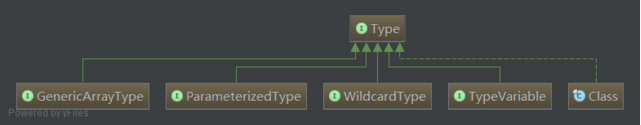
# s**pring-core组件详解——ResolvableType可解决的类型**

 ResolvableType，可解决的数据类型。它为java语言中的所有类型提供了相同的数据结构，其内部封装了一个java.lang.reflect.Type类型的对象

 在讲解这个数据结构之前，首先要了解一些预备知识，我们不妨思考如下2个问题：

1. 在java语法中，哪些元素可以代表一种类型？
2. 在java语法中，哪些元素具有类型？
3. 在jdk中，Type接口代表一种类型，所有的具体类型都需要实现这个接口。



语法中的类型可以分为五大类：组件类型为参数化类型或类型变量的数组、参数化类型、通配符表达式类型、类型变量以及所有定义的Class（每个类都是一个具体的类型）。除Class类以外的4个接口是jdk1.5以后出现的，因为单纯的Class类无法描述泛型信息。

回到之前提到的两个问题，现在第一个问题已经得到了答案。那么，java中哪些元素具有类型的属性呢？答案是：只有变量（或者说值，因为变量是值的载体）才具有类型。那么什么是变量呢？变量根据其所在位置不同，包括：成员变量、局部变量、方法形参以及方法返回值。

Class是一种类型，但它本身不具有类型的属性。

ResolvableType为所有的java类型提供了统一的数据结构以及API，换句话说，一个ResolvableType对象就对应着一种java类型。我们可以通过ResolvableType对象获取类型携带的信息（举例如下）：

1. getSuperType()：获取直接父类型
2. getInterfaces()：获取接口类型
3. getGeneric(int...)：获取类型携带的泛型类型
4. resolve()：Type对象到Class对象的转换

另外，ResolvableType的构造方法全部为私有的，我们不能直接new，只能使用其提供的静态方法进行类型获取：

1. forField(Field)：获取指定字段的类型
2. forMethodParameter(Method, int)：获取指定方法的指定形参的类型
3. forMethodReturnType(Method)：获取指定方法的返回值的类型
4. forClass(Class)：直接封装指定的类型

最后，总结一下ResolvableType的使用场景。它的使用场景是非常广泛的，在spring框架中需要反射的时候，为了不丢失泛型信息，通常都会使用ResolvableType封装所有的类型。