Introduction to JABB
– Java Application Building Blocks

Version 1.0.2

2012-04-19

James Hu

Table of Contents

1   What is JABB

JABB is a collection of reusable Java components. Some components within JABB may be inter-dependent, but the intent is not to make JABB a framework. Most of the components are just utility classes that may be used in your projects.

JABB is open source software licensed under Apache License.

Let me know if you are using or considering using JABB in your projects, so that I can better help you.

Change log may be found at the end of this document.

JABB 是一些可重复使用的 Java 应用程序组件。JABB 里面的一些组件可能互相有依赖关系，但是 JABB 的目的不是做成一个框架。大部分的组件只是普通的工具类，你可以在自己的项目里使用它们。

JABB 是依据 Apache 许可发布的开源软件。

如果你正在使用或正在考虑使用 JABB，请务必告诉我，以便我能更好地帮助你。

Change log 在本文档的末尾处。

2　　How to get JABB

JABB is hosted on sourceforge.net. Below are the related URLs:

JABB 在 sourceforge.net 上。相关链接如下：

- Project summary page: http://sourceforge.net/projects/jabb/

- Project home page: http://jabb.sourceforge.net/

- Introduction to JABB (PDF): http://jabb.sourceforge.net/doc/jabb_intro.pdf

- Javadoc: http://jabb.sourceforge.net/javadoc/

- Downloads: http://sourceforge.net/projects/jabb/files

- Source code browsing: http://jabb.svn.sourceforge.net/

- SVN: https://jabb.svn.sourceforge.net/svnroot/jabb

If you want to contact the author(s), find contact information on project home page:

如果你想同作者（们）联系，请在 JABB 主页上找到联络方式：

- Project home page: http://jabb.sourceforge.net/

3   Components guide

3.1   *Apache Camel related*

3.1.1   CamelContextController

3.1.1.1 Related classes

- public class net.sf.jabb.camel.CamelContextController

3.1.1.2 Overview

What if you want to control CamelContext remotely through Telnet? CamelContextController is the answer for you. It creates a server socket and receives commands from the socket. According to the commands received, the encapsulated CamelContext will be controlled.

如果你想通过 Telnet 远程控制一个 CamelContext 该怎么做？CamelContextController 可以帮你做到这一点。它会在一个端口上监听你发来的命令，然后对其所封装的 CamelContext 进行控制。
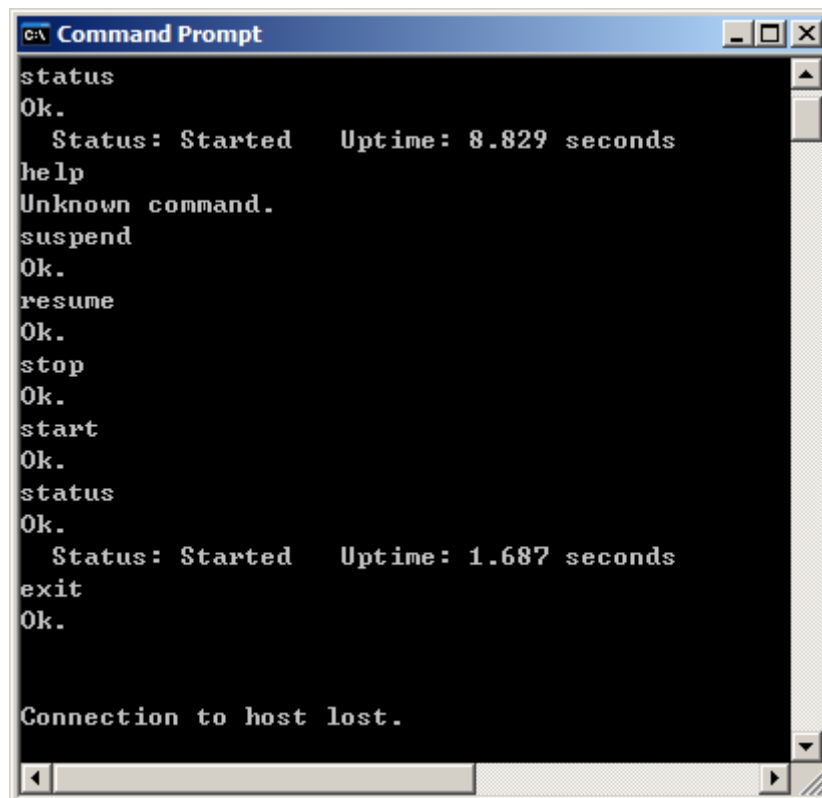
3.1.1.3 Examples

Below you can find the sample code, and the screen shot of an Telnet interaction session.

