第 15 天常用 API

今日内容介绍

- Object
- String
- StringBuilder

第1章 Java 的 API 及 Object 类

在以前的学习过程中，我们都在学习对象基本特征、对象的使用以及对象的关系。接下来我们开始使用对象做事情，那么在使用对象做事情之前，我们要学习一些 API 中提供的常用对象。首先在学习 API 中的 Object 类之前,先来学习如何使用 API。

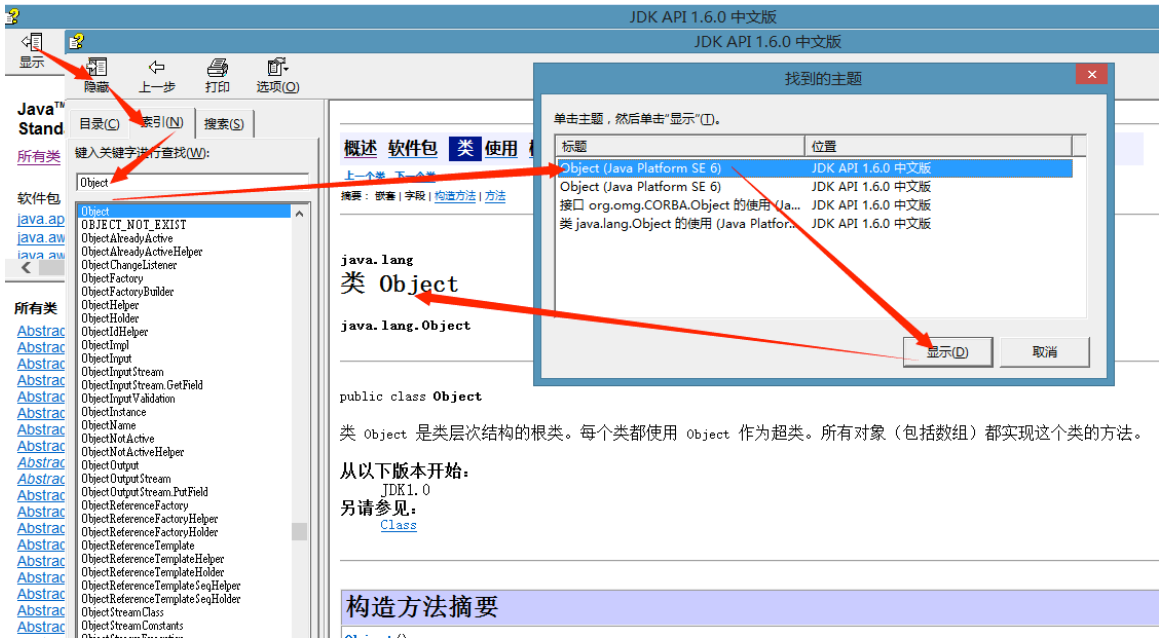
1.1Java 的 API

Java 的 API（API: Application(应用) Programming(程序) Interface(接口)）

Java API 就是 JDK 中提供给我们使用的类，这些类将底层的代码实现封装了起来，我们不需要关心这些类是如何实现的，只需要学习这些类如何使用即可。

在 JDK 安装目录下有个 src.zip 文件，这个文件解压缩后里面的内容是所有 Java 类的源文件。可以在其中查看相对应的类的源码。

我们在每次查看类中的方法时，都打开源代码进行查看，这种方式过于麻烦。其实，我们可以通过查帮助文档的方式，来了解 Java 提供的 API 如何使用。如下图操作：查找 Object 类



