



OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

Онлайн-образование

Не забыть включить запись!





Меня хорошо видно && слышно?

Ставьте +, если все хорошо
Напишите в чат, если есть проблемы

Правила вебинара



Активно участвуем



Задаем вопрос в чат или голосом



Off-topic обсуждаем в Slack #канал группы или #general



Вопросы вижу в чате, могу ответить не сразу

Reflection (Рефлексия / Отражение)



Усманов Александр

Ведущий разработчик, стаж 10+ лет

<https://t.me/AlexanderUsmanov>

Цели вебинара | После занятия вы

1

Узнаете что такое рефлексия

2

Сможете обрабатывать экземпляры разных
или неизвестных заранее классов

3

Сможете писать свои компоненты

The image features a background of a dense city skyline, likely New York City, with numerous skyscrapers. The entire image is overlaid with a semi-transparent network of white lines and dots, creating a digital or technological feel. The color palette is dominated by shades of blue and green, with a gradient from a lighter green on the left to a darker blue on the right.

Теория

Сборки в .NET

Сборки представляют собой базовые элементы

- развертывания,
 - управления версиями,
 - повторного использования,
 - назначения областей активации,
 - прав доступа для приложений
- на основе платформы .NET.

Сборки в .NET

MyAssembly.dll

Assembly manifest

Type metadata

MSIL code

Resources

Рефлексия

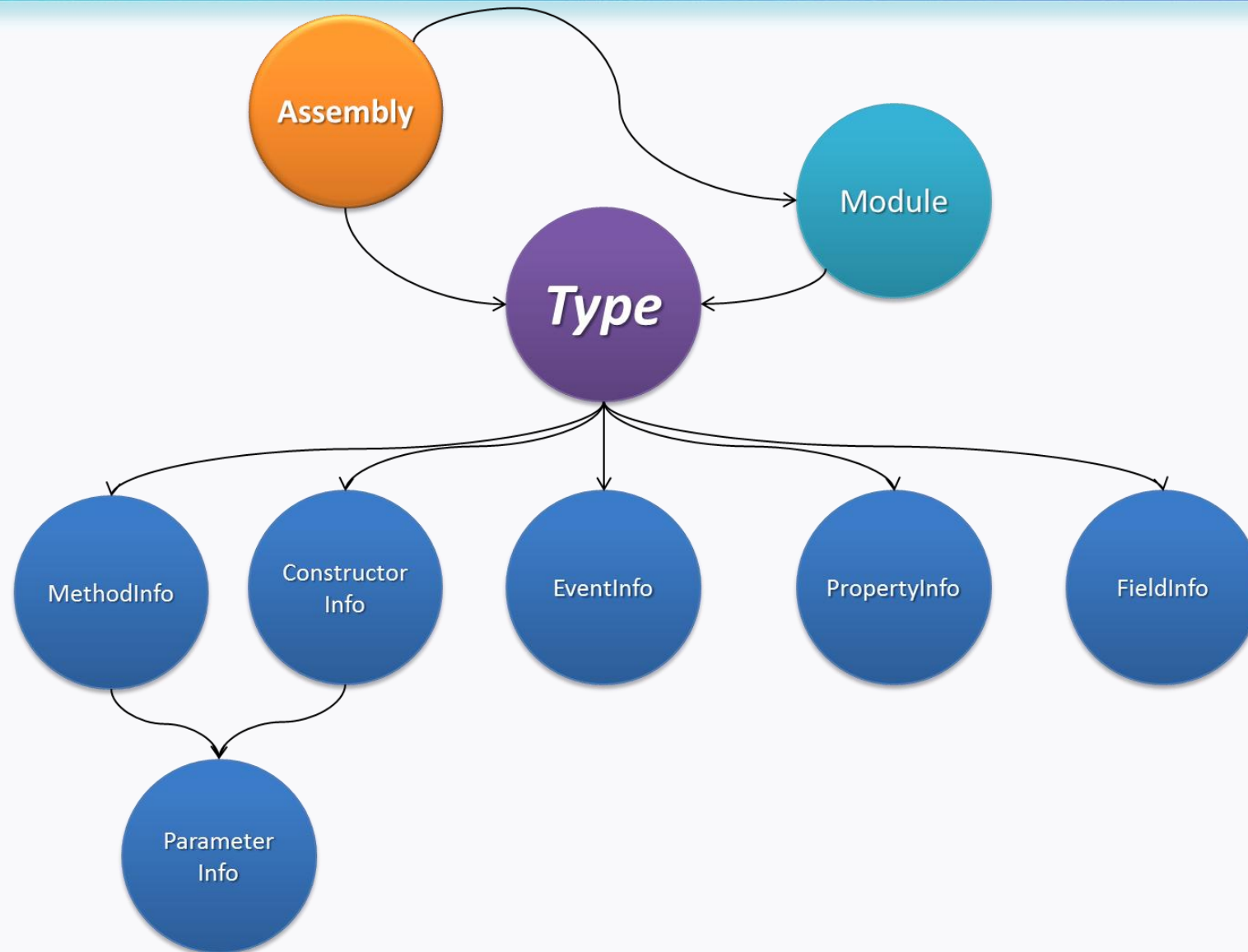
Механизм отражения позволяет получать объекты (типа `Type`) во время выполнения программы, которые описывают сборки, модули и типы.

