# Практическая работа 23. Создание сборки

**Цель занятия:** Получить практический опыт создания сборки.

**Перечень оборудования и программного обеспечения:**

* Персональный компьютер
* Microsoft Office (Word, Visio)
* Microsoft Visual Studio

**Краткие теоретические сведения**

**Сборки .NET**

**Роль сборок в приложениях .NET**

Когда мы создаем приложение в результате компиляции в Visual Studio или в консоли, результатом этой работы является файл exe или dll (в зависимости от выбранных настроек), который называется сборкой приложения. Сборка является базовой структурной единицей в .NET, на уровне которой проходит контроль версий, развертывание и конфигурация приложения.

Сборки кристаллизуют всю библиотеку классов .NET - при написании кода и создании сборки своего приложения мы используем пространства имен, которые размещены в других сборках .NET.

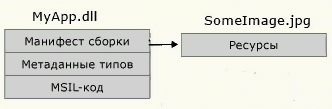
**Сборки имеют следующие составляющие:**

* Манифест, который содержит метаданные сборки
* Метаданные типов. Используя эти метаданные, сборка определяет местоположение типов в файле приложения, а также места размещения их в памяти
* Собственно код приложения на языке MSIL, в который компилируется код C#
* Ресурсы

Все эти компоненты могут находиться в одном файле, и тогда сборка представляет один единственный файл в формате exe или dll.



Но также может быть, что эти компоненты хранятся в отдельных файлах. Например, есть основной файл exe, который имеет метаданные сборки и типов и который использует дополнительные файлы ресурсов - различные изображения, звуковые файлы, вспомогательные модули, элементы интерфейса, локализованные для различных культур.



**Манифест сборки**

Ключевым компонентом сборки является ее манифест. Если у сборки отсутствует манифест, то заключенный в ней код MSIL выполняться не будет. Манифест может находиться в одном файле с исполняемым кодом сборки, а может размещаться и в отдельном файле.

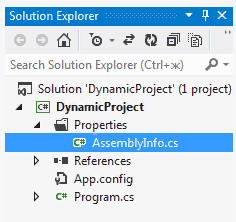
Манифест хранит следующие данные:

* Имя сборки
* Номер версии: основной и дополнительный номера. Используется для управления версиями
* Язык и региональные параметры: информация о языке и региональных параметрах, которые поддерживает сборка
* Информация о строгом имени: открытый ключ издателя
* Список всех файлов сборки: хэш и имя каждого из входящих в сборку файлов
* Список ссылок на другие сборки, которые использует текущая сборка
* Список ссылок на типы, используемые сборкой

Таким образом, манифест позволяет системе определить все файлы, входящие в сборку, сопоставить ссылки на типы, ресурсы, сборки с их файлами, управлять контролем версий.

**Атрибуты сборки**

По умолчанию Visual Studio при создании проекта добавляет файл AssemblyInfo.cs, который можно найти в узле Properties:



Это обычный файл на языке C# и в моем случае выглядит так:

using System.Reflection;

using System.Runtime.CompilerServices;

using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following

// set of attributes. Change these attribute values to modify the information

// associated with an assembly.

[assembly: AssemblyTitle("DynamicProject")]

[assembly: AssemblyDescription("")]

[assembly: AssemblyConfiguration("")]

[assembly: AssemblyCompany("")]

[assembly: AssemblyProduct("DynamicProject")]

[assembly: AssemblyCopyright("Copyright ©  2014")]

[assembly: AssemblyTrademark("")]

[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible

// to COM components.  If you need to access a type in this assembly from

// COM, set the ComVisible attribute to true on that type.

[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM

[assembly: Guid("8794dabf-2f91-423d-87c4-6317e2913be7")]

// Version information for an assembly consists of the following four values:

//

//      Major Version

//      Minor Version

//      Build Number

//      Revision

//

// You can specify all the values or you can default the Build and Revision Numbers

// by using the '\*' as shown below:

// [assembly: AssemblyVersion("1.0.\*")]

[assembly: AssemblyVersion("1.0.0.0")]

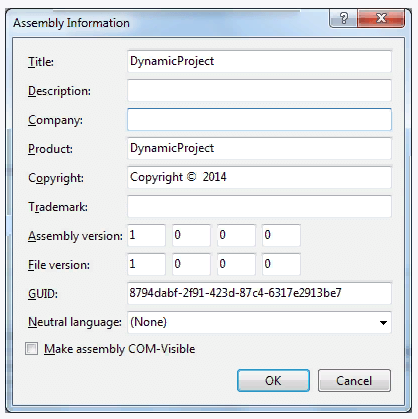
[assembly: AssemblyFileVersion("1.0.0.0")]

Смысл этого файла состоит в том, что он задает настройки манифесту сборки. Через атрибуты типа [assembly: AssemblyVersion("1.0.0.0")] можно установить значения в манифесте. Префикс assembly: перед атрибутом указывает на то, что это атрибут уровня сборки, в данном случае атрибут номера версии сборки.

В принципе, исходя из названия, предназначение многих атрибутов итак понятно:

* AssemblyCompany: название компании
* AssemblyConfiguration: конфигурация сборки (Retail или Debug)
* AssemblyCopyright: авторское право на программу
* AssemblyDefaultAlias: псевдоним по умолчанию, используемый при ссылке на данную сборку из других сборок
* AssemblyDescription: краткое описание сборки
* AssemblyProduct: информация о продукте
* AssemblyTitle: название сборки как информационного продукта
* AssemblyTrademark: сведения о торговой марке
* AssemblyCulture: задает язык и региональные параметры, поддерживаемые сборкой (например, установка русской культуры: [assembly:AssemblyCultureAttribute("ru")])
* AssemblyInformationalVersion: полная версия сборки
* AssemblyVersion: версия сборки
* AssemblyFileVersion: номер версии файла Win32. По умолчанию совпадает с версией сборки.

Также есть графический способ определения информации о сборке. Для этого нажмем в окне Solution Explorer (Обозреватель решений) на название проекта правой кнопкой мыши и выберем в появившемся меню пункт Properties (Свойства). В открывшемся окне нажмем на кнопку Assembly Information..., и нам предстанет окно изменения атрибутов:



После любых изменений, произведенных в этом окне, соответствующей информацией будет автоматически обновлен и файл AssemblyInfo.cs.

**Задания**

1 Изучить теоретические сведения и задание к работе

2 В соответствии с вариантом задания создать сборку для вашего проекта.

**Порядок выполнения работы**

**…**

**Содержание отчета**

1 Название работы

2 Цель работы

3 Перечень оборудования и программного обеспечения

4 Задание

5 Порядок выполнения работы

6 Вывод

**Варианты заданий**

Варианты заданий представлены в практической работе 2.

**Используемая литература**

1. Федорова Г.Н. Участие в интеграции программных модулей: учеб. пособие для студ. учреждений сред. проф. образования / Г.Н. Федорова. –М.: Издательский центр «Академия», 2020. – 304 с.
2. Рудаков А.В. Технология разработки программных продуктов: учеб. пособие для студ. учреждений сред. проф. образования – 12-е изд., стер. – М.: Издательский центр «Академия», 2018. – 208 с.
3. Федорова Г.Н. Разработка программных модулей программного обеспечения для компьютерных систем: учебник для студ. учреждений сред. проф. образования / Г.Н. Федорова. – М.: Издательский центр «Академия», 2020. – 336 с.
4. https://metanit.com/sharp/tutorial/10.1.php