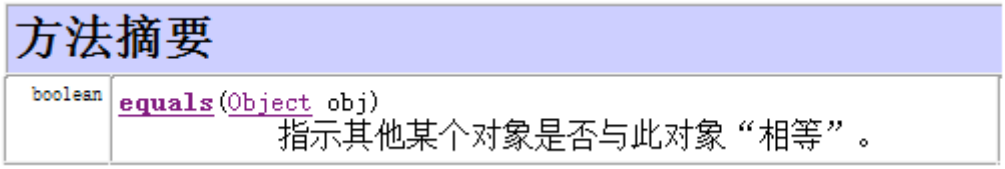
通过帮助文档中类与方法的介绍，我们就能够使用这个类了。

1.2 Object 类概述

Object 类是 Java 语言中的根类，即所有类的父类。它中描述的所有方法子类都可以使用。所有类在创建对象的时候，最终找的父类就是 Object。

在 Object 类众多方法中，我们先学习 equals 方法与 toString 方法，其他方法后面课程中会陆续学到。

1.3 equals 方法



equals 方法，用于比较两个对象是否相同，它其实就是使用两个对象的内存地址在比较。Object 类中的 equals 方法内部使用的就是==比较运算符。

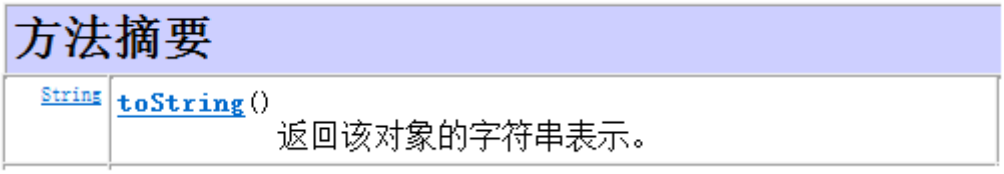
在开发中要比较两个对象是否相同，经常会根据对象中的属性值进行比较，也就是在开发经常需要子类重写 equals 方法根据对象的属性值进行比较。如下代码演示：

```
/*
    描述人这个类，并定义功能根据年龄判断是否是同龄人
    由于要根据指定类的属性进行比较，这时只要覆盖 Object 中的 equals 方法
    在方法体中根据类的属性值进行比较
 */
class Person extends Object{
    int age ;
```

```
    //复写父类的 equals 方法，实现自己的比较方式
    public boolean equals(Object obj) {
        //判断当前调用 equals 方法的对象和传递进来的对象是否是同一个
        if(this == obj){
            return true;
        }
        //判断传递进来的对象是否是 Person 类型
        if(!(obj instanceof Person)){
            return false;
        }
        //将 obj 向下转型为 Perosn 引用，访问其属性
        Person p = (Person)obj;
        return this.age == p.age;
    }
}
```

注意：在复写 Object 中的 equals 方法时，一定要注意 public boolean equals(Object obj)的参数是 Object 类型，在调用对象的属性时，一定要进行类型转换，在转换之前必须进行类型判断。

1.4toString 方法



toString 方法返回该对象的字符串表示，其实该字符串内容就是对象的类型+@+内存地址值。

由于 toString 方法返回的结果是内存地址，而在开发中，经常需要按照对象的属性得到相应的字符串表现形式，因此也需要重写它。

```
class Person extends Object{
    int age ;
    //根据 Person 类的属性重写 toString 方法
    public String toString() {
        return "Person [age=" + age + "]";
    }
}
```

第2章 String 类

2.1String 类的概述

查阅 API 中的 String 类的描述，发现 String 类代表字符串。Java 程序中的所有字符串字面值（如

"abc" ）都作为此类的实例实现。

```
//演示字符串
String str  = "itcast";
str = "传智播客";
```

继续查阅 API 发现说字符串是常量；它们的值在创建之后不能更改，这是什么意思呢？其实就是说一旦这个字符串确定了，那么就会在内存区域中就生成了这个字符串。字符串本身不能改变，但 str 变量中记录的地址值是可以改变的。

继续查 API 发现，字符串有大量的重载的构造方法。通过 String 类的构造方法可以完成字符串对象的创建，那么，通过使用双引号的方式创建对象与 new 的方式创建对象，有什么不同呢？看如下程序与图解：

```
String s3 = "abc";
String s4 = new String("abc");
System.out.println(s3==s4);//false
System.out.println(s3.equals(s4));//true,
//因为 String 重写了 equals 方法，建立了字符串自己的判断相同的依据（通过字符串对象中的字符来判断）
```

s3 和 s4 的创建方式有什么不同呢？

- s3 创建，在内存中只有一个对象。这个对象在字符串常量池中
- s4 创建，在内存中有两个对象。一个 new 的对象在堆中，一个字符串本身对象，在字符串常量池中