下面可以看到一段简单的示范代码，以及使用 Telnet 与之交互的截屏。

```java
public static void main(String[] args) {
    Registry registry = new SimpleRegistry();
    CamelContext context = new DefaultCamelContext(registry);
    CamelContextController controller;
    try {
        controller = new CamelContextController(
                context, "tcp://localhost:9001", false);
    } catch (Exception e) {
        e.printStackTrace();
        return;
    }
    controller.start();
}
```

```
status
Ok.
   Status: Started    Uptime: 8.829 seconds
help
Unknown command.
suspend
Ok.
resume
Ok.
stop
Ok.
start
Ok.
status
Ok.
   Status: Started    Uptime: 1.687 seconds
exit
Ok.


Connection to host lost.
```

3.1.2   Registry Manipulation

3.1.2.1 Related classes

- `public class` <u>`net.sf.jabb.camel.CombinedRegistry`</u>
- `public class` <u>`net.sf.jabb.camel.RegistryUtility`</u>

3.1.2.2 Overview

The `Registry` interface of Camel defined some look up methods but did not provide any method for adding or modifying its entries. If in your code you need to add your own entry into `Registry`, you may consider using `CombinedRegistry`.

Camel 的 `Registry` 定义了一些查找接口，但是并不提供对其中内容进行修改的功能。如果你需要在程序中向 `Registry` 里添加内容，则可以考虑使用 `CombinedRegistry`。

If it is the codec information needed by Netty component that you want to add into the `Registry`, then you may consider using those methods provided by

假如你需要向 `Registry` 添加的是 Netty 组件所需要的 Codec 信息，则可以考虑直接使用 `RegistryUtility` 中提供的相应

```
RegistryUtility.                        方法。
```

3.1.2.3 Examples

```java
// You must use CombinedRegistry to construct CamelContext.
Registry registry = new CombinedRegistry(new SimpleRegistry());
CamelContext context = new DefaultCamelContext(registry);

// Then, you can get the CombinedRegistry and put your own entry into it.
CombinedRegistry reg = RegistryUtility.getCombinedRegistry(context);
reg.getDefaultSimpleRegistry().put("my name", "my value");

// And you can put Netty encoder into the Registry.
StringEncoder stringEncoder = new StringEncoder(Charset.defaultCharset());
RegistryUtility.addEncoder(context, "myEncoder", stringEncoder);
context.addRoutes(new RouteBuilder(){
  @Override
  public void configure(){
      from("netty:tcp://localhost:8000?sync=true&encoders=#myEncoder")
      .to("seda:none");
  }
});
```

3.1.3   XmlSocketUtility

3.1.3.1 Related classes

- public class net.sf.jabb.camel.XmlSocketUtility

3.1.3.2 Overview

Nowadays Web Service is the most popular protocol between software modules, which is based on SOAP over HTTP. However in many special occasions, XML over socket protocols are being used.

现在 Web Service 接口已经非常流行了，它的基础其实就是 SOAP over HTTP。但有时候，在特定的场合下会用到 XML over Socket 接口。

Netty component in Camel already makes it easy to receive and process data from socket.

Camel 中的 Netty 组件可以很方便地接收和处理来自 socket 的数据，

XmlSocketUtility makes it even more convenient to receive XML messages from socket and process them.

XmlSocketUtility 使得接收 XML over Socket 形式的消息更为简便，更易于处理。

3.1.3.3 Examples

Code below will enable a server socket on port 9000, all XML messages received will be sent to seda:input for further processing.

下面这段代码会在 9000 端口上开一个服务端口，所有接收到的 XML 消息会被送到 seda:input 以便进行后续处理。

```
XmlSocketUtility.addServer(
      camelContext,                 // The CamelContext
      "tcp://localhost:9000",       // The listening port
      false, "seda:input",          // Destination of the messages received.
            // The combination of false and seda: can be used
            // for multi-threaded processing.
            // If single-threaded processing is desired, then use
            // the combination of true and direct: instead.
      "env:Envelope",         // The top level tag of the XML messages
      Charset.forName("GB2312"),    // Encoding of the XML message
      5000);                        // Possible maximum length of the messages
```

3.2   Jetty related

3.2.1   ServerHelper

3.2.1.1 Related classes:

- public class net.sf.jabb.jetty.ServerHelper

3.2.1.2 Overview

It helps putting Jetty server into Spring container as a bean that can be started and stopped.

