1. Что такое рефлексия?

Рефлексия — это механизм, который позволяет получать информацию о типах, методах, свойствах, событиях и других членах классов во время выполнения программы. С помощью рефлексии можно динамически создавать объекты, вызывать методы и доступ к членам классов, что делает её полезной для создания более гибких и адаптивных приложений.

2. Какие есть способы получения объекта типа Type. Приведите три способа получения типа.

1. Использование оператора typeof:

Type type = typeof(MyClass);

2. Использование метода GetType на экземпляре объекта:

MyClass obj = new MyClass();

Type type = obj.GetType();

3. Использование метода Assembly.GetType:

Type type = Assembly.GetExecutingAssembly().GetType("Namespace.MyClass");

3. Охарактеризуйте классы из пространства имен System.Reflection.

- Assembly: Представляет сборку в .NET. Позволяет получать информацию о метаданных, загружать сборки, а также управлять версиями и зависимостями.

- Module: Представляет модуль в сборке, который содержит типы и другие члены.

- Type: Предоставляет информацию о типах, включая классы, интерфейсы, структуры и перечисления.

- MethodInfo: Предоставляет информацию о методах, таких как параметры, возвращаемые значения, и позволяет вызывать методы динамически.

- PropertyInfo: Позволяет получать информацию о свойствах, включая типы и методы доступа.

- FieldInfo: Предоставляет информацию о полях, их типах и доступных модификаторах.

- ConstructorInfo: Представляет конструктор типа и позволяет создавать экземпляры объектов.

4. Как можно использовать класс System.Type? Перечислите его свойства и методы.

Свойства:

- Name: Получает имя типа.

- Namespace: Получает пространство имен типа.

- Assembly: Получает сборку, в которой определен тип.

- BaseType: Получает базовый тип для данного типа.

- IsClass: Определяет, является ли тип классом.

- IsInterface: Определяет, является ли тип интерфейсом.

Методы:

- GetMethods(): Получает методы, объявленные в типе.

- GetProperties(): Получает свойства, объявленные в типе.

- GetFields(): Получает поля, объявленные в типе.

- GetConstructors(): Получает конструкторы, объявленные в типе.

- GetInterfaces(): Получает интерфейсы, которые реализует данный тип.

- GetCustomAttributes(): Получает атрибуты, примененные к типу.

5. Что такое позднее и раннее связывание?

- Раннее связывание (early binding): Компилятор знает о типах и их методах во время компиляции.

Это обеспечивает безопасность типов и повышает производительность, так как вызовы методов определены заранее.

- Позднее связывание (late binding): Типы и методы определяются во время выполнения.

Это позволяет использовать динамические типы и методы, что делает код более гибким,

но может привести к ошибкам во время выполнения и снижению производительности.

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.

6. Как динамически загрузить сборку в приложение?

Для динамической загрузки сборки можно использовать метод Assembly.Load или Assembly.LoadFrom. Например:

Assembly assembly = Assembly.Load("MyAssembly");

или

Assembly assembly = Assembly.LoadFrom("C:\\path\\to\\MyAssembly.dll");

8. Для чего предназначены BindingFlags? Какую комбинацию флагов необходимо использовать, чтобы иметь возможность получать приватные члены класса?

BindingFlags — это перечисление, которое определяет параметры для поиска членов типов.

Оно позволяет уточнить, какие члены (методы, свойства, поля и т. д.) нужно получить, и в каком контексте (например, только публичные или только приватные).

Чтобы получить приватные члены класса, необходимо использовать комбинацию флагов BindingFlags следующим образом:

BindingFlags flags = BindingFlags.Public | BindingFlags.NonPublic | BindingFlags.Instance | BindingFlags.Static;

Таким образом, вы получите все публичные и приватные члены как экземпляров, так и статических членов класса.