**String 类常用方法**

1. 具体内容
2. 在实际的项目开发过程中，只要是我们的项目就一定会存在String类的定义。所以掌握String类的常用开发方法对于我们开发责者是非常重要的。

1. JavaDoc简介
2. 在以后进行开发的过程中肯定要大量的使用Java的Api文档（JavDoc）这个文档可以通过oracle在线浏览https://docs.oracle.com/javase/9/docs/api/overview-summary.html

1. 在JDK 1.9之前，所有的Java中的常用类库都会在JVM启动的时候进行全部的加载，这样会影响性能，而在1.9以后开始提供有模块化的设计，将一些程序放在不同的模块，在模块中包含大量的程序开发包，而如果想去看String 类的相关定义，则可以通过打开java.long这个包
2. 字符串与字符串数组
3. 在JDK1.9以前，所有的String 类都是用的字符串数组进行包装的处理，所以在String 类里面使用提供一个转换处理的方法

这些方法包含我们的构造方法与普通方法。

·str.charAt(int index) //获取指定索引位置上的字符

|  |
| --- |
| 范例:利用chartAt() 可以获取指定索引的字符  public class StringDemo{  public static void main (String args[]){  String str = "www.mld.com";  System.out.println(str.charAt(4)); // 输出 m  }  } |

|  |
| --- |
| 范例:实现字符串与字符数组之间的转换  public class StringDemo{  public static void main (String args []){  String str = "helloword";  char [] result = str.toCharArray(); //将字符串转换为字符数组  for(int x = 0; x < result.length ; x++){  result[x] -=32; //转换为大写字符  }  //将字符数组转换为字符串  String NewStr = new String(result);  System.out.println(NewStr);//输出大写的HELLOWORD  System.out.println(new String (result , 0 ,5)) //HELLO 将部分字符数组转换为字符串  }  } |

|  |
| --- |
| 范例:(验证将某个字符串是为全部为数字组成)  public class StringDemo{  public static void main(String args[]){  String str = "123A456" ;  System.out.println(isNumber(str)?"是由数字组成" : "不是由数字组成");  }  public static boolean isNumber(Stirng str){  char [] result = str.toCharArray(); //将字符串变为字符数组  for(int x = 0; x < result.length ; x ++){  if(result[x] < "0" || result > "9"){  return false;  }  }  return true;  }  }    注意:在实际开发中 我们处理中文字符时往往使用char 因为char包含中文数据。 |

1. 字符串与字节数组
2. 字符串与字节数组之间也可以实现转换操作，但是当进行字符串与字节数组操作时，其主要目的是为了进行二进制的数据传输，或者进行编码转换。

|  |  |
| --- | --- |
| public String(byte[] bytes); | //构造方法 将全部的字节数组变为字符串 |
| public String (bytes[] bytes , int offset,  int length) | //构造方法 将部分的字节数组变为字符串 |
| public byte[] getBytes() | //将字符串转换为字节数组 |
| \*(最重要的)\*  public byte[] getBytes​(String charsetName)  throws UnsupportedEncodingException | //编码转换 |

1. 字符串比较方法

|  |  |
| --- | --- |
| public boolean equals(String anObject); | //比较字符串内容 |
| Public boolean equalsIgnoreCase(String anotherString) | //不区分大小写比较字符串大小 |
| \*（很重要）\*  Public int compareTo(String anotherString) | //进行字符串大小比较 |
| Public int compareToIgnoreCase(String anotherString) | //不区分大小写进行字符串大小写比较 |

1. 字符串查找

从一个完整的字符串中查找一个子字符串的操作, 1.5版本之后

public boolean contains​(CharSequence s) //判断此 字符串是否存在

* 1. 1.5版本之前 （很重要）

public int indexOf​(String str) //从前向后查找字符串的第一个位置的索引，找不到返回 -1

该方法的目的是为了字符串位置索引的查询，用来确定一些字符串索引的位置）

public boolean startsWith​(String prefix) //判断是否以指定的字符串开头

//重载方法

public boolean startsWith​(String prefix,

int toffset) //从指定位置开始判断是否以指定的字符串开头

public boolean endsWith​(String suffix) //判断是否以指定的字符串结尾

1. 字符串替换

|  |  |
| --- | --- |
| public String replaceAll​(String regex,  String replacement) | //替换全部 |
| public String replaceFirst​(String regex,  String replacement) | //替换首个 |
| 范例:  public class StringDemo{  public static void main(){  String str = "helloword" ;  System.out.println(str.replaceAll​("l","X")); //hexxoword;  System.out.println(str.replaceAll​("l","X")); //hexloword;  }  } | |

1. 字符串拆分

|  |  |
| --- | --- |
| public String[] split​(String regex) | //全部拆分 |
| 范例:  public class StringDemo{  public static void main(){  String str = "hello word hello mld" ;  System.out.println(str.split​(" "));  }  } | |
| public String[] split​(String regex, int limit) | //拆到指定位置不拆了 |
| public class StringDemo{  public static void main(){  String str = "hello word hello mld" ;  String [] str2 = str.split​(" "，2)  for(int i = 0; i <str.length; i++){  System.out.println(str[i])  }  //输出：  hello  word hello mld  }  } | |

1. 字符串截取操作

public String substring​(int beginIndex) //从指定位置开始截取到结束

public String substring​(int beginIndex,

int endIndex) //截取指定范围内的字符串

|  |
| --- |
| 范例:  public class StringDemo{  public static void main(String args[]){  //字符串结构，"用户id-photot-姓名"，现要截取姓名  String str = "mld-photo-张三.jpg";  int beginIndex = str.indexOf("-",str.indexOf("photo")) +1;  Int endIndex = str.indexOf(".");  System.out.println(str.substring(beginIndex,endIndex)); //张三  }  }  注意:在实际开发中，通过这样的计算来获得相应的索引，是非常的常见的 |

1. 字符串的格式化

JDK1.5之后Java为了吸引更多的传统开发人员，Java提供了格式化数据的处理操作，类似于C语言之中的格式化输 出语句，可以利用我们的占位符实现我们数据的输出，如:字符串（%s） ，字符（%c）， 整数（%d），小数（%f） 等来描述

public static String format​(String format,

Object... args)) //格式化数组

public static String format​("姓名: %s 、年龄: %d 、成绩: %5.2f、",name ,age,score)

//姓名: 小王 、年龄: 18 、成绩: 97.58

1. 字符串的其它方法:

public String trim​() //取出字符串两边的空格

String toLowerCase​() //将字符串转为小写

public String toUpperCase​() //将字符串转为大写

|  |
| --- |
| **范例:利用各种方法的组合写一个首字母大写打方法**  class StringUilt{  public static String initcpa(String str){  if(str == null || "".equlas(str)){  return str ;  }  if(str.length() == 1){  str.toLowerCase();  }  return str.substring(0,1).toLowweCase()+str.substring(1) ;  }  } |