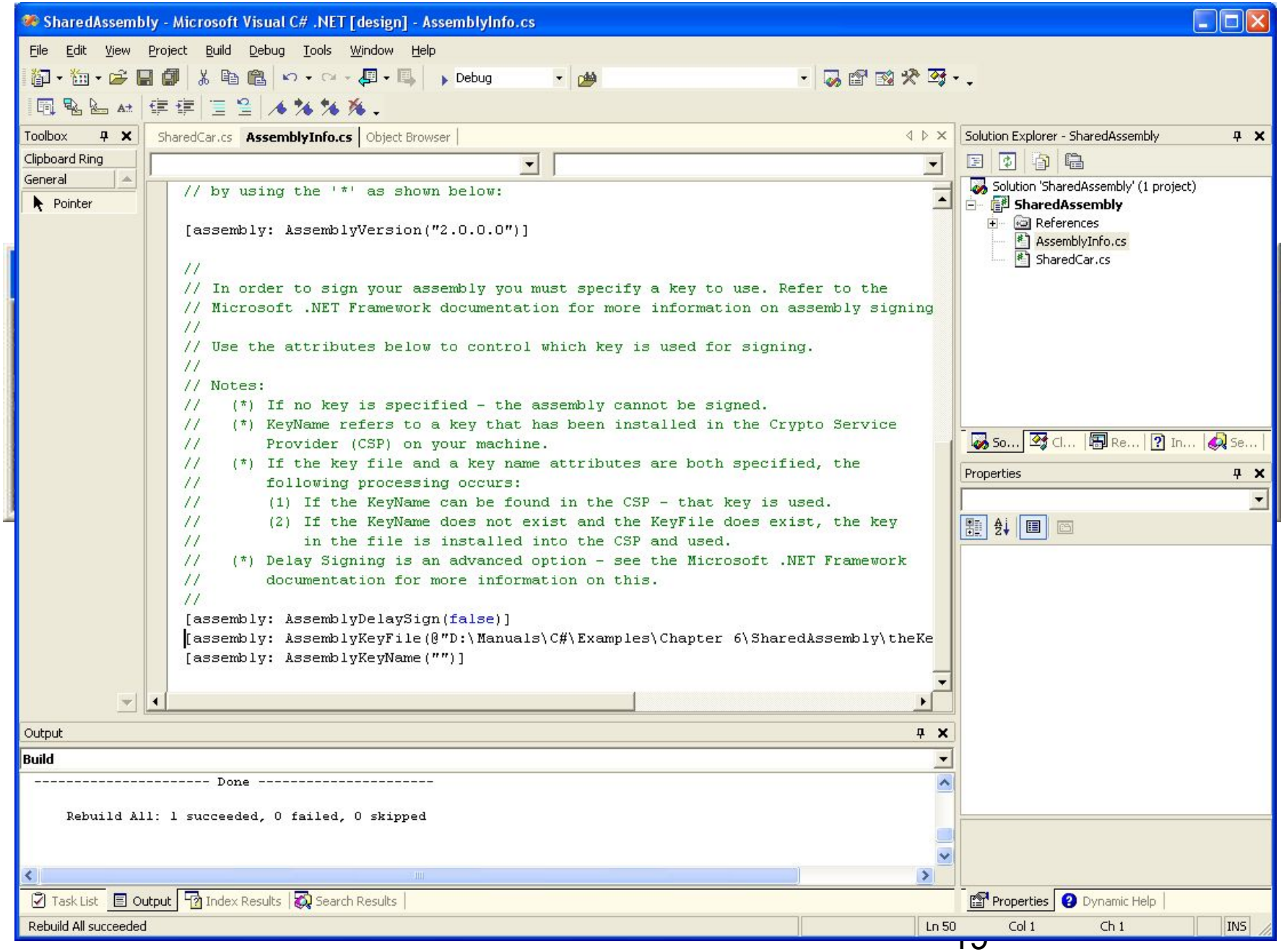

assembly
Total Commander 7.0 - SamLab.ws
Файлы Выделение Навигация Сеть FTP Вид Вкладки Конфигурация Инструменты Запуск Оптимизация Системные папки Справка
[-c-] [system] 18 981 240 Кб из 40 957 684 Кб свободно
*SFK TEMP assembly *Vector Sperical Funcntions c: *NOMAD
c:\WINDOWS\assembly*.*
↑Имя Тип Размер Дата
[..] <папка> 13.01.2008 23:49
[GAC] <папка> 03.10.2007 19:42
[GAC_32] <папка> 03.12.2007 23:18
[GAC_MSIL] <папка> 03.12.2007 23:18
[NativeImages1_v1.1.4322] <папка> 03.10.2007 19:42
[NativeImages_v2.0.50727_32] <папка> 13.01.2008 23:49
[temp] <папка> 13.01.2008 23:49
[tmp] <папка> 03.12.2007 23:18
Desktop ini 227 03.10.2007 19:04
PublisherPolicy tme 0 03.12.2007 23:18
pubpol1 dat 0 03.12.2007 23:18
0 Кб из 0 Кб, файлов: 0 из 3, папок: 0 из 7
[texmf]
[WINDOWS]
[hf_mig]
[$MSI31Uninstall_KB893803v2$]
[$NtServicePackUninstallIDNMitigationAPIs$]
[$NtServicePackUninstallNLSDownlevelMapping$]
[$NtUninstallKB915865$]
[addins]
[AppPatch]
[assembly]
[GAC]
[Accessibility]
[ADODB]
[ConMan]
[ConManDataStore]
[ConManServer]
[CRVsPackageLib]
[CrystalDecisions.CrystalReports.Engine]
[CrystalDecisions.ReportSource]
[CrystalDecisions.Shared]
[CrystalDecisions.Web]
[CrystalDecisions.Windows.Forms]
[CrystalEnterpriseLib]
[CrystalInfoStoreLib]
[C:\WINDOWS\assembly]
c:\WINDOWS\assembly>
F3 Просмотр F4 Правка F5 Копирование F6 Перемещение F7 Каталог F8 Удаление Alt+F4 Выход

«Сильные» имена сборок

- Дружественное текстовое имя и «культурная информация»
- Идентификатор версии
- Пара открытый/закрытый ключ
- Цифровая подпись

