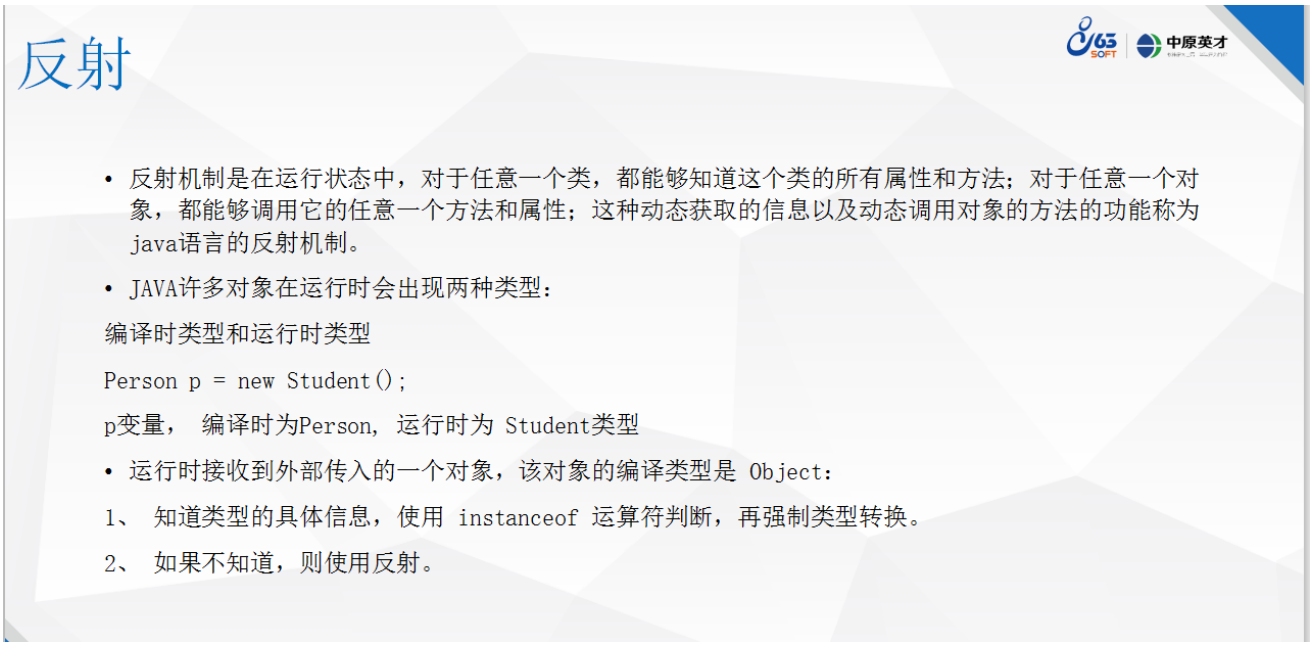
Java 反射

1、反射的定义




```
反射机制是在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意一个方法和属性；这种动态获取的信息以及动态调用对象的方法的功能称为java语言的反射机制。
JAVA许多对象在运行时会出现两种类型：
编译时类型和运行时类型
Person p = new Student();
p变量， 编译时为Person，运行时为 Student类型
运行时接收到外部传入的一个对象，该对象的编译类型是 Object：
1、 知道类型的具体信息，使用 instanceof 运算符判断，再强制类型转换。
2、 如果不知道，则使用反射。
```

2、通过反射获得类

```java
//1、已知类名，获得反射对象（Class对象）  类型.class
       Class<Student> clazz = Student.class;

       Class<Integer> clazz2 = int.class;

       //2、已知对象，获得反射对象（Class对象），对象.getClass();
       Student s = new Student();
       Class<? extends Student> clazz3 = s.getClass();

       //3、已知完整的包名.类名
      Class<Student> clazz4 = (Class<Student>) Class.forName("base.Student");

```

3、通过反射创建对象

```
        //1、通过反射获取到类名
        Class<Student> clazz = Student.class;
        //2、创建对象，使用的是无参构造函数,如果提供的类中没有无参构造函数，那么反射就会报错
        //等价于了 Student  s = new Student()
        Student s = clazz.newInstance();
        System.out.println(s);

        Class<Teacher> clazzTeacher = (Class<Teacher>)Class.forName("base.Teacher");
        Teacher t = clazzTeacher.newInstance();
        System.out.println(t);
        t.setId(1);
```