用于帮助将 Jetty 服务器嵌入 Spring 容器中进行启动、停止。

3.2.1.3 Examples

```
<bean id="webServer" class="net.sf.jabb.jetty.ServerHelper" >
      <property name="server" ref="jettyServer"/>
```

```
</bean>
```

3.3   JBoss Netty related

3.3.1   XmlDecoder

3.3.1.1 Related classes:

- public class net.sf.jabb.netty.XmlDecoder

3.3.1.2 Overview

A decoder for Netty to handle XML messages.

使得 Netty 能够对 XML 文本消息进行分段的 decoder。

3.3.1.3 Examples

```java
void initialize(){
    // construct a decoder
    XmlDecoder xmlDecoder = new XmlDecoder(
            10000, // maximum length of the XML messages
            "env:Envelope", // top level tag of the XML messages
            CharsetUtil.UTF_8); // character encoding of the messages

    // setup the pipeline to use the decoder
    pipeline.addLast("frameDecoder", xmlDecoder.getFrameDecoder());
    pipeline.addLast("stringDecoder", xmlDecoder.getStringDecoder());
}


void messageReceived(ChannelHandlerContext ctx, MessageEvent e) {
    String msg = (String) e.getMessage();
    System.out.print("The XML body is '" + msg + "'\n");
}
```

3.4   STDR (Struts2 Template Dispatcher Result)

3.4.1   Related classes:

- public class net.sf.jabb.stdr.dispatcher.TemplateDispatcherResult
- public class net.sf.jabb.stdr.jsp.ConfigTag
- public class net.sf.jabb.stdr.jsp.IncludeTag
- public class net.sf.jabb.stdr.jsp.SetTag

3.4.2   Overview

Many times we face the situation that pages of similar layouts need to be developed in Struts2 based projects. For example, on the top there are logo images and main menu, on the left there is a navigation tree, on the right different forms or results are displayed according to the currently selected items and operations in the navigation tree.

Copying similar code pieces among similar jsp files is the most unwise way to implement those pages mentioned above, therefore this approach should be avoided.

Traditionally, the frame mechanism of HTML is used to solve the issue. Different parts of the page are implemented in different files and are assembled by HTML frames. However, user experience of frames is not good, some users just won't accept the partial refresh and content out-of-sync effect. To overcome that, additional Javascript codes need to be written carefully to handle synchronization issues.

AJAX is a quite modern technology, which can also be adopted to conquer this issue. However, not in all projects can AJAX yield the best ROI. For the investment I mean the efforts for learning, design, coding and testing. For the return, I mean the implementation with correct functionality and acceptable UI effect.

Of course there are frameworks designed specifically for template based web site development, such as Tiles, but they seem of heavy weight to most simple applications.

STDR (Struts2 Template Dispatcher Result) addressed above issues with a light weight implementation:

- Page layouts are defined in some jsp files which are used as templates.

我们经常在基于 Struts2 的项目中会需要开发很多布局类似的页面，比如顶上是 logo 和菜单，左边是导航树，右边根据在导航树中当前激活的项目和操作而展现不同的表单或结果。

为了实现上述页面效果，把类似的内容于代码在多份 jsp 页面中重复拷贝是最笨的方法，是应该避免的。

传统的方法是用 HTML 的 frame 机制，把届面中不同的部分放在不同的页面中，靠 frame 机制把它们组合在一起。但是在有的项目中，这样所带来的局部刷新、页面不同步的效果可能不能为用户所接受。而要实现较好的用户体验则必须小心地编写一些 Javascript 代码来实现联动效果。

AJAX 是比较时髦的技术，也可以用来解决上述问题，但是它并不能在任何情形下都给出最佳的投入产出比。这里所说的投入是指学习及设计、编码、测试的人力，而产出是指功能能正确且界面效果让用户满意的实现。

当然还有 Tiles 之类的专门的框架可以使用，但它们对于简单的应用而言显得比较重量级。

STDR (Struts2 Template Dispatcher Result) 用一个轻量级的实现解决了上述的问题：

- 一些 jsp 文件作为页面布局的模板，它们定义布局（一般用普通的 <table>来实现），留出各个页面内

Usually <table> tags are used to separate and reserve the places of content areas in template files.