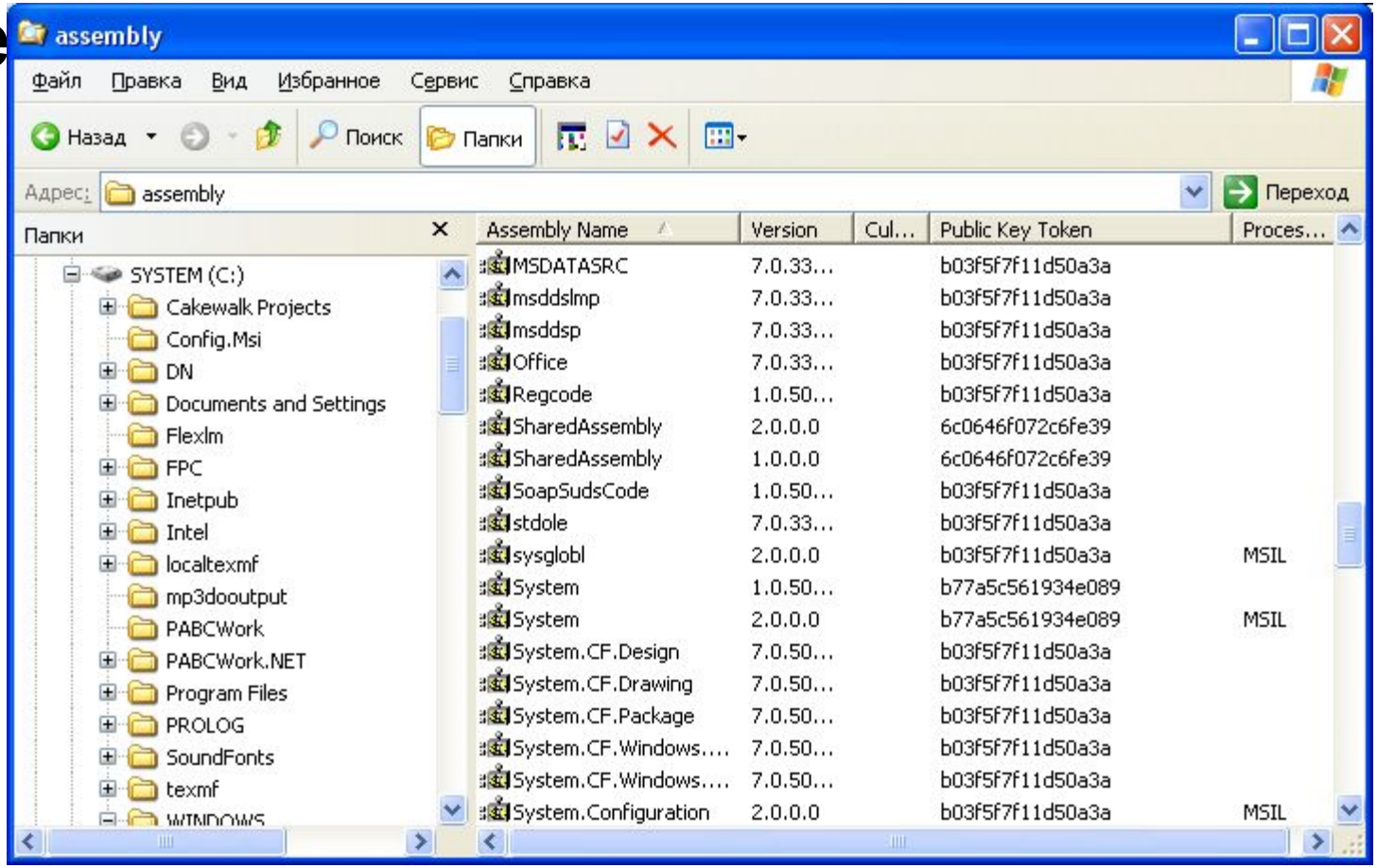
Текст сборки

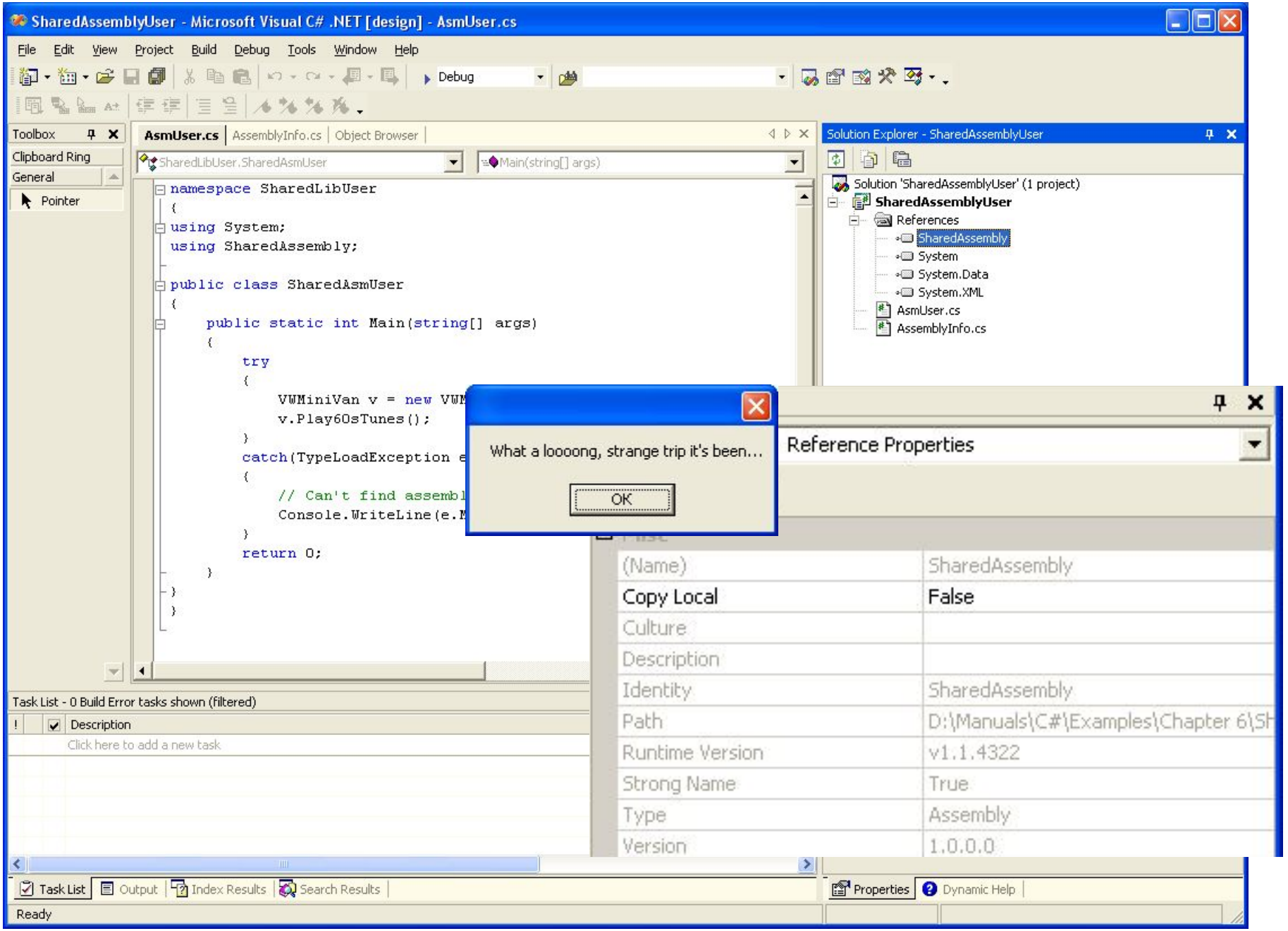
```
using System;
using System.Windows.Forms;

namespace SharedAssembly
{
public class VWMiniVan
{
    public VWMiniVan(){}

    public void Play60sTunes()
    {
      MessageBox.Show("What a loooong, strange trip it's been...");
    }

    private bool isBustedByTheFuzz = false;
    public bool Busted
    {
     get { return isBustedByTheFuzz; }
     set { isBustedByTheFuzz = value; }
     }
}
}
```


