01_注解介绍

- A.定义
  - 注解annotation是Java语言中用于描述类,成员变量,构造方法,成员方法,方法参数及包声明的特殊的修饰符.用于描述这些信息的元数据.例如@Override用于描述一个方法是在子类中重写的方法
- B.特点
  - 是JDK5.0之后引入的特性.注解是以”@注解名”在代码中存在的
- C.作用
  - 跟踪代码依赖性
  - 执行编译时格式检查
  - 代替已有的配置文件

02_JDK内置注解

- JDK内置注解,JDK1.5之后内置了三个系统注解
- A.@Override
  - 标记在成员方法上,用于标识当前方法是重写父类方法,编译器在对该方法进行编译时会检查是否符合重写规则,如果不符合,编译报错
- B.@Deprecated
  - 用于标记当前类、成员变量、成员方法或者构造方法过时 如果开发者调用了被标记为过时的方法，编译器在编译期进行警告
- C.@SuppressWarnings
  - 可放置在类和方法上,该注解的作用是阻止编译器发出某些警告信息,该注解为单值注解,只有一个value参数,该参数为字符串数组类型,参数值常用的有
    - unchecked 未检查的转化，如集合没有指定类型还添加元素
    - unused 未使用的变量
    - resource 有泛型未指定类型
    - path 在类路径，原文件路径中有不存在的路径
    - deprecation 使用了某些不赞成使用的类和方法
    - fallthrough switch语句执行到底没有break关键字
    - rawtypes 没有写泛型,比如: List list = new ArrayList();
    - all 全部类型的警告

03_注解分类

- 根据注解的参数个数分为三类:

- 标记注解:没有参数的注解,仅用自身的存在与否为程序提供信息,如@Override 注解,该注解没有参数,用于表示当前方法为重写方法
- 单值注解:只有一个参数的注解,如果该参数的名字为value,那么可以省略参数名,如@SuppressWarnings(“all”),可以简写为@SuppressWarnings(“all”).
- 完整注解:有多个参数的注解

04_元注解

- A.自定义注解:
  - 格式 @interface 注解名
- B.定义
  - 用户负责描述其他注解的注解
  - 在自定义注解时,需要指定注解可以应用在哪些位置(如:方法,参数,类上等)以及自定义注解的停留策略等信息,那么就可以通过元注解去描述这些信息了.
- C.分类
  - JDK1.5中定义了四个元注解,常用的两个注解，如下:
  - @Target
    - 作用：用于描述当前定义的注解可以应用的范围 该注解仅有一个属性value，该属性值为ElementType数组类型 ElementType为枚举类型，枚举值和作用说明如下：
    - TYPE 当前定义的注解可以应用在类、接口和枚举的类型定义部分
    - FILED 当前定义的注解可以应用在成员变量上
    - METHOD 当前定义的注解可以应用在成员方法上
    - PARAMETER 当前定义的注解可以应用在方法参数上
    - CONSTRUCTOR 当前定义的注解可以应用在构造方法上
    - LOCAL_VARIABLE 当前定义的注解可以应用在局部变量上
    - ANNOTATION_TYPE 当前定义的注解可以应用在注解类型上
    - PACKAGE 当前定义的注解可以应用在包定义语句上
  - @Retention
    - 作用:用于描述当前定义的注解可以保留的时间长短 该注解只有一个value参数,参数类型为RetentionPolicy. RetentionPolicy类型为枚举类型
    - SOURCE 当前定义的注解仅仅停留在源码中，编译时即除去
    - CLASS 当前定义的注解保留到编译后的字节码中,运行时无法获取注解信息
    - RUNTIME 当前定义的注解可以保留到运行时，通过反射机制可以获取注解信息

05_自定义注解案例

- A.需求:
  - 可以使添加有自定义注解的方法执行
  - 在类的某些方法上添加@MyTest的注解.
  - 可以使加有@MyTest的注解的方法执行.
- B.实现

```
public class MyTestCore {

    public static void main(String[] args) throws ClassNotFoundException,
IllegalArgumentException, IllegalAccessException, InvocationTargetException,
InstantiationException {
        /**
         * 获得类的Class对象.
         * 通过Class找出里面的所有的方法.
         * 遍历所有的方法.
         * 需要在每个方法上查找是否有@MyTest注解.
         *  * 如果有该注解:
         *      * 执行该方法.
         */
        // 获得类的Class对象.
        Class clazz = Class.forName("cn.qzw.annotation.demo3.AnnotationDemo1");
        // 获得类中的所有的方法.
        Method[] methods = clazz.getMethods();
        // 遍历所有的方法:
        for (Method method : methods) {
            // 判断方法上是否有某个注解:
            boolean flag = method.isAnnotationPresent(MyTest.class);
            // System.out.println(method.getName()+"   "+flag);
            if(flag == true){
                // 执行该方法:
                method.invoke(clazz.newInstance(), null);
            }
        }
    }

}
```

06_自定义注解之改变JDBC工具类

- A.自定义注解:

```
@Retention(RetentionPolicy.RUNTIME)
@Target(value=ElementType.METHOD)
public @interface JDBCInfo {
    String driverClass() default "com.mysql.jdbc.Driver";
    String url() default "jdbc:mysql:///web024";
    String username() default "root";
    String password() default "123";
}
```

- B.定义JDBC的工具类:

```
public class JDBCUtils {

    /**
     * 工具类中获得数据库连接的方法:
     * @return
```

```
     * @throws ClassNotFoundException
     */
    @JDBCInfo(password="1234")
    public static Connection getConnection() throws Exception{
        /**
         * 获得当前类上的getConnection方法.
         * 获得该方法上的@JDBCInfo这个注解.
         * 获得这个注解中的属性的值.
         * 使用这些值为下面参数设置值.
         */
        // 获得JDBCUtils的类的Class对象.
        Class clazz = JDBCUtils.class;
        // 获得getConnection方法:
        Method method = clazz.getMethod("getConnection", null);
        // 获得方法上的注解:
        JDBCInfo jdbcInfo = method.getAnnotation(JDBCInfo.class);
        // 获得注解中的属性的值:
        String driverClass = jdbcInfo.driverClass();
        String url = jdbcInfo.url();
        String username = jdbcInfo.username();
        String password = jdbcInfo.password();
        // 加载驱动:
        Class.forName(driverClass);
        // 获得连接:
        Connection conn = DriverManager.getConnection(url, username, password);
        return conn;
    }
}
```