String类的常用方法

构造方法：

public String (byte[] bytes)

public String (char[] value)

普通方法：

public **char charAt(int index)**--->返回第index个字符

public **boolean contacins(CharSequence s)** --->当且仅当包含指定char返回true

public **int length()**--->返回字符串长度

public **int indexOf(String str)**--->返回出现str的第一个位置

public **boolean equals(Object obj)** --->将此字符串与指定对象比较

public **boolean equalsIgnoreCase(String str)** --->与str比较是否相等（忽略大小写）

public **String replace(char oldChar,char newChar)** --->用newChar替换oldChar

public **boolean startsWith(String str)** --->是否以str开头

public **boolean endsWith(String str)** --->是否以str结尾

public **String toUpperCase()**--->返回大写形式

public **String toLowerCase()**--->返回小写形式

public **String subString(int i)** --->返回从i开始到结尾的子字符串

public **String substring(int i ,int j)** --->返回从i到j的子字符串

public **String trim()**--->去掉开头和结尾空格

public **char[] toCharArray**--->将字符串转换为一个新的字符数组

## reflect反射

1. 获取类的class对象
2. Class.forName(“包名+类名”);--->常用
3. 类.class();
4. 对象.getClass();
5. 获得构造器

getDeclaredConstrustors()--->返回本类中所有的构造方法

getDecleardConstructor([Class](mk:@MSITStore:C:\\Users\\yuanjie\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/lang/Class.html" \o "java.lang 中的类)<?>...parameterTypes) --->返回本类中指定的构造方法

getConstructors()--->返回所有公开的构造方法

getConstructor([Class](mk:@MSITStore:C:\\Users\\yuanjie\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/lang/Class.html" \o "java.lang 中的类)<?>... parameterTypes)--->返回指定的构造方法

1. 实例化对象

NewInstance()

注意：newInstance()是调用空构造，如果空构造不存在，会出现异常。由此可知，使用其他构造器创建对象比较麻烦，使用空构造特别简单。确保空构造存在。

私有构造器实例化对象时必须打开权限

1. 接口与父类

getInterfaces()--->确定此对象所表示的类或接口实现的接口

getSuperClass()--->返回此class所表示的实体（类、接口、基本类型或void）的超类class

1. 修饰符

getModifiers()--->返回修饰符

1. 属性

getDeclaredFields()--->返回本类中所有属性的一个Field数组

getFields()--->返回本类中所有公开属性的一个Field数组

1. 方法

getDeclaredMethod(方法名，参数)---> 获取某个方法，可以是私有

getDeclaredMethods()--->获取所有方法（本类中，包括私有）

getMethod(方法名，参数) --->获取某个方法，只能是公开的

getMethods()--->获取所有公共方法（本类，包括父类的）

1. 数组

Array

1. 创建数组

Object obj = Array.newInstance(int.class,5)

1. 获取大小

If(obj.getClass().isArray()){ 3) 判断是否为数组

System.out.println(Array.getLength(obj));

4) 设置值

Array.set(obj,0,100);

5) 获取值

System.out.println(Array.get(obj,0));

}

1. 类加载器

Bootstrap ClassLoader 此加载器采用c++编写，一般开发中很少见。

Extension ClassLoader 用来进行扩展类的加载，一般对应的是 jre\lib\ext目录中的类AppClassLoader 加载 classpath指定的类，是最常用的加载器。同时也是java中默认的加载器。 了解即可

1. 反射的相关操作
2. 操作属性

//1、获取Class对象

Class<?> clz=Class.forName("com.shsxt.ref.simple.User");

//2、获取对象

User u=(User)clz.newInstance();

//3、设置属性

Field field=clz.getDeclaredField("uname");

field.setAccessible(true);//打开权限

field.set(u, "0523");

//4、获取此属性

System.out.println(field.get(u));

1. 调用方法
2. 方法调用

//1、获取Class对象

Class<?> clz=Class.forName("com.shsxt.ref.simple.User");

//2、获取对象

User u=(User)clz.newInstance();

//3、获取方法

Method m =clz.getMethod("coding", String.class,String.class);

//4、成员方法的调用

m.invoke(u, "反射","多个参数");

//若是静态方法，传递null即可 因为静态方法属性类，不属于对象 m=clz.getMethod("testStatic",int.class);

m.invoke(null, 100);//与对象无关

1. 操作setter与getter 访问器

"set" + name.substring(0, 1).toUpperCase() + name.substring(1);