## **Использование аннотации**

public class User {

public static enum Permission {

USER\_MANAGEMENT, CONTENT\_MANAGEMENT

}

private List<Permission> permissions;

public List<Permission> getPermissions() {

return new ArrayList<Permission>(permissions);

}

// ...

}

Создадим аннотацию, которую будем использовать для проверки прав доступа :

@Retention(RetentionPolicy.RUNTIME)

public @interface PermissionRequired {

User.Permission value();

}

Предположим, что у нас есть некоторое действие, право на выполнение которого мы хотим ограничить, например, *UserDeleteAction*. Для этого добавляем аннотацию на это действие следующим образом:

@PermissionRequired(User.Permission.USER\_MANAGEMENT)

public class UserDeleteAction {

public void invoke(User user) { /\* \*/ }

}

Теперь используя [reflection](http://java-online.ru/java-reflection.xhtml), можно принимать решение, разрешать или не разрешать выполнение определенного действия :

User user = ...;

Class<?> actionClass = ...;

PermissionRequired permissionRequired = actionClass.getAnnotation(PermissionRequired.class);

if (permissionRequired != null){

if (user != null && user.getPermissions().contains(permissionRequired.value())){

// выполнить действие

}

}

## **Пример анализатора аннотации**

Создадим класс анализатора, который будет определять аннотации и выполнять некоторые действия, связанные с аннотируемыми параметрами. Необходимо иметь в виду, что если используется более чем одна пользовательская аннотации, то целесообразно иметь отдельный анализатор для каждой аннотации.

Что должен делать анализатор? Он использует reflection для доступа к аннотируемым данным. Пример анализатора для класса @Test:

public class AnnotationAnalyzer {

public void parse(Class<?> clazz) throws Exception {

Method[] methods = clazz.getMethods();

int pass = 0;

int fail = 0;

for (Method method : methods) {

if (method.isAnnotationPresent(Test.class)) {

try {

// вызов аннотируемого метода

method.invoke(null);

pass++;

} catch (Exception e) {

fail++;

}

}

}

}

}

Сочетание использования аннотации и [reflection](http://java-online.ru/java-reflection.xhtml) позволяет выполнить определенную проверку и вызвать метод на исполнение через invoke. Анализатор готов к использованию. Для использования атрибутов аннотации расширим код.

public class AnnotationAnalyzer {

public void analyze(Class<?> clazz) throws Exception {

Method[] methods = clazz.getMethods();

int pass = 0;

int fail = 0;

for (Method method : methods) {

if (method.isAnnotationPresent(Test.class)) {

// Получаем доступ к атрибутам

Test test = method.getAnnotation(Test.class);

Class expected = test.expected();

try {

method.invoke(null);

pass++;

} catch (Exception e) {

if (Exception.class != expected) {

fail++;

} else {

pass++;

}

}

}

}

}

}

После получения доступа к атрибуту аннотации определяем ее значение. В нашем случае это значение типа Class, так как expected — это ожидаемая ошибка и мы будем получать exception.

## **Пример использования класса анализа аннотации:**

public class Demo

{

public static void main(String [] args)

{

AnnotationAnalyzer analyzer = new AnnotationAnalyzer();

analyzer.analyze(MyTest.class);

}

}

## 