

# 通用计数器 Counter

方案撰写人：杨英强

联系方式：[cevin15@gmail.com](mailto:cevin15@gmail.com)

微博：[weibo.com/cevin15](http://weibo.com/cevin15)

## 如何使用

### 1、建好数据表

表结构在后面会给出。

### 2、配置数据源

首先，需要在应用启动的时候配置好数据源。比如，在监听器 `ServletContextListener` 的 `contextInitialized( ServletContextEvent event )` 中配置。以确保你在使用计数器前，计数器已配置完毕。使用 `Counter.initDataSource( DataSource dataSource)` 进行配置。

### 2、应用停止前的收尾工作

为确保应用关闭或者重启的时候，能把缓存中的数据入库，需要在应用停止前调用 `new Counter().updateAllBuffer2Db()` 方法。比如，在监听器 `ServletContextListener` 的 `contextDestroyed( ServletContextEvent event )` 中调用此方法。

### 3、为每个计数对象生成一个计数 ID

`new Counter().initOneCounter()` 方法。这个方法会返回一个新生成的 `counterId`，你需要做的是把这个 `id` 跟你的计数对象关联起来。

### 4、为计数对象增加计数

`new Counter().incCounts(long counterId)` 方法。在需要计数的时候，使用这个方法进行计数增长。

### 5、获取计数对象的计数次数

`new Counter().getCurrentCounts(long counterId, boolean containBuffer)` 此方法会将计数对象的当前计数返回，可选是否需要缓存中的计数对象  
`new Counter().getCurrentCounts(long counterId)` 方法。此方法会将计数对象的当前计数返回，包含缓存，也即提供无缓存的计数数据。缺陷，在负载均衡的环境中，使用默认的缓存容器（`HashMap`），获取到的计数数据将可能不准确。

## 数据入库方式

有两种。

一种是自动入库。统计对象的计数次数到达规定的次数之后入库，规定次数可使用下述方法设定


```
/**
 * 每个缓存的最大存储次数，默认为20
 * @param count
 */
public static void initMaxBufferCount(Long count)
```

一种是手动入库。使用下述方式

```
/**
 * 更新所有缓存数据到数据库
 */
public void updateAllBuffer2Db ()
```

## 数据库表设计

使用此通用计数器组件的时候，需要建好以下数据表

	Field	Type	Comment
	id	int(11) NOT NULL	
	counts	int(11) NULL	计数
	today	int(11) NULL	
	this_week	int(11) NULL	
	this_month	int(11) NULL	
	yesterday	int(11) NULL	
	last_week	int(11) NULL	
	last_month	int(11) NULL	
	create_at	datetime NULL	
	update_at	datetime NULL	

## 常用方法

这个组件的常用方法都在 `cn.com.counter.core.Counter` 类中。

```
/**
 * 初始化数据源
 * @param dataSource
 */
public static void initDataSource (DataSource dataSource)

/**
 * 每个缓存的最大存储次数，默认为20
 * @param count
 */
public static void initMaxBufferCount (Long count)

/**
 * 设置存储计数的缓存容器。
 * 只需设置一次
```

```

    * @param container
    */
    public static void setBufferContainer(BufferContainer container)

    /**
     * 初始化counter一行数据
     * @return 刚初始化的主键，为0则初始化失败
     */
    public long initOneCounter()

    /**
     * 计数次数增长
     * @param counterId
     */
    public void incCounts(long counterId)

    /**
     * 获取指定counterId的次数，默认container在负载均衡的环境中无法正常使用
     * @param counterId
     * @return
     */
    public long getCurrentCounts(long counterId)

    /**
     * 获取指定counterId的次数
     * @param counterId
     * @param containBuffer false时，默认container在负载均衡的环境中无法正常使用
     * @return
     */
    public long getCurrentCounts(long counterId, boolean containBuffer)

    /**
     * 更新所有缓存数据到数据库
     */
    public void updateAllBuffer2Db()

    /**
     * 找出counter中指定id的数据
     * @param counterId
     * @return
     */

```

```
public Map<String, Object> findConutsInDb(long counterId)
```

方法备注说明，更新 counter 数据的时候，会更新两个数据：counts 和 today。

## 关于缓存容器

默认我们使用了 `HashMap` 来作为计数器的缓存容器。如果你有使用自定义缓存容器的需求，比如说希望计数器在负载均衡的环境中有更好的表现。我们提供了实现缓存容器的方法。

首先实现这个接口 `cn.com.counter.container.BufferContainer`。

然后调用 `Counter.setBufferContainer(BufferContainer container)`。这样计数器将会使用你自定义的缓存容器来存储未存入数据库中的缓存数据。