SharedAssembly - Microsoft Visual C# .NET [design] - AssemblyInfo.cs
File Edit View Project Build Debug Tools Window Help
Debug
Toolbox
Clipboard Ring
General
Pointer
SharedCar.cs
AssemblyInfo.cs
Object Browser
// by using the '*' as shown below:
[assembly: AssemblyVersion("2.0.0.0")]
//
// In order to sign your assembly you must specify a key to use. Refer to the
// Microsoft .NET Framework documentation for more information on assembly signing
//
// Use the attributes below to control which key is used for signing.
//
// Notes:
//   (*) If no key is specified - the assembly cannot be signed.
//   (*) KeyName refers to a key that has been installed in the Crypto Service
//   Provider (CSP) on your machine.
//   (*) If the key file and a key name attributes are both specified, the
//   following processing occurs:
//   (1) If the KeyName can be found in the CSP - that key is used.
//   (2) If the KeyName does not exist and the KeyFile does exist, the key
//   in the file is installed into the CSP and used.
//   (*) Delay Signing is an advanced option - see the Microsoft .NET Framework
//   documentation for more information on this.
//
[assembly: AssemblyDelaySign(false)]
[assembly: AssemblyKeyFile(@"D:\Manuals\C#\Examples\Chapter 6\SharedAssembly\theKe
[assembly: AssemblyKeyName("")]
Solution Explorer - SharedAssembly
Solution 'SharedAssembly' (1 project)
SharedAssembly
References
AssemblyInfo.cs
SharedCar.cs
Properties
Output
Build
---------- Done ----------
Rebuild All: 1 succeeded, 0 failed, 0 skipped
Task List
Output
Index Results
Search Results
Properties
Dynamic Help
Rebuild All succeeded
Ln 50
Col 1
Ch 1
INS

