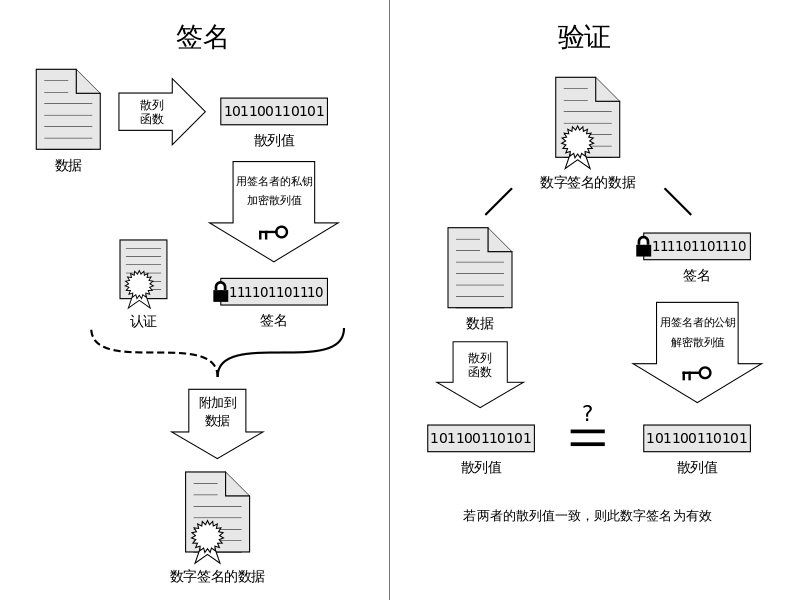
2015-12-8

数字签名



Map

HashMap 是一个散列表，它存储的内容是键值对(key-value)映射。  
HashMap 继承于AbstractMap，实现了Map、Cloneable、java.io.Serializable接口。  
HashMap 的实现不是同步的，这意味着它不是线程安全的。它的key、value都可以为null。此外，HashMap中的映射不是有序的。

构造函数：

// 默认构造函数。

HashMap()

// 指定“容量大小”的构造函数，**默认加载因子是 0.75**

HashMap(int capacity)

// 指定“容量大小”和“加载因子”的构造函数

HashMap(int capacity, float loadFactor)

// 包含“子Map”的构造函数

HashMap(Map<? extends K, ? extends V> map)

**TreeMap 简介**

TreeMap 是一个**有序的key-value集合**，它是通过[红黑树](http://www.cnblogs.com/skywang12345/p/3245399.html)实现的。  
TreeMap **继承于AbstractMap**，所以它是一个Map，即一个key-value集合。  
TreeMap 实现了NavigableMap接口，意味着它**支持一系列的导航方法。**比如返回有序的key集合。  
TreeMap 实现了Cloneable接口，意味着**它能被克隆**。  
TreeMap 实现了java.io.Serializable接口，意味着**它支持序列化**。

TreeMap基于**红黑树（Red-Black tree）实现**。该映射根据**其键的自然顺序进行排序**，或者根据**创建映射时提供的 Comparator 进行排序**，具体取决于使用的构造方法。  
TreeMap的基本操作 containsKey、get、put 和 remove 的时间复杂度是 log(n) 。  
另外，TreeMap是**非同步**的。 它的iterator 方法返回的**迭代器是fail-fastl**的。

**TreeMap的构造函数**

[复制代码](javascript:void(0);)

// 默认构造函数。使用该构造函数，TreeMap中的元素按照自然排序进行排列。

TreeMap()

// 创建的TreeMap包含Map

TreeMap(Map<? extends K, ? extends V> copyFrom)

// 指定Tree的比较器

TreeMap(Comparator<? super K> comparator)

// 创建的TreeSet包含copyFrom

TreeMap(SortedMap<K, ? extends V> copyFrom)

[复制代码](javascript:void(0);)

SortedMap

<http://stackoverflow.com/questions/1440006/java-sortedmap-treemap-comparable-how-to-use>

Set无须，可以使用TreeSet 实现Comparator中CompareTo方法实现自然排序或者自定义排序

Slf4j

基本包导入：

import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;

分别在slf4j-api-X，和slf4j-simple.X下

Debug()

Error()

Trace()

Info()

**Cache vs Buffer**

The terms "buffer" and "cache" tend to be used interchangeably; note however they represent different things. A buffer is used traditionally as an intermediate temporary store for data between a fast and a slow entity. As one party would have to wait for the other affecting performance, the buffer alleviates this by allowing entire blocks of data to move at once rather then in small chunks. The data is written and read only once from the buffer. Furthermore, the buffers are visible to at least one party which is aware of it.

A cache on the other hand by definition is hidden and neither party is aware that caching occurs.It as well improves performance but does that by allowing the same data to be read multiple times in a fast fashion.

A further explanation of the differences between two can be found [here](http://en.wikipedia.org/wiki/Cache_(computing)#The_difference_between_buffer_and_cache).