- Other jsp files implement the display of specific content areas. Data to be displayed usually are retrieved from request or session.

- Finally, templates and content area implementations are assembled in Struts2 configuration file (struts.xml).

容区域的位置。

- 一些 jsp 文件实现具体区域的实现，它们所展现的内容一般取自 request 或 session。

- 最后在 Struts2 的配置文件（struts.xml）中，把上述这些 jsp 文件装配起来。

Names of each content area can be defined in template jsp files. And those names can be referred in Struts2 configuration file (struts.xml). Names of variables can also be defined, which can be used in both template jsp files and content area implementation jsp file. Below is a piece of a sample struts.xml file:

在模板 jsp 中，可以定义各个内容区域的名称；在 Struts2 的配置文件（struts.xml）中可以引用这些名称，还可以定义在模板 jsp 以及区域实现 jsp 里可以使用的变量。以下是一个示范用的 struts.xml 文件的片段：

```xml
<action name="expense" class="flowAction" method="expenseInput">
    <result type="template" name="input">
            <param name="location">/WEB-INF/pages/common/MainFrame.jsp</param>
            <param name="type">TopBottom</param>
            <param name="top">/WEB-INF/pages/flow/ExpenseInput.jsp</param>
            <param name="bottom">/WEB-INF/pages/flow/ExpenseList.jsp</param>
            <param name="selectedMainMenuItem">expense</param>
    </result>
</action>
<action name="income" class="flowAction" method="incomeInput">
    <result type="template" name="input">
            <param name="location">/WEB-INF/pages/common/MainFrame.jsp</param>
            <param name="type">TopBottom</param>
            <param name="top">/WEB-INF/pages/flow/IncomeInput.jsp</param>
            <param name="bottom">/WEB-INF/pages/flow/IncomeList.jsp</param>
            <param name="selectedMainMenuItem">income</param>
    </result>
</action>
```

```xml
<action name="allFlow" class="flowAction">
    <result type="template" name="input">
        <param name="location">/WEB-INF/pages/common/MainFrame.jsp</param>
        <param name="type">TopBottom</param>
        <param name="top">/WEB-INF/pages/allFlow/SearchCriteria.jsp</param>
        <param name="bottom">/WEB-INF/pages/allFlow/SearchResult.jsp</param>
        <param name="selectedMainMenuItem">allFlow</param>
    </result>
</action>
<action name="planning" class="planningAction">
    <result type="template" name="input">
        <param name="location">/WEB-INF/pages/common/MainFrame.jsp</param>
        <param name="type">Simple</param>
        <param name="content">/WEB-INF/pages/common/NotImplemented.jsp</param>
        <param name="selectedMainMenuItem">planning</param>
    </result>
</action>
```

And the relevant part of MainFrame.jsp: 以及 MainFrame.jsp 文件的相关片段：

```xml
<stdr:set var="pageType" paramName="type"/>
<s:if test="%{#pageType == 'TopBottom'}">
    <div id="top" class="top-content">
        <stdr:include paramName="top"/>
    </div>
    <hr/>
    <div id="bottom" class="bottom-content">
        <stdr:include paramName="bottom"/>
    </div>
</s:if>
<s:elseif test="%{#pageType == 'Simple'}">
    <div id="content" class="simple-content">
        <stdr:include paramName="content"/>
    </div>
</s:elseif>
<s:else>
    Page type not recognised.
```

```
</s:else>
```

3.4.3   Getting started

3.4.3.1 Finding the common page layout

首先要在页面中发现可重用的布局形式，只有在多个页面中能够重复利用的布局形式才有提取成模板的价值。

另外，还要在模板的简洁性与内容布局的灵活性之间取得平衡。

比如下，下面这三个页面就可以归纳出一个公共的布局形式：

这个公共的布局形式就是：

Position indicator, login area, and site level menu		
Advertisement area (can have one or zero image/flash element)		

3.4.3.2 Preparing the action class

3.4.3.3 Implementing the content area pages

3.4.3.4 Implementing the template pages

3.4.3.5 Assembling the pages in struts.xml

3.4.3.6 Checking the result

3.5   Generic beans

3.5.1   GenericResult

3.5.1.1 Related classes:

- `public class` <u>`net.sf.jabb.util.bean.GenericResult`</u>

3.5.1.2 Overview

This is the bean to hold generic result/response message.