Установка сборки в GAC

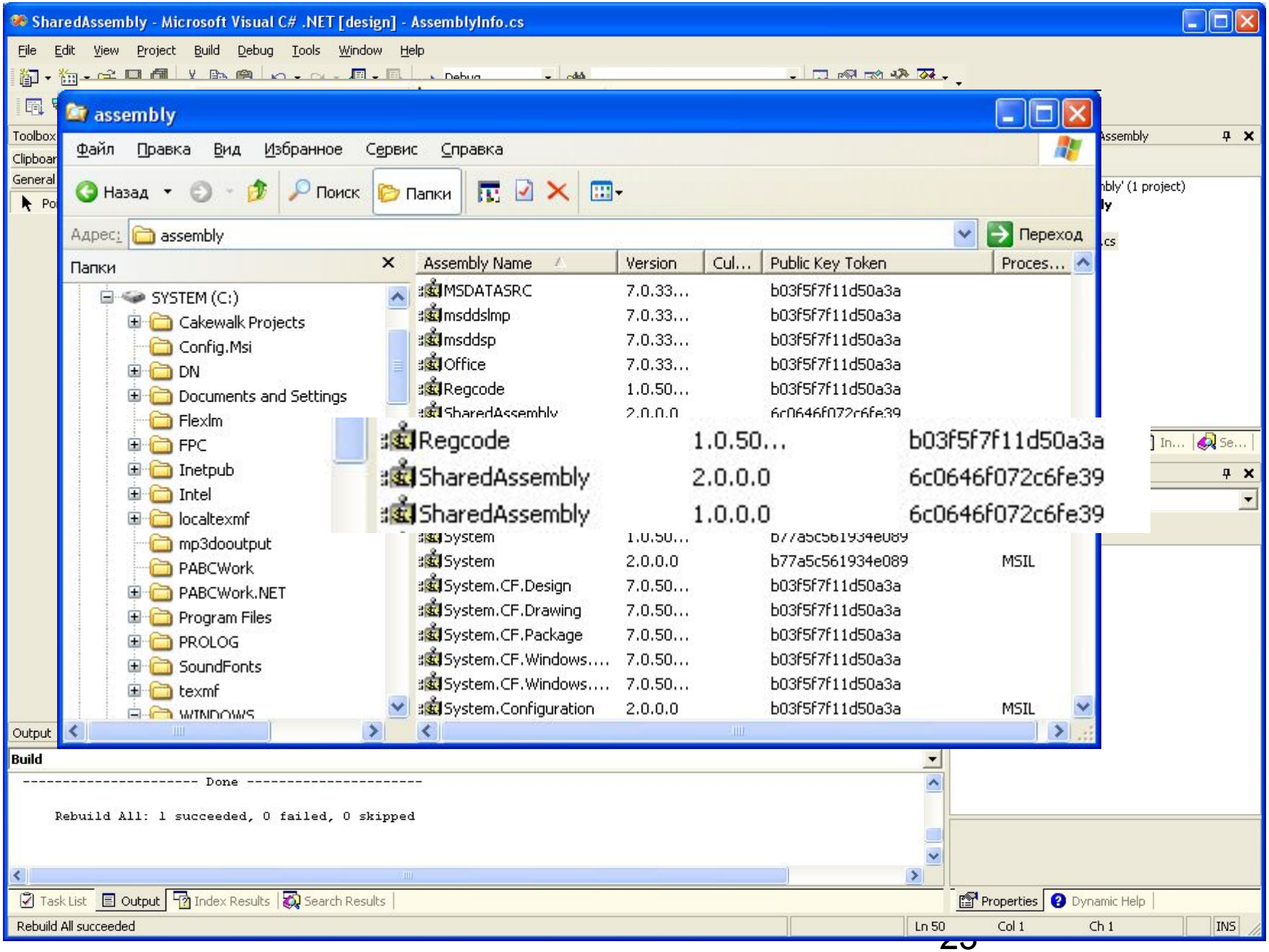
- Утилита gacutil.exe
- Пере




SharedAssemblyUser - Microsoft Visual C# .NET [design] - AsmUser.cs
File Edit View Project Build Debug Tools Window Help
Debug
Toolbox
Clipboard Ring
General
Pointer
AsmUser.cs AssemblyInfo.cs Object Browser
SharedLibUser.SharedAsmUser
Main(string[] args)
namespace SharedLibUser
{
using System;
using SharedAssembly;
public class SharedAsmUser
{
public static int Main(string[] args)
{
try
{
VWMiniVan v = new VWM
v.Play60sTunes();
}
catch(TypeLoadException e
{
// Can't find assembl
Console.WriteLine(e.M
}
return 0;
}
}
}
Solution Explorer - SharedAssemblyUser
Solution 'SharedAssemblyUser' (1 project)
SharedAssemblyUser
References
SharedAssembly
System
System.Data
System.XML
AsmUser.cs
AssemblyInfo.cs
What a loooong, strange trip it's been...
OK
Reference Properties
(Name) SharedAssembly
Copy Local False
Culture
Description
Identity SharedAssembly
Path D:\Manuals\C#\Examples\Chapter 6\SH
Runtime Version v1.1.4322
Strong Name True
Type Assembly
Version 1.0.0.0
Task List - 0 Build Error tasks shown (filtered)
Description
Click here to add a new task
Task List Output Index Results Search Results
Properties Dynamic Help
Ready

Анатомия версии сборки

- Номер основной версии (Несовместимые)
- Номер дополнительной версии (Несовместимые)
- Номер редакции (Возможно совместимые)
- Номер сборки (Quick Fix Engineering)


SharedAssembly - Microsoft Visual C# .NET [design] - AssemblyInfo.cs
File Edit View Project Build Debug Tools Window Help
assembly
Файл Правка Вид Избранное Сервис Справка
Назад Поиск Папки
Адрес: assembly
Переход
Папки
SYSTEM (C:)
Cakewalk Projects
Config.Msi
DN
Documents and Settings
Flexlm
FPC
Inetpub
Intel
localtexmf
mp3dooutput
PABCWork
PABCWork.NET
Program Files
PROLOG
SoundFonts
texmf
Assembly Name
Version
Cul...
Public Key Token
Proces...
MSDATASRC 7.0.33... b03f5f7f11d50a3a
msddslmp 7.0.33... b03f5f7f11d50a3a
msddsp 7.0.33... b03f5f7f11d50a3a
Office 7.0.33... b03f5f7f11d50a3a
Regcode 1.0.50... b03f5f7f11d50a3a
SharedAssembly 2.0.0.0 6c0646f072c6fe39
Regcode 1.0.50... b03f5f7f11d50a3a
SharedAssembly 2.0.0.0 6c0646f072c6fe39
SharedAssembly 1.0.0.0 6c0646f072c6fe39
System 2.0.0.0 b77a5c561934e089 MSIL
System.CF.Design 7.0.50... b03f5f7f11d50a3a
System.CF.Drawing 7.0.50... b03f5f7f11d50a3a
System.CF.Package 7.0.50... b03f5f7f11d50a3a
System.CF.Windows.... 7.0.50... b03f5f7f11d50a3a
System.CF.Windows.... 7.0.50... b03f5f7f11d50a3a
System.Configuration 2.0.0.0 b03f5f7f11d50a3a MSIL
Build
---------- Done ----------
Rebuild All: 1 succeeded, 0 failed, 0 skipped
Task List Output Index Results Search Results
Properties Dynamic Help
Rebuild All succeeded
Ln 50 Col 1 Ch 1 INS