2.2String 类构造方法

构造方法是用来完成 String 对象的创建，下图中给出了一部分构造方法需要在 API 中找到，并能够使用下列构造方法创建对象。

构造方法摘要

String() 初始化一个新创建的 String 对象，使其表示一个空字符序列。
String(byte[] bytes) 通过使用平台的默认字符集解码指定的 byte 数组，构造一个新的 String。
String(byte[] bytes, int offset, int length) 通过使用平台的默认字符集解码指定的 byte 子数组，构造一个新的 String。
String(char[] value) 分配一个新的 String，使其表示字符数组参数中当前包含的字符序列。
String(char[] value, int offset, int count) 分配一个新的 String，它包含取自字符数组参数一个子数组的字符。
String(String original) 初始化一个新创建的 String 对象，使其表示一个与参数相同的字符序列

```
String s1 = new String(); //创建 String 对象，字符串中没有内容
```

```
    byte[] bys = new byte[]{97,98,99,100};
    String s2 = new String(bys); // 创建 String 对象，把数组元素作为字符串的内容
    String s3 = new String(bys, 1, 3); //创建 String 对象，把一部分数组元素作为字符串的内容，参数 offset 为数组元素的起始索引位置，参数 length 为要几个元素

    char[] chs = new char[]{'a','b','c','d','e'};
    String s4 = new String(chs); //创建 String 对象，把数组元素作为字符串的内容
    String s5 = new String(chs, 0, 3);//创建 String 对象，把一部分数组元素作为字符串的内容，参数 offset 为数组元素的起始索引位置，参数 count 为要几个元素

    String s6 = new String("abc"); //创建 String 对象，字符串内容为 abc
```

2.3String 类的方法查找

String 类中有很多的常用的方法，我们在学习一个类的时候，不要盲目的把所有的方法尝试去使用一遍，这时我们应该根据这个对象的特点分析这个对象应该具备那些功能，这样大家应用起来更方便。

字符串是一个对象，那么它的方法必然是围绕操作这个对象的数据而定义的。我们想想字符串中有哪些功能呢？

- 字符串中有多少个字符?

int	**length**() 返回此字符串的长度。

```
String str = "abcde";
int len = str.length();
System.out.println("len="+len);
```

- 获取部分字符串。

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。
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。

```
String str = "abcde";
String s1 = str.substring(1); //返回一个新字符串，内容为指定位置开始到字符串末尾的所有字符
String s2 = str.substring(2, 4);//返回一个新字符串，内容为指定位置开始到指定位置结束所有字符
System.out.println("str="+str);
System.out.println("s1="+s1);
System.out.println("s2="+s2);
```

2.4String 类中方法查找练习

前面给大家简单介绍了几个字符串中常用的方法，这个过程中主要让大家学会如何去查阅 API，

如何找到自己想用的方法。接下来我们来练习下字符串方法的查找。

- 字符串是否以指定字符串开头。结尾同理。

boolean	**startsWith**(String prefix) 测试此字符串是否以指定的前缀开始。
boolean	**endsWith**(String suffix) 测试此字符串是否以指定的后缀结束。

```
String str = "StringDemo.java";
boolean b1 = str.startsWith("Demo");//判断是否以给定字符串开头
boolean b2 = str.startsWith("String");
boolean b3 = str.endsWith("java");//判断是否以给定字符串结尾
```

- 字符串中是否包含另一个字符串。

boolean	**contains**(CharSequence s) 当且仅当此字符串包含指定的 char 值序列时，返回 true。
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。

```
String str = "abcde";
int index = str.indexOf("bcd"); //判断是否包含指定字符串，包含则返回第一次出现该字符串的索引，不包含则返回-1
boolean b2 = str.contains("bcd");//判断是否包含指定字符串，包含返回 true，不包含返回 false
```

