解析注解

一、方法介绍

Java获取类上的注解有下面3个方法：

Class.getAnnotations()　　获取所有的注解，包括自己声明的以及继承的，返回Annotations[ ]

Class.getAnnotation(Class< A > annotationClass) 获取指定的注解，该注解可以是自己声明的，也可以是继承的

Class.getDeclaredAnnotations() 获取自己声明的注解

二、实现步骤

定义一个注解

```
@Documented
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE,ElementType.METHOD,ElementType.FIELD})
public @interface User {
      String name() default "user";
}
```

定义一个类使用注解

```
@User(name="qiu1")
public class Qiu {

   @User(name="qiu2")
   public String name;


   @User(name="qiu3")
   public String getName() {
      return name;
   }
   @User(name="qiu4")
```

```
    public void setName(String name) {
        this.name = name;
    }


    @Override
    public String toString() {
        return "Qiu{" +
                "name='" + name + '\'' +
                '}';
    }
}
```

1、获取类上的注解

```
public class testUser {
public static void main(String[] args) {
     Qiu qiu=new Qiu();
    if(Qiu.class.isAnnotationPresent(User.class)){
        User user=  Qiu.class.getAnnotation(User.class);
        System.out.println(user.name());
    }
}
```

2、获取方法上的注解

```
public class testUser {
    public static void main(String[] args) {
        Method[] methods=Qiu.class.getMethods();
        for (Method method : methods) {
            if(method.getName().equals("getName")){
                User user = method.getAnnotation(User.class);
                Annotation[] allAnnos2 = method.getAnnotations();
                  System.out.println(user.name());
             }
}
```

3、获取属性上的注解

```
 public class testUser {
   public static void main(String[] args) {
          Field[] fields = Qiu.class.getFields();
```

```
        for (Field field : fields) {
            Annotation[] allFAnnos= field.getAnnotations();
            User user= field.getAnnotation(User.class);
           // System.out.println(user);
            //System.out.println(allFAnnos);
}
```