Загрузка разных версий сборок

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
          <dependentAssembly>
             <assemblyIdentity name="sharedassembly"
                          publicKeyToken="6c0646f072c6fe39"
                                   culture=""/>

             <bindingRedirect oldVersion="1.0.0.0"
                          newVersion="2.0.0.0"/>

          </dependentAssembly>
       </assemblyBinding>
    </runtime>
</configuration>
```

Домены приложения

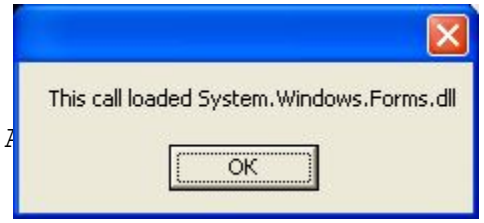
- Процесс состоит из одного или нескольких **доменов**
- В рамках домена работает один или несколько **потоков**
- Домен приложения полностью изолирует используемые в его рамках ресурсы от других доменов (за исключением удаленного доступа к данным)

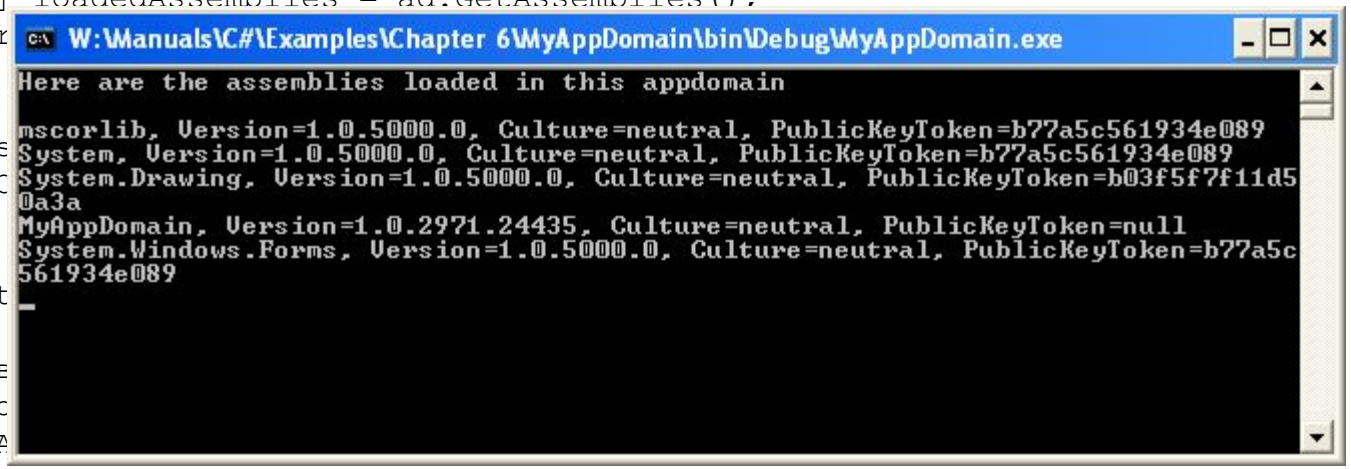
Тип AppDomain

CreateDomain()	Стат. метод для создания нового домена приложения в текущем
GetCurrentThreaId()	Стат. метод возвращает идент. текущего потока
Unload()	Стат. метод выгрузки указанного домена
BaseDirectory	Свойство, возвращает базовый каталог сборки
CreateInstance()	Создает экземпляр указанного типа, определенного в сборке
ExecuteAssembly()	Запускает на выполнение сборку, имя которой указано в качестве параметра
GetAssemblies()	Возвращает список сборок, загруженных в текущий домен приложения
Load()	Загружает сборку с домен приложения

Работаем с доменами приложения

```
namespace MyAppDomain
{
  using System;
  using System.Windows.Forms;

   // Это пространство имен требуется для работы с типом A
   using System.Reflection;
   public class MyAppDomain
   {
     public static void PrintAllAssemblies()
     {
      // Получаем список всех загруженных в текущий домен приложения  сборок
      AppDomain ad = AppDomain.CurrentDomain;
      Assembly[] loadedAssemblies = ad.GetAssemblies();
      Console.Wr

