**Java String类**

字符串广泛应用在Java编程中，在Java中字符串属于对象，Java提供了String类来创建和操作字符串。

**创建字符串**

创建字符串最简单的方式如下:

String greeting = "Hello world!";

在代码中遇到字符串常量时，这里的值是"Hello world!"，编译器会使用该值创建一个String对象。

和其它对象一样，可以使用关键字和构造方法来创建String对象。

String类有11种构造方法，这些方法提供不同的参数来初始化字符串，比如提供一个字符数组参数:

public class StringDemo{

public static void main(String args[]){

char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.'};

String helloString = new String(helloArray);

System.out.println( helloString );

}

}

以上实例编译运行结果如下：

hello.

**注意:**String类是不可改变的，所以你一旦创建了String对象，那它的值就无法改变了。 如果需要对字符串做很多修改，那么应该选择使用[StringBuffer & StringBuilder 类](http://www.w3cschool.cc/java/java-stringbuffer.html)。

**字符串长度**

用于获取有关对象的信息的方法称为访问器方法。

String类的一个访问器方法是length()方法，它返回字符串对象包含的字符数。

下面的代码执行后，len变量等于17:

public class StringDemo {

public static void main(String args[]) {

String palindrome = "Dot saw I was Tod";

int len = palindrome.length();

System.out.println( "String Length is : " + len );

}

}

以上实例编译运行结果如下：

String Length is : 17

**连接字符串**

String类提供了连接两个字符串的方法：

string1.concat(string2);

返回string2连接string1的新字符串。也可以对字符串常量使用concat()方法，如：

"My name is ".concat("Zara");

更常用的是使用'+'操作符来连接字符串，如：

"Hello," + " world" + "!"

结果如下:

"Hello, world!"

下面是一个例子:

public class StringDemo {

public static void main(String args[]) {

String string1 = "saw I was ";

System.out.println("Dot " + string1 + "Tod");

}

}

以上实例编译运行结果如下：

Dot saw I was Tod

**创建格式化字符串**

我们知道输出格式化数字可以使用printf()和format()方法。String类使用静态方法format()返回一个String对象而不是PrintStream对象。

String类的静态方法format()能用来创建可复用的格式化字符串，而不仅仅是用于一次打印输出。如下所示：

System.out.printf("The value of the float variable is " +

"%f, while the value of the integer " +

"variable is %d, and the string " +

"is %s", floatVar, intVar, stringVar);

你也可以这样写

String fs;

fs = String.format("The value of the float variable is " +

"%f, while the value of the integer " +

"variable is %d, and the string " +

"is %s", floatVar, intVar, stringVar);

System.out.println(fs);

**String 方法**

下面是String类支持的方法，更多详细，参看Java API文档:

|  |  |
| --- | --- |
| **SN(序号)** | **方法描述** |
| 1 | char charAt(int index) 返回指定索引处的 char 值。 |
| 2 | int compareTo(Object o) 把这个字符串和另一个对象比较。 |
| 3 | int compareTo(String anotherString) 按字典顺序比较两个字符串。 |
| 4 | int compareToIgnoreCase(String str) 按字典顺序比较两个字符串，不考虑大小写。 |
| 5 | String concat(String str) 将指定字符串连接到此字符串的结尾。 |
| 6 | boolean contentEquals(StringBuffer sb) 当且仅当字符串与指定的StringButter有相同顺序的字符时候返回真。 |
| 7 | static String copyValueOf(char[] data) 返回指定数组中表示该字符序列的 String。 |
| 8 | static String copyValueOf(char[] data, int offset, int count) 返回指定数组中表示该字符序列的 String。 |
| 9 | boolean endsWith(String suffix) 测试此字符串是否以指定的后缀结束。 |
| 10 | boolean equals(Object anObject) 将此字符串与指定的对象比较。 |
| 11 | boolean equalsIgnoreCase(String anotherString) 将此 String 与另一个 String 比较，不考虑大小写。 |
| 12 | byte[] getBytes()  使用平台的默认字符集将此 String 编码为 byte 序列，并将结果存储到一个新的 byte 数组中。 |
| 13 | byte[] getBytes(String charsetName) 使用指定的字符集将此 String 编码为 byte 序列，并将结果存储到一个新的 byte 数组中。 |
| 14 | void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin) 将字符从此字符串复制到目标字符数组。 |
| 15 | int hashCode() 返回此字符串的哈希码。 |
| 16 | int indexOf(int ch) 返回指定字符在此字符串中第一次出现处的索引。 |
| 17 | int indexOf(int ch, int fromIndex) 返回在此字符串中第一次出现指定字符处的索引，从指定的索引开始搜索。 |
| 18 | int indexOf(String str)  返回指定子字符串在此字符串中第一次出现处的索引。 |
| 19 | int indexOf(String str, int fromIndex) 返回指定子字符串在此字符串中第一次出现处的索引，从指定的索引开始。 |
| 20 | String intern()  返回字符串对象的规范化表示形式。 |
| 21 | int lastIndexOf(int ch)  返回指定字符在此字符串中最后一次出现处的索引。 |
| 22 | int lastIndexOf(int ch, int fromIndex) 返回指定字符在此字符串中最后一次出现处的索引，从指定的索引处开始进行反向搜索。 |
| 23 | int lastIndexOf(String str) 返回指定子字符串在此字符串中最右边出现处的索引。 |
| 24 | int lastIndexOf(String str, int fromIndex)  返回指定子字符串在此字符串中最后一次出现处的索引，从指定的索引开始反向搜索。 |
| 25 | int length() 返回此字符串的长度。 |
| 26 | boolean matches(String regex) 告知此字符串是否匹配给定的正则表达式。 |
| 27 | boolean regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len) 测试两个字符串区域是否相等。 |
| 28 | boolean regionMatches(int toffset, String other, int ooffset, int len) 测试两个字符串区域是否相等。 |
| 29 | String replace(char oldChar, char newChar) 返回一个新的字符串，它是通过用 newChar 替换此字符串中出现的所有 oldChar 得到的。 |
| 30 | String replaceAll(String regex, String replacement 使用给定的 replacement 替换此字符串所有匹配给定的正则表达式的子字符串。 |
| 31 | String replaceFirst(String regex, String replacement)  使用给定的 replacement 替换此字符串匹配给定的正则表达式的第一个子字符串。 |
| 32 | String[] split(String regex) 根据给定正则表达式的匹配拆分此字符串。 |
| 33 | String[] split(String regex, int limit) 根据匹配给定的正则表达式来拆分此字符串。 |
| 34 | boolean startsWith(String prefix) 测试此字符串是否以指定的前缀开始。 |
| 35 | boolean startsWith(String prefix, int toffset) 测试此字符串从指定索引开始的子字符串是否以指定前缀开始。 |
| 36 | CharSequence subSequence(int beginIndex, int endIndex)  返回一个新的字符序列，它是此序列的一个子序列。 |
| 37 | String substring(int beginIndex) 返回一个新的字符串，它是此字符串的一个子字符串。 |
| 38 | String substring(int beginIndex, int endIndex) 返回一个新字符串，它是此字符串的一个子字符串。 |
| 39 | char[] toCharArray() 将此字符串转换为一个新的字符数组。 |
| 40 | String toLowerCase() 使用默认语言环境的规则将此 String 中的所有字符都转换为小写。 |
| 41 | String toLowerCase(Locale locale)  使用给定 Locale 的规则将此 String 中的所有字符都转换为小写。 |
| 42 | String toString()  返回此对象本身（它已经是一个字符串！）。 |
| 43 | String toUpperCase() 使用默认语言环境的规则将此 String 中的所有字符都转换为大写。 |
| 44 | String toUpperCase(Locale locale) 使用给定 Locale 的规则将此 String 中的所有字符都转换为大写。 |
| 45 | String trim() 返回字符串的副本，忽略前导空白和尾部空白。 |
| 46 | static String valueOf(primitive data type x) 返回给定data type类型x参数的字符串表示形式。 |