这是用来存放通用的返回/结果消息的Bean。

3.5.1.3 Properties

boolean	successful	Whether the request was processed successfully.	是否成功处理了请求。
String	errorMessage	Detail of the error if there is any.	出错消息（仅针对操作不成功的情况）
Object	attachment	Any further information.	更进一步的信息。

3.6   *Collections related*

3.6.1   **PutIfAbsentMap**

3.6.1.1 Related classes:

- `public class` `net.sf.jabb.util.col.PutIfAbsentMap`

3.6.1.2 Overview

`PutIfAbsentMap` is a wrapper for `Map`. It adds the following mechanism to the encapsulated `Map`:

1. Whenever the `get` method is invoked, it checks whether a value can be found in the `Map`;

2. If a value can be found in the encapsulated `Map`, go to step 4 directly;

3. If no value can be found, it creates an instance of value object and puts it into the `Map`;

4. Return the value object;

Therefore, it can be ensured that each call to the `get` method can return a not `null` value object.

PutIfAbsentMap 类是对 Map 的一个封装。它给普通的 Map 增加了如下机制：

1. 当 get 方法被调用的时候，先检查是不是被封装的 Map 里有对应的值；

2. 如果可以找到对应的值，则跳到步骤 4；

3. 如果找不到对应的值，则创建一个值对象的实例，并放到 Map 当中去；

4. 返回值对象；

这样一来，无论如何，对 get 方法的调用总能返回一个非 null 的值对象。

In order to create an instance of `PutIfAbsentMap`, you need to give the following to its constructors:

- The `Map` class to be encapsulated

- `Class` of the value object to be created automatically

- (Optional) Parameters for the constructor of the value class.

为了创建 PutIfAbsentMap 的实例，需要把这些传递给它的构造方法：

- 被封装的 Map 的类

- 值对象的类

- （可选）值对象的构造方法所需的参数

These methods are synchronized, which means calls to these methods are multi-thread safe: `get, put, putAll,`

这些方法的调用是同步的，也就是说，是多线程安全的：get, put, putAll, remove, clear

```
remove, clear
```

3.6.1.3 Examples

The following line of code creates an instance based on ConcurrentSkipListMap which has Long as the key and AtomicLong as the value.

下面这行代码创建了一个基于 ConcurrentSkipListMap 的实例，它的 Key 是 Long 类型，Value 是 AtomicLong 类型。

```
counters = new PutIfAbsentMap<Long, AtomicLong>(
        new ConcurrentSkipListMap<Long, AtomicLong>(), AtomicLong.class);
```

The following piece of code shows how to use PutIfAbsentMap. Please note that there is no need to check the return value of get method for null. The count method is multi-thead safe.

下面这段代码展示了如何使用 PutIfAbsentMap。请注意在调用 get 方法之后无需判断返回值是否为 null。count 方法是多线程安全的。

```
public void count(long key, int times){
        counters.get(key).addAndGet(times);
    }
```

3.6.1.4 Cautions

Although PutIfAbsentMap implemented SortedMap and NavigableMap, if the encapsulated Map does not support those interfaces, then if any method of those interfaces was called, Exception will be thrown.

虽然 PutIfAbsentMap 实现了 SortedMap 与 NavigableMap 接口，但是如果被封装的 Map 本身不支持这些接口，那么当运行时调用这些接口所特有的方法的时候，会抛出 Exception。

3.6.2   Number arrays

3.6.2.1 Related classes

- public class <u>net.sf.jabb.util.col.IntegerArray</u>

- public class <u>net.sf.jabb.util.col.LongArray</u>

- ```
  public class net.sf.jabb.util.col.NumberArray
  ```

3.6.2.2 Overview

These classes encapsulate multiple number (such as `int`, `long`, `Integer`, `Long` and `Number`) objects into one object, which makes them suitable to be used as the key of `Map`.

这些类把多个数值（比如 `int`, `long`, `Integer`, `Long`, 和 `Number`）对象封装到一个单一对象中，从而它们适合用来作为 `Map` 的 key。

These methods are supported: hashCode(), toString(), equals(), compareTo().

