*1. Что такое Рефлексия?*

**Рефлексия** представляет собой процесс выявления типов во время выполнения приложения. Каждое приложение содержит набор используемых классов, интерфейсов, а также их методов, свойств и прочих кирпичиков, из которых складывается приложение. И рефлексия как раз и позволяет определить все эти составные элементы приложения.

*2. Охарактеризуйте классы из пространства имен System.Reflection.*

Основной функционал рефлексии сосредоточен в пространстве имен System.Reflection. В нем мы можем выделить следующие основные классы:

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

*3. Как можно использовать класс System.Type? Перечислит его свойства и методы.*

Класс System.Type представляет изучаемый тип, инкапсулируя всю информацию о нем. С помощью его свойств и методов можно получить эту информацию. Некоторые из его свойств и методов:

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 IsAbstract возвращает true, если тип является абстрактным

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

*4. Приведите три способа получения типа.*

С помощью ключевого слова typeof, с помощью метода GetType() класса Object и применяя статический метод Type.GetType().

*5. Как динамически загрузить сборку в приложение?*

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

Метод LoadFrom() принимает в качестве параметра путь к сборке.

Метод Load() в качестве его параметра передается дружественное имя сборки, которое нередко совпадает с именем приложения.

*6. Что такое позднее связывание?*

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.

С помощью его статического метода **Activator.CreateInstance()** можно создавать экземпляры заданного типа.