**1)Рефлексия** представляет собой процесс выявления типов во время выполнения приложения.

2) **Получение типа** Чтобы управлять типом и получать всю информацию о нем, нам надо сперва **получить** данный **тип**. Это можно сделать тремя **способами**: с помощью ключевого слова typeof, с помощью метода GetType() класса Object и применяя статический метод **Type**. GetType() .

* 3) **Assembly**: класс, представляющий сборку и позволяющий манипулировать этой сборкой
* **AssemblyName**: класс, хранящий информацию о сборке
* **MemberInfo**: базовый абстрактный класс, определяющий общий функционал для классов EventInfo, FieldInfo, MethodInfo и PropertyInfo
* **EventInfo**: класс, хранящий информацию о событии
* **FieldInfo**: хранит информацию об определенном поле типа
* **MethodInfo**: хранит информацию об определенном методе
* **PropertyInfo**: хранит информацию о свойстве
* **ConstructorInfo**: класс, представляющий конструктор
* **Module**: класс, позволяющий получить доступ к определенному модулю внутри сборки
* **ParameterInfo**: класс, хранящий информацию о параметре метода
* 4) Метод **FindMembers()** возвращает массив объектов MemberInfo данного типа
* Метод **GetConstructors()** возвращает все конструкторы данного типа в виде набора объектов ConstructorInfo
* Метод **GetEvents()** возвращает все события данного типа в виде массива объектов EventInfo
* Метод **GetFields()** возвращает все поля данного типа в виде массива объектов FieldInfo
* Метод **GetInterfaces()** получает все реализуемые данным типом интерфейсы в виде массива объектов Type
* Метод **GetMembers()** возвращает все члены типа в виде массива объектов MemberInfo
* Метод **GetMethods()** получает все методы типа в виде массива объектов MethodInfo
* Метод **GetProperties()** получает все свойства в виде массива объектов PropertyInfo
* Свойство **Name** возвращает имя типа
* Свойство **Assembly** возвращает название сборки, где определен тип
* Свойство **Namespace** возвращает название пространства имен, где определен тип
* Свойство **IsArray** возвращает true, если тип является массивом
* Свойство **IsClass** возвращает true, если тип представляет класс
* Свойство **IsEnum** возвращает true, если тип является перечислением
* Свойство **IsInterface** возвращает true, если тип представляет интерфейс

5) С помощью динамической загрузки мы можем реализовать технологию позднего связывания. **Позднее связывание** позволяет создавать экземпляры некоторого типа, а также использовать его во время выполнения приложения.

Использование позднего связывания менее безопасно в том плане, что при жестком кодировании всех типов (ранее связывание) на этапе компиляции мы можем отследить многие ошибки. В то же время позднее связывание позволяет создавать расширяемые приложения, когда дополнительный функционал программы неизвестен, и его могут разработать и подключить сторонние разработчики.

Ключевую роль в позднем связывании играет класс **System.Activator**. С помощью его статического метода **Activator.CreateInstance()** можно создавать экземпляры заданного типа.в

6) Чтобы динамически загрузить сборку в приложение, надо использовать статические методы **Assembly.LoadFrom()** или **Assembly.Load()**.