IT-Academy

ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР
ПРОГРАММИРОВАНИЯ И ВЫСОКИХ ТЕХНОЛОГИЙ

2020г.

Сборки

- ✓ Понятие сборок в .NET
- ✓ Подпись сборок и строгие имена
- ✓ Домен приложения
- ✓ Загрузка сборок в домен
- ✓ Маршалинг, обмен данными между доменами
- ✓ GAC

Понятие сборок в .NET

Результатом компиляции в **Visual Studio** или в консоли 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 имеют следующие составляющие:

- Манифест, который содержит метаданные сборки.
- Метаданные типов. Используя эти метаданные, сборка определяет размещение типов в памяти.
- 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 или эти компоненты могут хранится в отдельных файлах. Основной файл **exe**, который имеет метаданные сборки и типов и который использует дополнительные файлы ресурсов.

Понятие сборок в .NET

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 - используется для управления версиями.
- Язык и региональные параметры.
- 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.

Понятие сборок в .NET

По способу взаимодействия с другими сборками и приложениями сборки можно разделить на две категории: закрытые и разделяемые.

Закрытые сборки это обычные сборки приложения, которые создаются, к примеру в **Visual Studio**. При создании библиотеки классов **dll** создается закрытая сборка. Такую закрытую сборку можно использовать подключив ее к другому проекту. Чтобы это сделать можно просто положить сборку рядом с исполняемым файлом и добавить в проект ссылку на нее через **«Add Reference»**.

Разделяемые сборки находятся в глобальном кэше сборок (**Global Assembly Cache**). Самое очевидное отличие между разделяемой и закрытой сборкой состоит в том, что одна копия разделяемой сборки может использоваться сразу в нескольких приложениях на одной и той же машине.

Подпись сборок и строгие имена

Формально любое строгое имя состоит из набора взаимосвязанных данных, большая часть из которых указывается с помощью перечисленных ниже атрибутов уровня сборки:

- Дружественное имя сборки (которое представляет собой имя сборки без файлового расширения).
- Номер версии сборки (назначается в атрибуте [**AssemblyVersion**]).
- Значение открытого ключа (назначается в атрибуте [**AssemblyKeyFile**]).
- Значение, обозначающее культуру, которое является необязательным и может предоставляться для локализации приложения (присваивается в атрибуте [**AssemblyCulture**]).
- Вставляемая цифровая подпись, созданная с использованием хеш-кода по содержимому сборки и значения секретного ключа.

Благодаря строгому имени гарантируется уникальность сборки в глобальном кэше.

Строгие имена также обеспечивают определенную степень защиты от возможной подделки содержимого сборок.

Домен приложения

В **.NET** исполняемые файлы не обслуживаются прямо внутри процесса **Windows**, как это происходит в случае традиционных неуправляемых приложений. Вместо этого они обслуживаются в отдельном логическом разделе внутри процесса, который называется доменом приложения (**Application Domain - AppDomain**).

Домен приложения – это механизм, реализованный в **.NET**, который позволяет запустить группу приложений в одном процессе, обеспечивая относительную изоляцию их друг от друга, в то же время позволяя им взаимодействовать друг с другом значительно быстрее, чем в случае отдельных процессов.

Один процесс может содержать любое число доменов приложения, каждый из которых полностью изолирован от других доменов приложения в рамках данного процесса (а также любого другого процесса). С учетом этого следует понимать, что приложение, выполняющееся в одном домене приложения, не может получить данные (в частности, значения глобальных переменных или статических полей) другого домена приложения иначе, как с помощью протокола удаленного взаимодействия .NET.

Домен приложения

Для управления домена платформа .NET предоставляет класс **AppDomain**.

Рассмотрим некоторые основные методы и свойства данного класса:

- Свойство **BaseDirectory** – базовый каталог, который используется для получения сборок (как правило, каталог самого приложения).
- Свойство **CurrentDomain** – домен текущего приложения.
- Свойство **FriendlyName** – имя домена приложения.
- Свойство **SetupInformation** – представляет объект AppDomainSetup и хранит конфигурацию домена приложения.
- Метод **ExecuteAssembly()** – запускает сборку exe в рамках текущего домена приложения.
- Метод **GetAssemblies()** – получает набор сборок .NET, загруженных в домен приложения.

Загрузка сборок в домен

Сборки, на которые ссылается программа, загружаются автоматически средой **CLR**, но в текущий домен приложения можно также динамически загрузить конкретные сборки.

Для загрузки сборок класс **AssemblyLoadContext** предоставляет ряд методов:

- **LoadFromAssemblyName (AssemblyName assemblyName)** – загружает определенную сборку по имени, которое представлено типом **System.Reflection.AssemblyName.**
- **LoadFromAssemblyPath (string assemblyPath)** – загружает сборку по определенному пути (путь должен быть абсолютным).
- **LoadFromStream (System.IO.Stream stream)** – загружает определенную сборку из потока **Stream.**

В **.NET Framework** можно было создавать вторичные домены и загружать в них различные сборки. В **.NET Core** можно использовать только один домен.

Маршалинг, обмен данными между доменами

Маршаллинг – это передача сущности из одного контекста в другой.

Сериализация – это запись в виде последовательности элементов.

Маршаллинг – это процесс более высокого уровня, чем сериализация. Обычно, если надо передать структуру данных из одного процесса в другой – ее сериализуют, передают и десериализуют. Если параметр двусторонний – то и передавать его надо будет два раза, при этом это будет одна операция маршаллинга.

Или возможна передача по ссылке, когда на другой стороне канала создается прокси-объект, а через канал передается не внутреннее состояние объекта, а вызовы его методов.

Если речь идет о взаимодействии между управляемым и неуправляемым кодом – то маршаллинг заключается в фиксировании адресов объектов или в копировании структур между управляемой и неуправляемой памятью, сериализация тут вовсе не используется.

GAC

Глобальный кэш сборок **.NET** – это кэш кода. Глобальный кэш сборок устанавливается автоматически на каждом компьютере с установленной средой **CLR .NET**. Любое приложение, установленное на компьютере, может получать доступ к глобальному кэшу сборок. Глобальный кэш сборок содержит сборки, предназначенные для совместного использования несколькими приложениями на компьютере.

К сборкам, расположенным в **GAC**, предъявляется несколько требований. В частности, они должны использовать строгое имя, соблюдать строгую схему указания версий и допускать исполнение нескольких версий кода в рамках единого приложения.

Устанавливайте сборку только в глобальном кэше сборок, когда вам потребуется предоставить доступ к сборке. Если не требуется предоставлять доступ к сборке явно, рекомендуется хранить ее в закрытом виде, а сборку указать в каталоге приложения. Кроме того, вам не нужно устанавливать сборку в глобальный кэш сборок, чтобы сделать сборку доступной для взаимодействия **COM** или неуправляемого кода.

GAC

Существует несколько причин для установки сборки в глобальном кэше сборок.

- **Общее расположение** – используемые несколькими приложениями сборки можно располагать в глобальном кэше сборок.

- **Безопасность файлов** – администраторы часто защищают папку systemroot с помощью списка управления доступом, определяющего права на запись и выполнение. Так как глобальный кэш сборок размещается в корневом каталоге системы, он наследует список управления доступом этого каталога.

- **Управление параллельными версиями** – в глобальном кэше сборок может храниться несколько сборок, имеющих одинаковые имена, но различные сведения о версии.

- **Дополнительное место для поиска** – перед проверкой или использованием сведений о базе кода в файле конфигурации среда **CLR** ищет в глобальном кэше сборки, соответствующие запросу.