4、通过反射获取构造函数

```
        //获取到类,有了类，就可以去调用类中的一切事物
        Class<Person> clazz = Person.class;
        //通过类去创建对象,采用的是无参构造函数
        Person p = clazz.newInstance();
        //获取到无参构造函数，如果类中没有对应的构造函数就报错
        Constructor<Person> con =  clazz.getConstructor();
        //基本数据.class != 包装类.class
        //公共，私有的都可以获取
        //获得构造函数的声明
        Constructor<Person> con3 = clazz.getDeclaredConstructor
                (int.class,String.class,double.class);

        //给构造函数升级权限
        con3.setAccessible(true);//表示public 都可以访问

        //通过构造函数去创建对象
        // Perosn  p1 = new Person();
        Person p1 = con.newInstance();

        //Person p2 = new Person(1,"wangbowen",100);
        //传递参数，使用构造函数
        Person p2 = con3.newInstance(1,"王博文",100);

        System.out.println(p2.getId()+"\t"+p2.getName()+"\t"+p2.getScore());

```

5、通过反射获取属性

```
        //1.获取Class对象
        Class<Person> clazz = Person.class;
        //2.获取某一个属性
        //只能获取公共的属性
        Field f =  clazz.getField("age");
```

```
        //注意f并不是属性值，它是属性的声明
        System.out.println(f);
        //设置值
        //Person p = new Person(); p.age = 1;
        Object o = clazz.newInstance();
        //将1存入到o对象中的f（age）属性中
        f.set(o,1);
        //获的age属性对应的取值
        //取得是o对象.f(age)属性
        Object result = f.get(o);

        System.out.println(result);

        System.out.println("---------通过反射获得私有属性-------------");
        Class<Person> clazz2 = Person.class;
        //公有，私有都可以
        Field nameField = clazz2.getDeclaredField("name");
        System.out.println(nameField);

        //设置值
        //1.构建一个对象
        Object person = clazz2.newInstance();
        //2.提升访问等级到公共的
        nameField.setAccessible(true);
        //3.使用nameField对象给person对象中的属性赋值
        nameField.set(person,"wangyang");
        //4.获取值
        Object obj = nameField.get(person);
        //obj就是我们保存的wangyang
        System.out.println(obj);
```

5、通过反射获取所有属性和方法

5-1、所有属性

```
        Class<Person> clazz = Person.class;
        //获得Person对象中所有属性
        Field[] fs = clazz.getDeclaredFields();


        //循环数据
        for(Field f : fs)
        {
            System.out.println(f);
        }
```

5-2、所有方法

```
        Class<Person> clazz = Person.class;
        //获得所有方法声明，只获得本类自己写的方法
        //getMethods 只获得public修饰的方法，但是可以获得父类中的方法
        Method[] ms = clazz.getDeclaredMethods();

        for(Method m : ms)
        {
            System.out.println(m);
        }
```

6、通过反射获得一个方法并执行

```
        Class<Person> clazz = Person.class;

        Method m = clazz.getDeclaredMethod("hello");
        System.out.println(m);
        m.setAccessible(true);
        //Person p = new Person();p.hello();
        Object obj = clazz.newInstance();

        //调用方法
        m.invoke(obj);

        System.out.println("-------------有参数有返回值方法-----------------");
        Class<Person> clazz2 = Person.class;
        Method m2 = clazz2.getDeclaredMethod("hello",int.class,String.class);
        m2.setAccessible(true);
        Object o2 = clazz2.newInstance();
        Object result = m2.invoke(o2,1,"wangyang");

        System.out.println(result);
```

7、通过反射获取常用类

```
        //万事万物即可反射
        //自定义
        System.out.println("自定义类："+ Person.class);
        //数组
        int[] nums = new int[10];
        System.out.println("数组："+nums.getClass());
        //集合
        List<String> list = new ArrayList<>();
        System.out.println("List集合："+list.getClass());
        System.out.println(Class.forName("java.util.HashMap"));
        //注释
        System.out.println("注解："+Override.class);
        //接口
        System.out.println("接口："+Boy.class);
```