Стандартная библиотека Java: Reflection API

Алексей Владыкин

28 октября 2013

1. Аннотации
2. Reflection API


АННОТАЦИИ?
risovach.ru

- Аннотации — это метаданные, сопровождающие исполняемый код
- В отличие от Javadoc, являются машиночитаемыми и могут быть доступны во время исполнения
- Примеры аннотаций:
  - `@Override`
  - `@Deprecated`
  - `@SuppressWarnings`

Что можно аннотировать

- Пакет
- Тип (класс, интерфейс, enum)
- Поле класса
- Метод, конструктор
- Параметр метода
- Локальная переменная

Создание аннотации

```
package ru.compscicenter.java2013;

import java.lang.annotation.*;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface Version {
    String value();
    String date() default "";
}
```

- Аннотация является особым видом интерфейса: `extends Annotation`

Использование аннотации

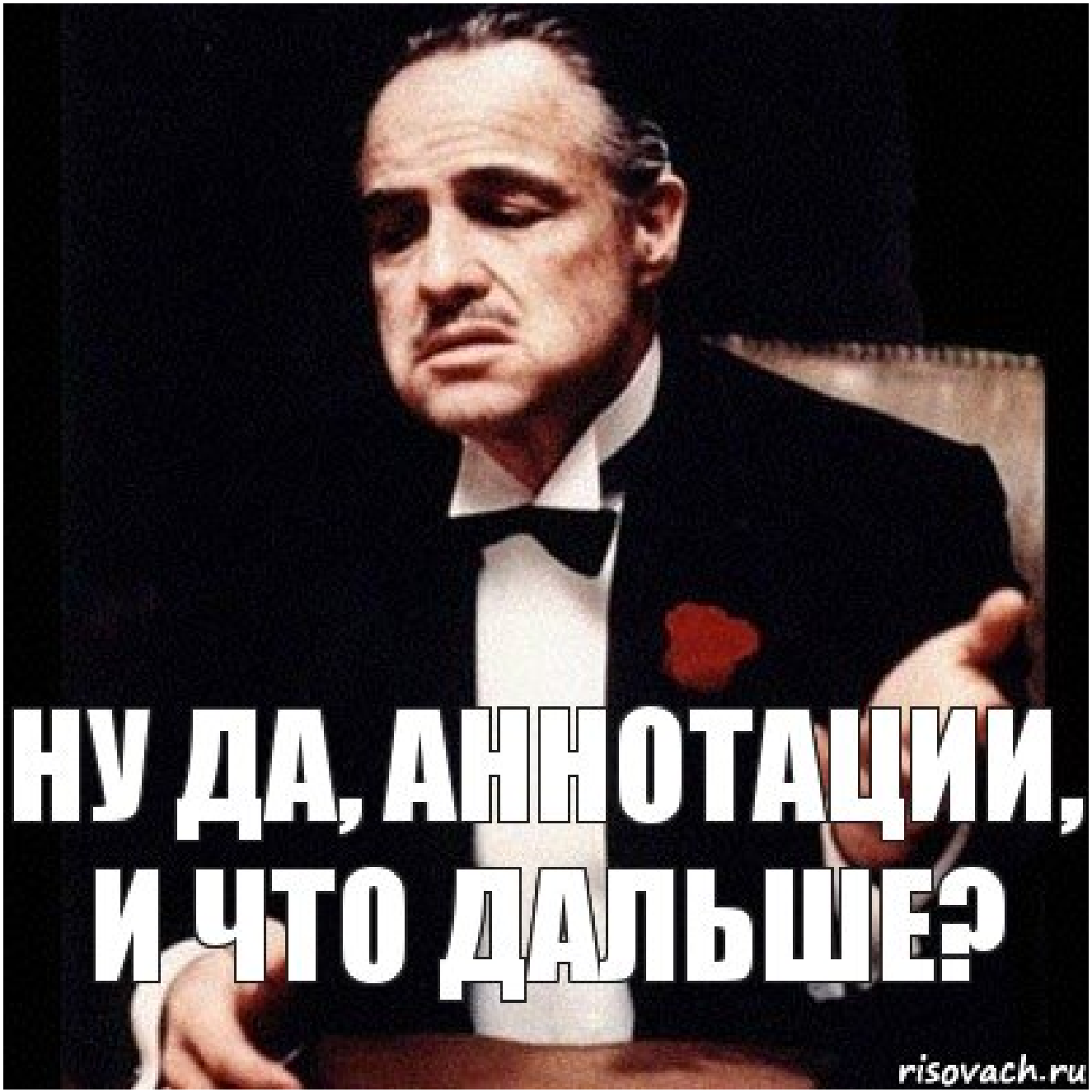
```
package ru.compscicenter.java2013;

@Version(value = "3.14", date = "01.01.2011")
public class Component {
    // ...
}
```

- Экземпляр аннотации является объектом,
  у которого можно вызвать методы `value()` и `date()`
- Нельзя создать экземпляр аннотации вызовом `new`
- Можно написать класс, реализующий интерфейс аннотации

Когда можно использовать аннотации

- Во время компиляции
  (Annotation Processing API)
- После компиляции, в class-файлах
  (например, статический анализ FindBugs)
- Во время исполнения программы
  (Reflection API)