- 将字符串转成一个字符数组。或者字节数组。

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。
char[]	**toCharArray**() 将此字符串转换为一个新的字符数组。

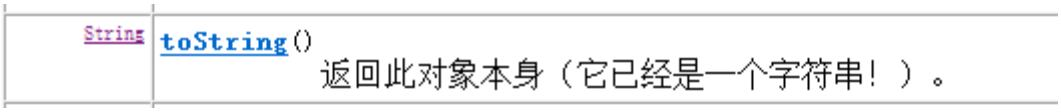
```
String str = "abcde";
char[] chs = str.toCharArray();
byte[] bytes = str.getBytes();
```

- 判断两个字符串中的内容是否相同

boolean	**equals**(Object anObject) 将此字符串与指定的对象比较。
boolean	**equalsIgnoreCase**(String anotherString) 将此 String 与另一个 String 比较，不考虑大小写。

```
String str = "abcde";
String str2 = "abcde";
String str3 = "hello";
boolean b1 = str.equals(str2);
boolean b2 = str.equals(str3);
```

- 获取该字符串对象中的内容




```
String str = new String("hello");
System.out.println( str.toString() );
System.out.pintln( str );
```

直接打印引用类型变量时，默认调用该类型进行重写后的 toString 方法

下面的需求所对应的方法，要求大家自己动手在 API 中查找，并进行方法使用。

- 判断该字符串的内容是否为空的字符串
- 获取给定的字符，在该字符串中第一次出现的位置
- 获取该字符串中指定位置上的字符
- 把该字符串转换成 小写字符串
- 把该字符串转换成 大写字符串
- 在该字符串中，将给定的旧字符，用新字符替换
- 在该字符串中， 将给定的旧字符串，用新字符串替换
- 去除字符串两端空格，中间的不会去除，返回一个新字符串

2.5String 类方法使用练习

- 题目一：获取指定字符串中，大写字母、小写字母、数字的个数。
- 思路：1.为了统计大写字母、小写字母、数字的个数。创建 3 个计数的变量。
  2.为了获取到字符串中的每个字符，进行字符串的遍历，得到每个字符。
  3.对得到的字符进行判断，如果该字符为大写字母，则大写字母个数+1；如果该字符为小写字母，则小写字母个数+1；如果该字符为数字，则数字个数+1。
  4.显示大写字母、小写字母、数字的个数
- 代码：

```
public static void method(String str){
    int bigCount = 0; //大写字母的个数
    int smallCount = 0; //小写字母的个数
    int numberCount = 0; //数字的个数
    for (int i=0; i < str.length(); i++) {
        char ch = str.charAt(i); //获取指定位置上的字符
        if (ch>='A' && ch<='Z') {
            bigCount++;
        } else if (ch>='a' && ch<='z') {
            smallCount++;
        } else if (ch>='0' && ch<='9') {
            numberCount++;
        }
    }
    System.out.println("大写字母个数："+bigCount);
    System.out.println("小写字母个数："+smallCount);
```

```
        System.out.println("数字个数："+numberCount);
    }
```

- 题目二：将字符串中，第一个字母转换成大写，其他字母转换成小写，并打印改变后的字符串。
- 思路：1.把字符串分为两个部分，第一部分为字符串中第一个字母，第二部分为剩下的字符串。
  2.把第一部分字符串转换成大写字母，把第二部分字符串转换成小写字母
  3.把两部分字符串连接在一起，得到一个完整的字符串
- 代码：

```
    public static String convert(String str){
        //获取第一部分为字符串
        String start = str.substring(0,1);
        //获取第二部分为字符串
        String end = str.substring(1);
        //把第一部分字符串转换成大写字母，把第二部分字符串转换成小写字母
        String big = start.toUpperCase();
        String small = end.toLowerCase();
        //把两部分字符串连接在一起，得到一个完整的字符串
        return big+small;
    }
```

