http://qinjiangbo.com/2016/11/06/Guava%E4%BC%98%E7%BE%8E%E4%BB%A3%E7%A0%81-15-%E5%9F%BA%E6%9C%AC%E7%B1%BB%E5%9E%8B/

## Guava基本类型

关于基本类型，我们在Java中有：byte、short、int、long、float、double、char和boolean。我们知道除了这些之外，还有关于这些基本类型的包装类：Byte、Short、Integer、Long、Float、Double、Character和Boolean。原因是这些基本类型没法当成对象或者泛型的参数使用。这意味着许多通用的方法都不支持它们。基于这个原因，Guava提供了若干通用工具，包括原生类型数组与集合API的交互，原生类型和字节数组的相互转换，以及对某些原生类型的无符号形式的支持。

## Java基本类型与Guava对应工具类

| 原生类型 | **Guava工具类** |
| --- | --- |
| byte | Bytes, SignedBytes, U |
| short | Shorts |
| int | Ints, UnsignedInteger, |
| long | Longs, UnsignedLong |
| float | Floats |
| double | Doubles |
| char | Chars |
| boolean | Booleans |

Bytes工具类没有定义任何区分有符号和无符号字节的方法，而是把它们都放到了SignedBytes和UnsignedBytes工具类中，因为字节类型的符号性比起其它类型要略微含糊一些。

int和long的无符号形式方法在UnsignedInts和UnsignedLongs类中，但由于这两个类型的大多数用法都是有符号的，Ints和Longs类按照有符号

形式处理方法的输入参数。

此外，Guava为int和long的无符号形式提供了包装类，即UnsignedInteger和UnsignedLong，以帮助你使用类型系统，以极小的性能消耗对有符号和无符号值进行强制转换。

## 基本类型使用实例

我们说基本类型的操作大多都是一样的，所以我就不每一个类都给出具体的实例啦，这里我们以Ints工具类为例，来解释Guava中基本类型工具类的使用。

```
package com.qinjiangbo;
import com.google.common.collect.Lists;
import com.google.common.primitives.Ints;
import org.junit.Test;
import java.util.List;
/**
 * Date: 9/20/16
 * Author: qinjiangbo@github.io
 */
public class IntsTest {
    @Test
    public void testFindGivenNumberInArray() {
        final int[] array1 = new int[]{0, 15, 49};
        final int[] array2 = new int[]{5, 2, 4, -18, 450};
        System.out.println(Ints.contains(array1, 49)); // true
        System.out.println(Ints.indexOf(array2, 4)); // 2
    }
    @Test
    public void testConcatArrays() {
        final int[] array1 = new int[]{0, 15, 4, 49};
        final int[] array2 = new int[]{5, 2, 4, -18, 450};
        System.out.println(Ints.concat(array1, array2).length); // 9
    }
    @Test
    public void testJoinArrayUsingSeprator() {
        final int[] array1 = new int[]{0, 15, 4, 49};
        final int[] array2 = new int[]{5, 2, 4, -18, 450};
```

```
        System.out.println(Ints.join(" : ", array2)); // 5 : 2 : 4 : -18 : 450
    }
    @Test
    public void testFindMaxAndMinInArray() {
        final int[] array = new int[]{5, 2, 4, -18, 450};
        System.out.println(Ints.min(array)); // -18
        System.out.println(Ints.max(array)); // 450
    }
    @Test
    public void testToArray() {
        List<Integer> ints = Lists.newArrayList(1, 45, 5, 76, 34, 26, 68);
        System.out.println(Ints.toArray(ints)); // [I@1b701da1
        System.out.println(Ints.toArray(ints)[2]); // 5
    }
    @Test
    public void testAsList() {
        final int[] ints = new int[]{1, 45, 36, 76, 23, 6};
        System.out.println(Ints.asList(ints)); // [1, 45, 36, 76, 23, 6]
    }
}
```

上面的代码中注释的部分表示每个测试方法的打印值，大家可以根据打印的值来查看Ints类相应的使用方法。我们注意到Ints类能处理数组转链表以及链表转数组的功能，以及在一个数组中找出最大值和最小值，将一个数组使用某个连接符连接起来成为一个字符串，还有很多很多使用的方法。为我们处理基本类型带来了极大的便利。其他的基本类型工具类可以参考Ints的使用方法。

## 总结

Guava以极小的代价为我们写了一套完整的基本类型处理工具类，让我们可以更高效地处理基本类型相关的问题。其实有很多方法在JDK中我们就会经常使用到，比如说toArray以及asList。Guava没有简单地复用这些方法作为它的方法，而是重新设计实现了一遍，所以效率非常高，建议多多使用Guava提供的方法，它和JDK的性能谁会更高一点？Guava在大多数情况下在相同的或者类似的方法下面

都做了大量的优化，所以Guava的效率更高一点，而且最重要的是Guava更优美！