НУ ДА, АННОТАЦИИ,
И ЧТО ДАЛЬШЕ?
risovach.ru

Пример использования аннотаций: JSR305

```
import javax.annotation.*;

public class JSR305Demo {

    @Nullable
    public Object getCurrentObject()
    { /*...*/}

    @Nonnull
    public Object newObject()
    { /*...*/ }
}
```

Пример использования аннотаций: JAXB

```
import javax.xml.bind.annotation.*;

@XmlRootElement(name = "Person")
@XmlType(propOrder = {"firstName", "lastName"})
public class Person {

    @XmlElement(name = "FirstName")
    public String firstName;

    @XmlElement(name = "LastName")
    public String lastName;
}
```

Пример использования аннотаций: JAXB

```
Person p = new Person();
p.firstName = "Ivan";
p.lastName = "Ivanov";

JAXBContext jaxbContext =
    JAXBContext.newInstance(Person.class);

Marshaller jaxbMarshaller =
    jaxbContext.createMarshaller();

jaxbMarshaller.marshal(p, System.out);
```

Пример использования аннотаций: JPA

```
import javax.persistence.*;

@Entity
public class Person {

    @Id @Column
    public long id;

    @Column(name = "first_name")
    public String firstName;

    @Column(name = "last_name")
    public String lastName;
}
```

Пример использования аннотаций: JPA

```
EntityManager entityManager = getEntityManager();

Person p = entityManager.find(Person.class, 333);
p.firstName = "Ivan";
p.lastName = "Ivanov";

entityManager.merge(p);
```


Сэр, вы осознаете, что вы кот?
risovach.ru

- Reflection API — программный интерфейс для получения информации об объектах и их классах во время исполнения программы
- Центральный класс — `java.lang.reflect.Class`
- Для каждого класса, загруженного в JVM, можно получить описывающий его экземпляр класса `Class`

Возможности Reflection API

- Получение списка конструкторов, методов и полей класса
- Создание экземпляров класса
- Вызов методов и чтение/запись полей, в том числе закрытых
- Но нельзя получить содержимое метода

Как получить Class

- Получение класса по объекту:
```
Class c1 = object.getClass();
```
- Получение класса через литерал:
```
Class c2 = String[].class;
```
- Загрузка класса по имени:
```
Class c3 = Class.forName("java.lang.Integer");
```

Как загрузить класс с диска

```
URL jarFileURL = new URL("file://electro.jar");

ClassLoader classLoader = new URLClassLoader(
        new URL[] {jarFileURL});

Class c4 = classLoader.loadClass("ElectroSolver");
```

Как создать класс на лету

```
public class MyClassLoader extends ClassLoader {

    protected Class<?> findClass(String name) {
        byte[] b = getClassData(name);
        return defineClass(name, b, 0, b.length);
    }

    private byte[] getClassData(String name) {
        // skip
    }
}
```

Имя класса

	int[]	Object[]	Foo.Bar
getName()	[I	Ljava.lang.Object;	Foo$Bar
getCanonicalName()	int[]	java.lang.Object[]	Foo.Bar
getSimpleName()	int[]	Object[]	Bar

Типы классов

- `boolean isPrimitive()`
- `boolean isInterface()`
- `boolean isAnnotation()`
- `Class getSuperclass()`
- `Class[] getInterfaces()`

Специфика массивов

```
if (clazz.isArray()) {
    System.out.println(
            "Array of " + c.getComponentType());
}
```

Специфика enum

```
if (clazz.isEnum()) {
    System.out.println("Enum of:");
    for (Object e : clazz.getEnumConstants()) {
        System.out.println(e);
    }
}
```

Конструкторы

- Открытые конструкторы:
  `Constructor getConstructor(Class... types)`
  `Constructor[] getConstructors()`

- Все конструкторы:
  `Constructor getDeclaredConstructor(Class... types)`
  `Constructor[] getDeclaredConstructors()`

Вызов конструктора

```
Constructor constructor =
        clazz.getConstructor(String.class);

Object instance =
        constructor.newInstance("Hello World!");
```

Методы

- Открытые методы, в том числе унаследованные:
  `Method getMethod(String name, Class... types)`
  `Method[] getMethods()`

- Все методы, но только из текущего класса:
  `Method getDeclaredMethod(String name, Class... types)`
  `Method[] getDeclaredMethods()`

Вызов метода

```
Method method =
        clazz.getMethod("doSomething", int.class);

Object result =
        method.invoke(instance, 42);
```

Поля

- Открытые поля, в том числе унаследованные:
  `Field getField(String name)`
  `Field[] getFields()`

- Все поля, но только из текущего класса:
  `Field getDeclaredField(String name)`
  `Field[] getDeclaredFields()`

Чтение/запись поля

```
Field field = clazz.getDeclaredField("x");
field.setAccessible(true);

Object value = field.get(instance);

field.set(instance, null);
```

Аннотации

```
Version version =
        clazz.getAnnotation(Version.class);

String versionNumber = version.value();
String versionDate = version.date();
```

Что сегодня узнали

- Что такое аннотации в Java и как их можно использовать
- Что такое Reflection API и какие возможности он предоставляет