      // Теперь
     foreach(Ass
               C
     }

     public stat
     {
       // Произв
       MessageBo
       PrintAllA
       return 0;
     }
   }
}
```





Пространство имен System.Threading

- **Interlocked** – синхронизация общего доступа к данным
- **Monitor** – синхронизация потоковых объектов
- **Mutex** – примитив синхронизации
- **Thread** – поток, работающий в .NET
- **ThreadPool** – управление набором взаимосвязанных потоков
- **Timer** – определяет делегат, который будет вызван в указанное время
- **WaitHandle** – представляет все объекты синхронизации
- **ThreadStart** – делегат со ссылкой на метод, который должен быть вызван перед запуском потока
- **TimerCallback** – делегат для объектов Timer
- **WaitCallback** – делегат для рабочих элементов ThreadPool

Статические члены класса Thread

CurrentThread	Свойство только для чтения возвращает ссылку на поток, выполняемый в настоящее время
GetData()	Возвращает значение для указанного слота в текущем потоке
SetData()	Устанавливает значение для указанного слота в текущем потоке
GetDomain()	Возвращает ссылку на домен приложения, в рамках которого работает указанный поток
GetDomainID()	Возвращает идентификатор домена приложения, в рамках которого работает указанный поток
Sleep()	Приостанавливает выполнение текущего потока на указанное пользователем время

Обычные члены класса Thread

IsAlive	Свойство возвращает true или false в зависимости от того, запущен ли поток
IsBackground	Свойство для получение и установки значения, которое показывает, является ли поток фоновым
Name	Свойство для установки дружественного имени потока
Priority	Получить/установить приоритет потока
ThreadState	Возвращает информацию о состоянии потока
Interrupt()	Прерывает работу текущего потока
Join()	Ждет появления другого потока и завершается
Resume()	Продолжает работу после приостановки работы
Start()	Начинает выполнение потока, определенного делегатом ThreadStart()
Suspend()	Приостанавливает выполнение потока

Запуск вторичных потоков

```
// вспомогательный класс
internal class WorkerClass
{
   public void DoSomeWork()
   {
    // Выводим на консоль информацию о рабочем потоке
    Console.WriteLine("ID of worker thread is: {0}",
        Thread.CurrentThread.GetHashCode());

    // Выполняем некоторые действия
    Console.Write("Worker says: ");
    for (int i = o; i < 10; i++)
    {
        Console.Write(i+ ", ");
    }
    Console.WriteLine();
   }
}
```

Запуск вторичных потоков

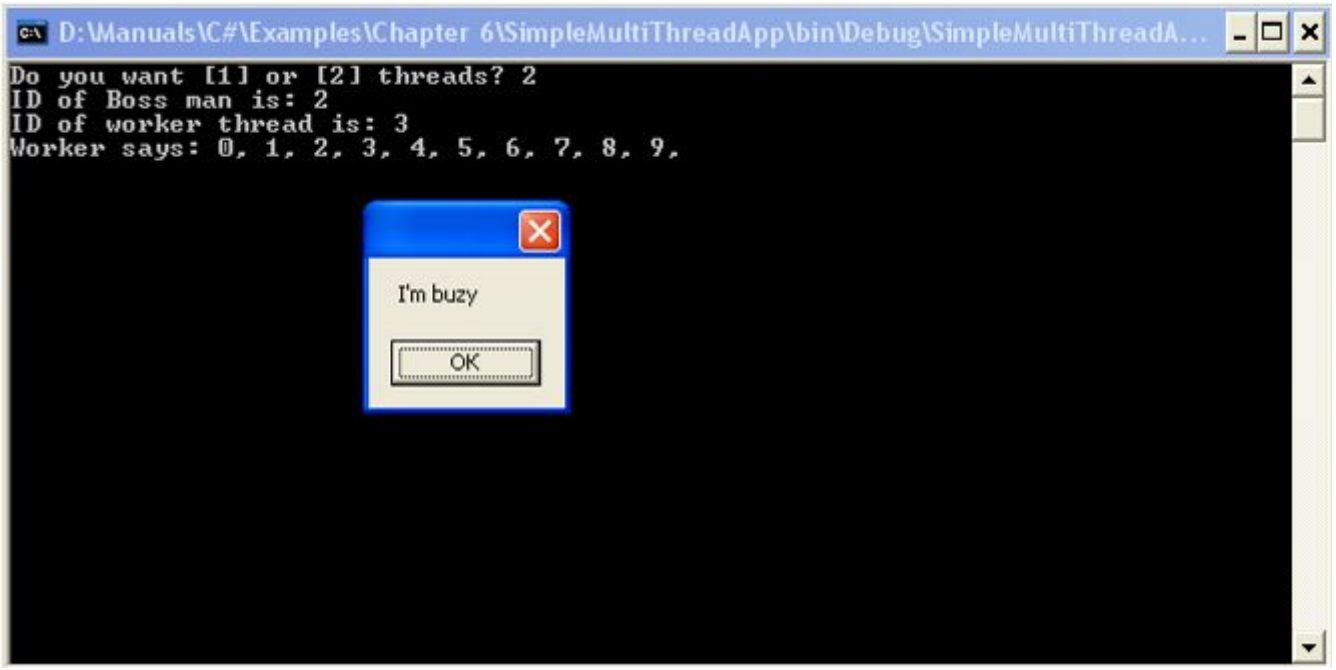
```
using System.Threading;