- 题目三：查询大字符串中，出现指定小字符串的次数。如“hellojava,nihaojava,javazhenbang”中查询出现“java”的次数。
- 思路：1.在大串中，查找小串出现的位置，出现了就次数+1
  2.在上次小串出现位置的后面继续查找，需要更改大串的内容为上次未查询到的字符串。
  3.回到第一步，继续查找小串出现的位置，直到大串中查询不到小串为止
- 代码：

```
    public static int getCount(String big, String small){
        int count = 0; //出现小串的次数
        int index = -1;//出现小串的位置
        /*
            while 的循环条件三步骤：
            步骤一. big.indexOf(small) 获取小串在大串中出现的位置
            步骤二. 把小串出现的位置，赋值给变量 index
            步骤三. 判断出现的位置是否为-1， 如果位置等于-1，说明大串中已经查询不到小串了；如果位置不等于-1，那么，进行循环，完成次数累加与修改大串的操作
         */
        while ((index = big.indexOf(small)) != -1 ){
            count++;//出现次数+1
            //更改大串内容
            big = big.substring(index+1);
        }
```

```
        return count;
    }
```

第3章 字符串缓冲区

3.1StringBuffer 类

在学习 String 类时，API 中说字符串缓冲区支持可变的字符串，什么是字符串缓冲区呢？接下来我们来研究下字符串缓冲区。

查阅 StringBuffer 的 API，StringBuffer 又称为可变字符序列，它是一个类似于 String 的字符串缓冲区，通过某些方法调用可以改变该序列的长度和内容。

原来 StringBuffer 是个字符串的缓冲区，即就是它是一个容器，容器中可以装很多字符串。并且能够对其中的字符串进行各种操作。

3.2StringBuffer 的方法使用

StringBuffer	**append**(String str) 将指定的字符串追加到此字符序列。
StringBuffer	**delete**(int start, int end) 移除此序列的子字符串中的字符。
StringBuffer	**insert**(int offset, String str) 将字符串插入此字符序列中。
StringBuffer	**replace**(int start, int end, String str) 使用给定 String 中的字符替换此序列的子字符串中的字符。
StringBuffer	**reverse**() 将此字符序列用其反转形式取代。
String	**toString**() 返回此序列中数据的字符串表示形式。

- 代码演示：

```
创建一个字符串缓冲区对象。用于存储数据。
StringBuffer sb = new StringBuffer();
sb.append("haha"); //添加字符串
sb.insert(2, "it");//在指定位置插入
sb.delete(1, 4);//删除
sb.replace(1, 4, "cast");//替换指定范围内的内容
String str = sb.toString();
```

- 注意：append、delete、insert、replace、reverse 方法调用后，返回值都是当前对象自己，所以说，StringBuffer 它可以改变字符序列的长度和内容。

3.3StringBuffer 类方法查找练习

下面的需求所对应的方法，要求大家自己动手在 API 中查找，并进行方法使用。

- 从指定位置开始，到末尾结束，截取该字符串缓冲区，返回新字符串
- 在原有字符串缓冲区内容基础上，删除指定位置上的字符

3.4对象的方法链式调用

在我们开发中，会遇到调用一个方法后，返回一个对象的情况。然后使用返回的对象继续调用方法。这种时候，我们就可以把代码现在一起，如 append 方法一样，代码如下：

```
创建一个字符串缓冲区对象。用于存储数据。
StringBuffer sb = new StringBuffer();
添加数据。不断的添加数据后，要对缓冲区的最后的数据进行操作，必须转成字符串才可以。
String str = sb.append(true).append("hehe").toString();
```

3.5StringBuffer 练习

练习：int[] arr = {34,12,89,68};将一个 int[]中元素转成字符串　格式 [34,12,89,68]

```
public static String toString_2(int[] arr) {
        StringBuffer sb = new StringBuffer();
        sb.append("[");
        for (int i = 0; i < arr.length; i++) {
            if(i!=arr.length-1){
                sb.append(arr[i]+",");
            }else{
                sb.append(arr[i]+"]");
            }
        }
        return sb.toString();
    }
```

