**Topic One：什么是注解**

注意：注释是comment；

**Topic Two：几种内置注解**

这个方法的上面就有这个  他的意思是不建议使用；

注意：直接使用all可以压制全部，你也可以对应的研制；

代码如下：

1. package Test;
3. import java.util.ArrayList;
4. import java.util.Date;
5. import java.util.List;
7. /\*\*
8. \*
9. \* 测试一下Annotation的作用
10. \*
11. \* @author Wang
12. \*
13. \*/
15. public class Demo01 {
17. @Override //重写
18. public String toString(){
19. return "";
20. }
22. @Deprecated //不建议使用的方法
23. public static int add(int a ,int b){
24. return a+b;
25. }
27. @SuppressWarnings("all")//压制警告的作用
28. public static void test(){
29. List L1 = new ArrayList();
30. }
32. @SuppressWarnings("all")
33. public static void main(String[] args){
34. Date a = new Date();
35. System.out.println(add(3,4));//我们可以看见我们在使用不建议使用的方法的时候 方法的上面会有一个横线；
36. }
37. }

**Topic Three：自定义注解**

我们在定义注解的时候需要先定义元注解：

元注解的作用就是说明注解的使用范围等等  的解释；

Target：来描述注解的使用范围：

元注解Retention：

自己定义的一个注解：

1. package Test;
3. import java.lang.annotation.ElementType;
4. import java.lang.annotation.Retention;
5. import java.lang.annotation.RetentionPolicy;
6. import java.lang.annotation.Target;
8. /\*\*
9. \*
10. \* 自定义一个自己的注解
11. \*
12. \* @author wangtong
13. \*
14. \*/

17. @Target(value={ElementType.METHOD,ElementType.TYPE})
18. //这一行的作用是 规定这个注解可以用于方法和Type（Type的类，接口，枚举等等）的前面
19. @Retention(RetentionPolicy.RUNTIME)
20. //这个元注解的作用就是使我们写的那个注解 保存为运行时有效
22. public @interface MyAnnotation {
23. //我们可以给注解添加参数列表，但是如果我们在添加参数后如果不写默认值的话
24. //那么我们在用注解的时候就要添加上去
25. String studentName() default "";
26. int age() default 0;
27. int id() default -1; //String indexOf("abc") -1
28. // -1的作用在这里是不存在
29. String[] schools() default {"清华大学","北京大学"};
30. }

测试自己写的这个注解：

1. package Test;
2. /\*\*
3. \*
4. \* 测试自己写的注解
5. \*
6. \* @author wangtong
7. \*
8. \*/
10. @MyAnnotation
11. //可以修饰类 这里没有对注解的参数进行复制 那这里使用的就是默认值
12. public class Demo02 {
13. //可以修饰方法 这里我们自己就给参数就行赋值了
14. @MyAnnotation(age=19,studentName="啊哈",id=1,
15. schools={"北京大学","南阳理工"})
16. public void test(){
17. }
18. }

**Unit 2： 反射机制读取注解**

在这里我们可以看出注解的真正的作用；

我们通过注解可以把对应的Sql的语句跟写进去；

然后我们根据反射来把注解给读出来这样我们我可以得到Sql语句进而得到那个表格；

说白了就我们Java和数据库的结合；

Topic 1：我们来完成尚学堂的这个作业

step 1：先写注解

我们可以明显的看出我们应该写一个Table的注解  来存放表的名字

然后我们应该写一个属性的注解

代码如下：

1. package Test;
3. import java.lang.annotation.ElementType;
4. import java.lang.annotation.Retention;
5. import java.lang.annotation.RetentionPolicy;
6. import java.lang.annotation.Target;
8. /\*\*
9. \*
10. \* 表的注解，这里只有一个表的名字
11. \*
12. \* @author Wang
13. \*
14. \*/
16. @Target(value={ElementType.METHOD,ElementType.TYPE})
17. //这一行的作用是 规定这个注解可以用于方法和Type（Type的类，接口，枚举等等）的前面
18. @Retention(RetentionPolicy.RUNTIME)
19. //这个元注解的作用就是使我们写的那个注解 保存为运行时有效
20. public @interface Table {
21. String value();
22. //当注解只有一个参数的时候我们把这个参数命名为value
23. }

1. package Test;
3. import java.lang.annotation.ElementType;
4. import java.lang.annotation.Retention;
5. import java.lang.annotation.RetentionPolicy;
6. import java.lang.annotation.Target;
8. /\*\*
9. \*
10. \* 属性的注解
11. \*
12. \* @author wangtong
13. \*
14. \*/
16. @Target(value={ElementType.METHOD,ElementType.TYPE})
17. //这一行的作用是 规定这个注解可以用于方法和Type（Type的类，接口，枚举等等）的前面
18. @Retention(RetentionPolicy.RUNTIME)
19. //这个元注解的作用就是使我们写的那个注解 保存为运行时有效
20. public @interface Filed {
21. String columnName();//列名
22. String type(); //类型
23. int length(); //长度
24. }

step 2 :  我们利用类来存放表的信息，利用注解来存放来转化为数据库语言的信息；

1. package Test;
2. /\*\*
3. \*
4. \* student类来存放信息；
5. \* 注解来存放数据库语言的信息
6. \*
7. \* @author wangtong
8. \*
9. \*/
11. @Table("tb\_student")
12. public class Student {
14. @Filed(columnName = "id" , type = "int" , length = 10)
15. private int id;
16. //这里我们带上注解 就可以很容易的利用反射来读取注解的内容 便于转化为数据库语言
17. @Filed(columnName = "sname" , type = "varchar" , length = 10)
18. private String sname;
19. @Filed(columnName = "age" , type = "int" , length = 3)
20. private int age;
22. public Student(int id, String sname, int age) {
23. super();
24. this.id = id;
25. this.sname = sname;
26. this.age = age;
27. }
29. public int getId() {
30. return id;
31. }
33. public void setId(int id) {
34. this.id = id;
35. }
37. public String getSname() {
38. return sname;
39. }
41. public void setSname(String sname) {
42. this.sname = sname;
43. }
45. public int getAge() {
46. return age;
47. }
49. public void setAge(int age) {
50. this.age = age;
51. }

54. }

step3 ： 利用反射来把注解里面的内容给转化成信息（用于数据库的编写）

1. package Test;
3. import java.lang.annotation.Annotation;
4. import java.lang.reflect.Field;
6. /\*\*
7. \*
8. \* 我们利用反射来读取信息
9. \*
10. \* @author Wang
11. \*
12. \*/
14. public class Demo03 {
16. public static void main(String[] args){
18. try {
19. //我们这里读取student的信息 注意这里关联的时候 只要包名和类名就好了
20. Class classInfo = Class.forName("Test.Student");
22. //获得类的所有有效注解
23. Annotation[] annotations=classInfo.getAnnotations();
24. //用增强for 遍历打印
25. for (Annotation a : annotations)
26. System.out.println(a);
28. //获得类的指定的注解
29. //关联了Table这个注解类
30. Table t = (Table) classInfo.getAnnotation(Table.class);
31. System.out.println(t.value());
33. //获得类的属性的注解
34. Field f = classInfo.getDeclaredField("sname");
35. MyField Field1 = f.getAnnotation(MyField.class);
36. System.out.println(Field1.columnName()+"--"+Field1.type()+"--"+Field1.length());
37. //根据获得的表名、字段的信息，拼出DDL语句，然后，使用JDBC执行这个SQL，
38. //在数据库中生成相关的表
40. } catch (Exception e) {
41. // TODO Auto-generated catch block
42. e.printStackTrace();
43. }
44. }
45. }