public class MainClass
{
    public static int Main(string[] args)
    {
     // Выводим на консоль информацию о текущем потоке
     Console.WriteLine("ID of primary thread is: {0}",
   Thread.CurrentThread.GetHashCode());

     // Создаем объект класса WorkerClass
     WorkerClass w = new WorkerClass();

    // А теперь создаем и запускаем фоновый поток
    Thread backgroundThread =
   new Thread(new ThreadStart(w.DoSomeWork));

    backgroundThread.Start();

    return 0;
   }
 }
```

Именованные потоки

```
using System.Threading;

public class MainClass
{
    public static int Main(string[] args)
    {

      // Присваиваем имя текущему потоку
      Thread primaryThread = Tread.CurrentThread;
      primaryThread.Name = "Boss man";

      Console.WriteLine("Id of {0} is {1}",  primaryThread.Name,
                        promaryThread.gethascode());

     // …



    return 0;
   }
 }
```

Параллельная работа потоков

```
public static int Mair
{
   Console.Write("Do y
   string threadCount

   // Создаем объект к
   WorkerClasss w = ne

   // Создаем дополнит
   if(threadCount = =
   { // Создаем дополн
     Thread backgrounc
     backgroundThread.
   }
   else
   w.DoSomeWork();

   // Даем первичному потоку свое задание
   MessageBox.Show("I'm busy!");

   return 0;
}
```



Как «усыпить» поток

```
internal class WorkerClass
{
    public void DoSomeWork()
    {
    // Выводим информацию о рабочем потоке
    Console.WriteLine("ID of worker thread is: {0}",
           Thread.CurrentThread.GetHashCode());

    // Делаем работу "с перекурами"
    Console.Write("Worker says: ");
    for(int i = 0; i < 5; i++)
        {
           Console.WriteLine(i + ", ");
           Thread.Sleep(5000);
        }
    Console.WriteLine();
    }
}
```

Одновременный доступ к данным из разных потоков

```
public class MainClass
{
  public static int Main(string[] args)
  {
    // Создаем единственный объект класса WorkerClass
    WorkerClass w = new WorkerClass();

    // Создаем три отдельных потока, каждый из которых производит вызов
    // к одному и тому же объекту
    Thread workerThreadA = new Thread(new ThreadStart(w.DoSomeWork));
    Thread workerThreadB = new Thread(new ThreadStart(w.DoSomeWork));
    Thread workerThreadC = new Thread(new ThreadStart(w.DoSomeWork));

    // Теперь запускаем все три потока
    WorkerThreadA.Start();
    WorkerThreadB.Start();
    WorkerThreadC.Start();
    return 0;
  }
}
```

Ключевое слово lock

```
internal class WorkerClass
{
  public void DoSomeWork()
  {
    // Только один поток в конкретный момент времени сможет
    // выполнять этот код!
    lock(this)
    {
   // Делаем все ту же работу
   for(int i=0; i < 5; i++)
   {
    Console.WriteLine("Worker says: {0},", i);
   }
    }
  }
}
```

Использование System.Threading.Monitor

```
internal class WorkerClass
{
  public void DoSomeWork()
  {
   // Определяем элемент для мониторинга в целях синхронизации
  Monitor.Enter(this);
  try
   {
    // Выполнить работу...
    for(int i = 0; i < 5; i++)
    {
        Console.WriteLine("Worker says: {0},", i);
    }
   }
  finally
   {
    // Была ошибка или нет, а из монитора придется выйти
    Monitor.Exit(this);
   }
  }
}
```

Опасность одновременного изменения переменной

```
public class IHaveNoIdea
{
    private long refCount = 0;

    public void AddRef()
    { ++refCount; }

    public void Release()
    {
        if(--refCount == 0)
        {
            GC.Collect();
        }
    }
}
```

Применение System.Threading.Interlocked

```
public class IHaveNoIdea
   {
    private long refCount = 0;

    public void AddRef()
    {
        Interlocked.Increment(ref refCount);
    }

    public void Release()
    {
        if (Interlocked.Decrement(ref refCount) == 0)
        {
            GC.Collect();
        }
    }
   }
```