这些方法是被支持的：hashCode(), toString(), equals(), compareTo()。

3.6.2.3 Examples

```java
LongArray l1 = new LongArray(3,4,5,Long.MAX_VALUE);
LongArray l2 = new LongArray(4,4,5,6);
LongArray l3 = new LongArray(3,4,5,Long.MAX_VALUE);
LongArray l4 = new LongArray(3,4,6,Long.MAX_VALUE);
LongArray l5 = new LongArray(3,5,5,Long.MAX_VALUE);
LongArray l6 = new LongArray(3,4,5,-1);
LongArray l7 = new LongArray(3,4,5,0);
LongArray l8 = new LongArray(3,5,5,Long.MAX_VALUE-100);

Assert.assertTrue(l1.compareTo(l2) == -1);
Assert.assertTrue(l1.compareTo(l3) == 0);
Assert.assertTrue(l1.equals(l3));
Assert.assertTrue(l3.equals(l1));
Assert.assertTrue(l2.compareTo(l3) == 1);

Assert.assertTrue(l1.compareTo(l6) == 1);
Assert.assertTrue(l6.compareTo(l1) == -1);
Assert.assertTrue(l3.compareTo(l4) == -1);
Assert.assertTrue(l5.compareTo(l4) == 1);

System.out.println(l1.getValue(3));
System.out.println(l1.getIntValue(3));

System.out.println(l1.hashCode());
System.out.println(l2.hashCode());
```

```
System.out.println(l3.hashCode());
System.out.println(l4.hashCode());
System.out.println(l5.hashCode());
System.out.println(l6.hashCode());
System.out.println(l7.hashCode());
System.out.println(l8.hashCode());

TreeSet<LongArray> s = new TreeSet<LongArray>();
s.add(l1);
s.add(l2);
s.add(l3);
s.add(l4);
s.add(l5);
s.add(l6);
s.add(l7);
s.add(l8);
System.out.println("****************");
System.out.println(s);
```

3.6.3   ComparableArray

3.6.3.1 Related classes

- `public class net.sf.jabb.util.col.ComparableArray`

3.6.3.2 Overview

3.6.3.3 Examples

3.6.4   MapGetter

3.6.4.1 Related classes

- `public class net.sf.jabb.util.col.MapGetter`

3.6.4.2 Overview

It will try to get value from Maps by several keys one by one. If the first key has no mapped value, then the second will be tried,

它会从 Map 里取 value，一个一个地用你指定的几个 key 来尝试。如果第一个 key 取不到，就用第二个，以此类推。

so one and so forth.

3.6.4.3 Examples

```java
protected Map<String, String> translatedNames;


        MapGetter<String> translator = new MapGetter<String>(locale, "");
        cv.setName(translator.get(c.getTranslatedNames()));
```

3.6.5   MapLister

3.6.5.1 Related classes

- `public class` net.sf.jabb.util.col.MapLister

3.6.5.2 Overview

An utility to list the content of the Map in a formatted manner. It can be used in toString() or elsewhere to output readable debug information.

把 Map 里的内容格式化输出的工具。它可以方便输出易读的调试信息，或者用在 toString()里。

3.6.5.3 Examples

```java
log.debug("The configuration is:\n" + MapLister.listToString(configuration));
```

3.7   Database related

3.7.1   Related classes

- `Public class` net.sf.jabb.util.db.ConnectionUtility
- `Public class` net.sf.jabb.util.db.DataSourceProvider
- `Public class` net.sf.jabb.util.db.StartAndStopSQL

3.7.2   Overview

You can configure database connections and pooling in one place, for plain old JDBC, or for Hibernate inside Spring, or for Hibernate alone, or for DataSource consumers.

你可以只在一个地方配置数据库连接和连接池，然后这个配置可以被传统的 JDBC、Spring 里的 Hibernate、单独的 Hibernate，或者是 DataSource 的使用者所

用到。

StartAndStopSQL enables you to execute SQL when Spring containers start and/or stop.