Рефлексия

Отражение можно использовать

- для динамического создания экземпляра типа во время выполнения
- для получения типа из существующего объекта и вызова его методов или доступа к его полям и свойствам
- при необходимости доступа к атрибутам в метаданных программы (для извлечение информации, сохраненной в атрибуте)
- для выполнения позднего связывания, которое обеспечивает доступ к методам в типах, созданных во время выполнения

Рефлексия (System.Reflection)



Как начать?

```
Type type = typeof(MyClass);
```

```
MyClass myClass = new MyClass();
```

```
Type type = myClass.GetType();
```

```
Type type = Type.GetType(typeName: "Namespace.MyClass, Library",  
throwOnError: false, ignoreCase: true);
```

Что можно получить?

System.Type

Методы

- **FindMembers()** возвращает массив объектов MemberInfo данного типа
- **GetConstructors()** возвращает все конструкторы данного типа в виде набора объектов ConstructorInfo
- **GetEvents()** возвращает все события данного типа в виде массива объектов EventInfo
- **GetFields()** возвращает все поля данного типа в виде массива объектов FieldInfo
- **GetInterfaces()** получает все реализуемые данным типом интерфейсы в виде массива объектов Type
- **GetMembers()** возвращает все члены типа в виде массива объектов MemberInfo
- **GetMethods()** получает все типы в виде массива объектов MethodInfo
- **GetProperties()** получает все свойства в виде массива объектов PropertyInfo

СВОЙСТВА

Name возвращает имя типа

- **Assembly** возвращает название сборки, где определен тип
- **Namespace** возвращает название пространства имен, где определен тип
- **IsArray** возвращает true, если тип является массивом
- **IsClass** возвращает true, если тип представляет класс
- **IsEnum** возвращает true, если тип является перечислением
- **IsInterface** возвращает true, если тип представляет интерфейс

Что можно получить?

Фильтры **BindingFlags**:

- **DeclaredOnly**: получает только методы непосредственно данного класса, унаследованные методы не извлекаются
- **Instance**: получает только методы экземпляра
- **NonPublic**: извлекает не публичные методы
- **Public**: получает только публичные методы
- **Static**: получает только статические методы

Динамическая загрузка сборок

Assembly.LoadFrom() и **Assembly.Load()**.

```
static void Main(string[] args)
{
    Assembly asm = Assembly.LoadFrom("MyApp.dll");

    Console.WriteLine(asm.FullName);
    // получаем все типы из сборки MyApp.dll
    Type[] types = asm.GetTypes();
    foreach(Type type in types)
    {
        Console.WriteLine(type.Name);
    }
    Console.ReadLine();
}
```


Позднее связывание

```
Assembly asm = Assembly.LoadFrom("MyApp.dll");  
Type type = asm.GetType("MyApp.Program", throwOnError: false, ignoreCase: true);  
  
// создаем экземпляр класса MyApp.Program  
object obj = Activator.CreateInstance(type);  
  
// получаем метод по имени  
MethodInfo methodInfo = type.GetMethod("Sum");  
  
// вызываем метод и получаем результат  
object result = methodInfo.Invoke(obj, new object[] { 6, 100, 3 });  
Console.WriteLine(result);
```




DEMO



Проверка знаний

? Что такое рефлексия?

? Для чего нужна рефлексия на практике?

- плагины/расширения
- атрибуты
- сериализация – де

? Какие сведения можно получить с помощью рефлексии?

Ваши вопросы



Задача на рефлексию

Основное задание:

1. Написать сериализацию свойств или полей класса в строку
2. Проверить на классе: `class F { int i1, i2, i3, i4, i5; Get() => new F(){ i1 = 1, i2 = 2, i3 = 3, i4 = 4, i5 = 5 }; }`
3. Замерить время до и после вызова функции (для большей точности можно сериализацию сделать в цикле 100-100000 раз)
4. Вывести в консоль полученную строку и разницу времен
5. Отправить в чат полученное время с указанием среды разработки и количества итераций
6. Замерить время еще раз и вывести в консоль сколько потребовалось времени на вывод текста в консоль
7. Провести сериализацию с помощью каких-нибудь стандартных механизмов (например в JSON)
8. И тоже посчитать время и прислать результат сравнения
9. Написать десериализацию/загрузку данных из строки (ini/csv-файла) в экземпляр любого класса
10. Замерить время на десериализацию
11. Общий результат прислать в чат с преподавателем в системе в таком виде:

Сериализуемый класс: `class F { int i1, i2, i3, i4, i5;}`

код сериализации-десериализации: ...

количество замеров: 1000 итераций

мой рефлексен:


Время на сериализацию = 100 мс

Время на десериализацию = 100 мс

стандартный механизм (Newtonsoft.Json):

Время на сериализацию = 100 мс

Время на десериализацию = 100 мс



Заполните, пожалуйста,
опрос о занятии
<https://otus.ru/polls/111434/>



До новых встреч!
Приходите на следующие занятия

