反射

```
 using System.Reflection;

class Program

{

private static void Main(string[] args)

{

 int a = 12;
  Type type = a.GetType();
  Console.WriteLine(type);

  Type type1 = typeof(int);
  Console.WriteLine(type1);

  Type type2 = Type.GetType("System.Int32");
  Console.WriteLine(type2);


  Type T = typeof(Test);
  //获取类中的所有公共成员
  MemberInfo[] infos = T.GetMembers();
  for (int i = 0; i < infos.Length; i++)
  {
      Console.WriteLine(infos[i]);
  }
  Console.WriteLine("///////////////////");
  //获取所有的构造函数
  ConstructorInfo[] cons = T.GetConstructors();
  for (int i = 0; i < cons.Length; i++)
  {

      Console.WriteLine(cons[i]);
  }
  //获取其中一个构造函数 并执行
  //1-1 无参构造
  ConstructorInfo con = T.GetConstructor(new Type[0]);//获得无参无返回值构造函数
  con.Invoke(null);//执行无参构造时，没有参数传null；
  Test obj = con.Invoke(null) as Test;
  Console.WriteLine(obj.str);
```

```
//1-2 有参构造
ConstructorInfo con2 = T.GetConstructor(new Type[] { typeof(int) });
Test obj2 = con2.Invoke(new object[] { 2 }) as Test;
Console.WriteLine(obj2.j);

ConstructorInfo Con3 = T.GetConstructor(new Type[] { typeof(int), typeof(string) });
Test obj3 = Con3.Invoke(new object[] { 123, "123445687" }) as Test;
Console.WriteLine(obj3.str);
Console.WriteLine("--------------------");

#region 获取类的公共成员变量
//获取所有变量
FieldInfo[] fieldInfo = T.GetFields();
for (int i = 0; i < fieldInfo.Length; i++)
{
    Console.WriteLine(fieldInfo[i]);
}
//获取指定变量
FieldInfo fieldJ = T.GetField("j");
Console.WriteLine(fieldJ);
//通过反射获得和设置对象的值
Test test = new Test();
test.j = 66;
test.str = "hello";
//1-1 通过反射获得对象的某个值
Console.WriteLine(fieldJ.GetValue(test));
//1-2 通过反射设置对象的某个值
fieldJ.SetValue(test, 100);
Console.WriteLine(fieldJ.GetValue(test));
#endregion

Console.WriteLine("+++++++++++++++++++++");



#region 获取类的公共方法
Type strTepy = typeof(string);
MethodInfo[] methodInfo = T.GetMethods();
for (int i = 0; i < methodInfo.Length; i++)
{
    Console.WriteLine(methodInfo[i]);
}
//1 如果出现重载，用Type数组表示参数类型  Substring是string中一个分割字符串的方法
MethodInfo substr = strTepy.GetMethod("Substring", new Type[] { typeof(int), typeof(int) });


//2调用该方法  如果时静态方法，第一个参数传null，
string str = "hello, world";
```

```
    //第一个参数相当于 是那个对象执行的这个成员方法
    object resurt = substr.Invoke(str, new object[] { 7, 5 });
    Console.WriteLine(resurt);
    #endregion
}
```

}

```
class Test

{

private int i = 1;

public int j = 0;

public string str = "123";


public Test()
{

}
public Test(int i)
{
    this.i = i;
}
public Test(int i, string str) : this(i)
{
    this.str = str;
}
public void Speak()
{
    Console.WriteLine(i);
}
}
```

加载程序集

Assembly

//Activator 实例化反射

Type testType = typeof(Test);

//无参

```
Test testObj=  Activator.CreateInstance(testType)as Test;

Console.WriteLine(testObj.str);

//有参

Test testobj1=Activator.CreateInstance(testType,23) as Test;

Console.WriteLine(testobj1.j);

//LoadFrom 加载不同文文件夹下的程序集

Assembly assembly = Assembly.LoadFrom("//路径");
```