Руководство по наследуемым аннотациям Java

Vladimir Ovchinnikov[1]

Библиотека наследуемых аннотаций предназначена для Java-разработчиков, перед которыми стоит задача наследования аннотаций, объявленных для классов и интерфейсов, а также для их методов. Используется модель непротиворечивого наследования: наследование осуществляется только в том случае, если базовые классы и интерфейсы не содержат одной и той же аннотации для одного и того же элемента (класса, интерфейса или метода). Возможна перегрузка аннотаций в потомках. Данная библиотека распространяется с открытым исходным кодом, бесплатно и без ограничений на коммерческое использование.

[1] Обо всех обнаруженных ошибках или неточностях, а также пожелания и предложения сообщайте на адрес info@fusionsoft-online.com

ОГЛАВЛЕНИЕ

Решаемая проблема

Механизм аннотаций Java построен таким образом, что наследование аннотаций не происходит. Рассмотрим следующий пример,

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface A {}

@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface B {}

@A
public interface BaseInterface {
	@B
	public void method1();
}

public class BaseClass {
	@B
	public void method2(){}
}

public class Derived extends BaseClass implements BaseInterface{
	public void method1(){}
	public void method2(){}
}
```

В этом примере мы объявили два типа аннотации: аннотацию `@A` для типов, и аннотацию `@B` для методов. Использовали эти аннотации для описания некоторого базового класса и интерфейса. Создали производный класс – наследник базового класса и интерфейса.

В соответствии с правилами аннотирования Java, ни производный класс, ни его методы не наследуют тех аннотаций, которые были объявлены в базовых классе и интерфейсе. Поэтому следующий код вернет пустоту в обоих случаях:

```
	Derived.class.getMethod(
			"method1", new Class[0]).getAnnotation(B.class)

	Derived.class.getMethod(
			"method2", new Class[0]).getAnnotation(B.class)
```

Зачастую, в приложениях, анализирующих аннотации, необходимо, чтобы происходило наследование аннотаций, как для классов, так и для методов. Для этого разработана данная библиотека. Получить унаследованные аннотации в нашем примере можно следующим образом:

```
AnnotationManager.getAnnotatedClass(Derived.class)
.getAnnotation(A.class);

AnnotationManager.getAnnotatedClass(Derived.class)
.getAnnotatedMethod("method1", new Class[0])
.getAnnotation(B.class);

AnnotationManager.getAnnotatedClass(Derived.class)
.getAnnotatedMethod("method2", new Class[0])
.getAnnotation(B.class);
```

Работа с наследуемыми аннотациями

Объявление аннотаций

Для того, чтобы объявляемые аннотации были доступны в run-time, необходимо указать для них аннотацию `@Retention(RetentionPolicy.RUNTIME)`. Также может использоваться аннотация `@Target` для ограничения того, к чему применима объявляемая аннотация. Например, объявим аннотацию `@A`, применимую только к интерфейсам и классам:

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface A {}
```

Наследование аннотаций

За наследование аннотаций отвечает класс `AnnotationManager`. Сами аннотации при этом не копируются, `AnnotationManager` лишь формирует информацию о том, каким элементам кода какие аннотации соответствуют. Для того, чтобы получить информацию о наследовании для произвольного класса или интерфейса, необходимо вызвать статический метод `AnnotationManager.getAnnotatedClass`:

```
AnnotationManager.getAnnotatedClass(Derived.class)
```

Данный метод возвращает объект – аннотированный класс, – содержащий информацию о наследовании для указанного класса.

Для того, чтобы получить аннотации класса, собственные и унаследованные, необходимо воспользоваться одним из методов: `getAllAnnotations` или `getAnnotation`, например, так:

```
AnnotationManager.getAnnotatedClass(Derived.class)
.getAnnotation(A.class);
```

Аннотированный класс содержит аннотированные методы для всех методов исходного класса, содержащие информацию об унаследованных аннотациях. Для перехода к аннотированным методам класса существуют `getAnnotatedMethods` и `getAnnotatedMethod`. Так, например, по сигнатуре аннотированный метод можно получить следующим образом:

```
AnnotationManager.getAnnotatedClass(Derived.class)
.getAnnotatedMethod("method1", new Class[0])
```

Существует также возможность получить аннотированный метод по самому исходному методу.

Аннотации аннотированного метода, собственные и унаследованные, можно получить через `getAllAnnotations` или `getAnnotation`, например:

```
AnnotationManager.getAnnotatedClass(Derived.class)
.getAnnotatedMethod("method1", new Class[0])
.getAnnotation(B.class);
```

Также аннотированный метод позволяет перейти к тому аннотированному классу, которому он принадлежит (`getAnnotatedClass`), и к методу исходного класса (`getMethod`).

Конфликт аннотаций

Наследование аннотации не произойдет в случае обнаружения конфликта аннотаций. Под конфликтом понимается ситуация, когда в нескольких базовых классах и интерфейсах присутствует аннотация одного и того же вида, и при этом она отсутствует в производном классе или интерфейсе. Например, следующая ситуация считается конфликтом, так как оба базовых типа имеют аннотацию `@A`.

```
@A
public interface BaseInterface {...

@A
public class BaseClass {...

public class Derived extends BaseClass implements BaseInterface{...
```

В этом случае производный класс не будет содержать аннотацию `@A` вовсе.

Невозможность наследования аннотаций в случае коллизии обуславливается тем, что аннотации могут иметь параметры. В результате, возникает неопределенность, какие именно параметры должны быть унаследованы.

Данная проблема может быть решена явным объявлением аннотации `@A` для производного класса:

```
@A
public class Derived extends BaseClass implements BaseInterface{...
```

Аналогичная ситуация будет иметь место, если базовые классы и интерфейсы содержат несколько методов с одной и той же сигнатурой, аннотированных одним видом аннотации несколько раз, например:

```
public interface BaseInterface {
      @B
      public void method1();
}

public class BaseClass {
      @B
      public void method1(){}
}

public class Derived extends BaseClass implements BaseInterface{...
```

Решение данной проблемы аналогично решению проблемы при наследовании аннотаций классов – в производном классе необходимо объявить метод с той же сигнатурой и с правильной аннотацией для него:

```
public class Derived extends BaseClass implements BaseInterface{
      @B
      public void method1(){super.method1();}
...
```

Перегрузка аннотаций

Если в производном классе или интерфейсе присутствует аннотация на элементе, для которого в базовом классе или интерфейсе есть аннотация того же вида, то в качестве аннотации для этого элемента будет взята та, которая объявлена на нем явно. Таким образом, исходная аннотация будет перегружена.

```
@A
public interface BaseInterface {...

public class BaseClass {...

@A
public class Derived extends BaseClass implements BaseInterface{...
```

Перегрузка полезна для аннотаций с параметрами – позволяет изменить параметры аннотации, оставив саму аннотацию в силе.