- 无论多少数据，数据是什么类型都不重要，只要最终变成字符串就可以使用 StringBuffer 这个容器

3.6StringBuilder 类

查阅 API 发现还有一个 StringBuilder 类，它也是字符串缓冲区，StringBuilder 与它和 StringBuffer 的有什么不同呢？

我们阅读 StringBuilder 的 API 说明发现，它也是一个可变的字符序列。此类提供一个与 StringBuffer 兼容的 API，但不保证同步。该类被设计用作 StringBuffer 的一个简易替换，用在字符串缓冲区被单个线程使用的时候（这种情况很普遍）。如果可能，建议优先采用该类，因为在大多数

实现中，它比 StringBuffer 要快。

目前，我们还没有涉及到线程与同步，知道结论 StringBuilder 比 StringBuffer 快即可。为什么快，我们会在学习线程时讲解。

第4章 总结

4.1知识点总结

- Object：它是所有类的超类，祖宗类。java 中所有的类都直接或间接的继承这个类
  - 方法

public String toString() 返回当前对象中的内容，对于 Object 类默认操作来说，返回的对象的类型+@+内存地址值

public boolean equals(Object obj) 比较两个对象内容是否相同，对于 Object 类默认操作来说，比较的是地址值

- String：字符串类，字符串是常量；它们的值在创建之后不能更改
  - 方法

boolean equals(Object obj) 判断两个字符串中的内容是否相同

boolean equalsIgnoreCase(String str) 判断两个字符串中的内容是否相同，忽略大小写

boolean contains(String str) 判断该字符串中 是否包含给定的字符串

boolean startsWith(String str) 判断该字符串 是否以给定的字符串开头

boolean endsWith(String str) 判断该字符串 是否以给定的字符串结尾

boolean isEmpty() 判断该字符串的内容是否为空的字符串 ""

int length() 获取该字符串的长度

char charAt(int index) 获取该字符串中指定位置上的字符

String substring(int start) 从指定位置开始，到末尾结束，截取该字符串，返回新字符串

String substring(int start,int end) 从指定位置开始，到指定位置结束，截取该字符串，返回新字符串

int indexOf(int ch) 获取给定的字符，在该字符串中第一次出现的位置

int indexOf(String str) 获取给定的字符串，在该字符串中第一次出现的位置

int indexOf(int ch,int fromIndex) 从指定位置开始，获取给定的字符，在该字符

byte[] getBytes() 把该字符串 转换成 字节数组

char[] toCharArray() 把该字符串 转换成 字符数组

String replace(char old,char new) 在该字符串中，将给定的旧字符，用新字符替换

String replace(String old,String new) 在该字符串中， 将给定的旧字符串，用新字符串替换

String trim() 去除字符串两端空格，中间的不会去除，返回一个新字符串

String toLowerCase() 把该字符串转换成 小写字符串

String toUpperCase() 把该字符串转换成 大写字符串

int indexOf(String str,int fromIndex) 从指定位置开始，获取给定的字符串，在该字符串中第一次出现的位置

- StringBuffer/StringBuilder:

■ 方法

public StringBuffer append(String str) 在原有字符串缓冲区内容基础上，在末尾追加新数据

public StringBuffer insert(int offset,String str) 在原有字符串缓冲区内容基础上，在指定位置插入新数据

public StringBuffer deleteCharAt(int index) 在原有字符串缓冲区内容基础上，删除指定位置上的字符

public StringBuffer delete(int start,int end) 在原有字符串缓冲区内容基础上，删除指定范围内的多个字符

public StringBuffer replace(int start,int end,String str)在原有字符串缓冲区内容基础上，将指定范围内的多个字符 用给定的字符串替换

public StringBuffer reverse() 将字符串缓冲区的内容 反转 "abc"----"cba"

public String substring(int start) 从指定位置开始，到末尾结束，截取该字符串缓冲区，返回新字符串

public String substring(int start,int end) 从指定位置开始，到指定位置结束，截取该字符串缓冲区，返回新字符串