<http://www.cnblogs.com/mingforyou/archive/2011/11/01/2231953.html>

JDK1.5加入的增强for和循环.

**foreach语句使用总结**  
  
增强for(part1:part2){part3};   
  
part2中是一个数组对象,或者是带有泛性的集合.   
part1定义了一个局部变量,这个局部变量的类型与part2中的对象元素的类型是一致的.   
part3当然还是循环体.

**foreach语句是java5的新特征之一，在遍历数组、集合方面，foreach为开发人员提供了极大的方便。**

**foreach语句是for语句的特殊简化版本，但是foreach语句并不能完全取代for语句，然而，任何的foreach语句都可以改写为for语句版本。**

**foreach并不是一个关键字，习惯上将这种特殊的for语句格式称之为“foreach”语句。从英文字面意思理解foreach也就是“for 每一个”的意思。实际上也就是这个意思。**

**foreach的语句格式：**

**for(元素类型t 元素变量x : 遍历对象obj){**

**引用了x的java语句;**

**}**

**下面通过两个例子简单例子看看foreach是如何简化编程的。代码如下：**

**一、foreach简化数组和集合的遍历**

**import java.util.Arrays;   
import java.util.List;   
import java.util.ArrayList;   
  
public class TestArray {   
public static void main(String args[]) {   
TestArray test = new TestArray();   
test.test1();   
test.listToArray();   
test.testArray3();   
  
}   
  
/\*\*   
\* foreach语句输出一维数组   
\*/   
public void test1() {   
//定义并初始化一个数组   
int arr[] = {2, 3, 1};   
System.out.println("----1----排序前的一维数组");   
for (int x : arr) {   
System.out.println(x); //逐个输出数组元素的值   
}   
  
//对数组排序   
Arrays.sort(arr);   
  
//利用java新特性for each循环输出数组   
System.out.println("----1----排序后的一维数组");   
for (int x : arr) {   
System.out.println(x); //逐个输出数组元素的值   
}   
}   
  
/\*\*   
\* 集合转换为一维数组   
\*/   
public void listToArray() {   
//创建List并添加元素   
List<String> list = new ArrayList<String>();   
list.add("1");   
list.add("3");   
list.add("4");   
  
//利用froeach语句输出集合元素   
System.out.println("----2----froeach语句输出集合元素");   
for (String x : list) {   
System.out.println(x);   
}   
  
//将ArrayList转换为数组   
Object s[] = list.toArray();   
  
//利用froeach语句输出集合元素   
System.out.println("----2----froeach语句输出集合转换而来的数组元素");   
for (Object x : s) {   
System.out.println(x.toString()); //逐个输出数组元素的值   
}   
}   
  
/\*\*   
\* foreach输出二维数组测试   
\*/   
public void testArray2() {   
int arr2[][] = {{4, 3}, {1, 2}};   
System.out.println("----3----foreach输出二维数组测试");   
for (int x[] : arr2) {   
for (int e : x) {   
System.out.println(e); //逐个输出数组元素的值   
}   
}   
}   
  
/\*\*   
\* foreach输出三维数组   
\*/   
public void testArray3() {   
int arr[][][] = {   
{{1, 2}, {3, 4}},   
{{5, 6}, {7, 8}}   
};   
  
System.out.println("----4----foreach输出三维数组测试");   
for (int[][] a2 : arr) {   
for (int[] a1 : a2) {   
for (int x : a1) {   
System.out.println(x);   
}   
}   
}   
}   
}**

**运行结果：**

**----1----排序前的一维数组   
2   
3   
1   
----1----排序后的一维数组   
1   
2   
3   
----2----froeach语句输出集合元素   
1   
3   
4   
----2----froeach语句输出集合转换而来的数组元素   
1   
3   
4   
----4----foreach输出三维数组测试   
1   
2   
3   
4   
5   
6   
7   
8   
  
Process finished with exit code 0**

**二、foreach语句的局限性**

**通过上面的例子可以发现，如果要引用数组或者集合的索引，则foreach语句无法做到，foreach仅仅老老实实地遍历数组或者集合一遍。下面看一个例子就明白了：**

**public class TestArray2 {   
  
public static void main(String args[]) {   
//定义一个一维数组   
int arr[] = new int[4];   
System.out.println("----未赋值前输出刚刚定义的数组----");   
for (int x : arr) {   
System.out.println(x);   
}   
  
//通过索引给数组元素赋值   
System.out.println("----通过循环变量给数组元素赋值----");   
for (int i = 3; i > 0; i--) {   
arr[i] = i;   
}   
//循环输出创建的数组   
System.out.println("----赋值后，foreach输出创建好的数组----");   
for (int x : arr) {   
System.out.println(x);   
}   
}   
}**

**运行结果：**

**----未赋值前输出刚刚定义的数组----   
0   
0   
0   
0   
----通过循环变量给数组元素赋值----   
----赋值后，foreach输出创建好的数组----   
0   
1   
2   
3   
  
Process finished with exit code 0**

**三、总结**

**foreach语句是for语句特殊情况下的增强版本，简化了编程，提高了代码的可读性和安全性（不用怕数组越界）。相对老的for语句来说是个很好的补充。提倡能用foreach的地方就不要再用for了。在用到对集合或者数组索引的情况下，foreach显得力不从心，这个时候是用for语句的时候了。foreach一般结合泛型使用**