StartAndStopSQL 使得你可以在 Spring 容器启动和停止的时候执行指定的 SQL。

3.7.3   Examples

3.7.3.1 Database connection configuration

db-connections.properties

```
##### DriverManager or vendor specific ##########
simple      = direct     db-xe.properties
oracle_cached = oracle    db-xe-oracle.properties


##### Connection pool #########
c3p0_basic   = c3p0, db-xe.properties, db-c3p0.properties
c3p0_nopool  = c3p0 db-xe.properties


dbcp_basic   = dbcp db-xe.properties, db-dbcp.properties


proxool_basic      = proxool    db-xe.properties, db-proxool.properties


##### JNDI   #########
tomcat = jndi java:comp/env/jdbc/aaa
weblogic= jndi      jdbc/aaa


##### Combined by try one by one ########
try1  = try simple, c3p0_basic
try2  = try  simple dhcp_basic
try3  = try tomcat weblogic
try4  = try tomcat, simple
```

db-xe.properties

```
_driver=oracle.jdbc.driver.OracleDriver
_url=jdbc:oracle:thin:@localhost:1521:xe
user=sys as sysdba
```

```
password=1234
SetBigStringTryClob=true
```

db-xe-oracle.properties

```
user=sys as sysdba
password=1234
SetBigStringTryClob=true
_databaseName=xe
_driverType=thin
_networkProtocol=tcp
_portNumber=1521
_serverName=localhost
```

db-dbcp.properties

```
defaultAutoCommit=true
initialSize=2
maxActive=3
validationQuery=select 1 from dual
removeAbandoned=false
```

db-proxool.properties

```
proxool.maximum-connection-count=10
proxool.house-keeping-test-sql=select 1 from DUAL
```

3.7.3.2 Providing database connection for Hibernate inside Spring

db-connections.properties

```
moneyflow_internal=proxool, net/sf/moneyflow/db-hsql.properties, net/sf/moneyflow/db-
proxool.properties
```

db-hsql.properties

```
_driver=org.hsqldb.jdbc.JDBCDriver
_url=jdbc\:hsqldb\:file\:database/moneyflow
password=
user=SA
```

db-proxool.properties

```
proxool.maximum-connection-count=10
```

back-end-context.xml

```xml
<bean id="dataSource" class="net.sf.jabb.util.db.ConnectionUtility" factory-
method="createDataSource">

      <constructor-arg value="moneyflow_internal"/>

</bean>


<bean id="sessionFactory"
class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">

      <property name="dataSource" ref="dataSource"/>

      ...

      </property>

</bean>


<bean id="dataManager" class="net.sf.moneyflow.core.DataManager">

      <property name="sessionFactory" ref="sessionFactory"/>

</bean>
```

3.7.3.3 StartAndStopSQL

```xml
<bean id="startAndStopSQL" class="net.sf.jabb.util.db.StartAndStopSQL">

      <property name="dataSource" ref="dataSource"/>

      <property name="stopSQL" value="SHUTDOWN COMPACT;"/>

</bean>
```

3.8   Networking related

3.8.1   Related classes

- Public class net.sf.jabb.util.net.SocketUtility

3.8.2   Overview

3.8.3   Examples

3.8.3.1 Get free port

```java
protected int[] preferredPorts = new int[] {8888, 9999, 8898, 8889, 9998, 9988};
```

```
        int port = SocketUtility.getFreeServerPort(preferredPorts);
        Server server = new Server(new InetSocketAddress("localhost", port));
```

3.9   *PropertiesLoader*

3.9.1   Related classes

- Public class <u>net.sf.jabb.util.prop.PropertiesLoader</u>

3.9.2   Overview

`PropertiesLoader` makes organizing of properties files easy. You can include other files in your properties file, and you can use "*" to tell `PropertiesLoader` to load both classic text properties file format and new XML properties file format.

`PropertiesLoader` is a good utility to load properties files that contain configuration information. It is not a substitution for `ResourceBundle`, because `PropertiesLoader` cannot load locale-specific resource automatically.

`PropertiesLoader` 使得 properties 文件的组织更为方便，因为它允许你在 properties 文件里 include 别的文件，也允许你用星号"*"来告诉它同时匹配传统的 properties 文件和新的 XML 格式 properties 文件。

`PropertiesLoader` 适合用来载入存放配置信息的 properties 文件，但是它不能用来替代 `ResourceBundle`，因为它不支持自动选择载入本地化语言资源。

3.9.3   Examples

3.9.3.1 Properties files

simple.properties

```
in_simple = this is defined in simple.properties
```

inclusion.properties

```
.include = simple.properties xml.xml

inclusion = This is defined in inclusion.properties
```

xml.xml

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties version="1.0">
<comment></comment>
<entry key="xml attribute">This is defined in xml.xml</entry>
</properties>
```

widecard.properties

```
widecard_properties = In widecard.properties
```

widecard.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties version="1.0">
<comment></comment>
<entry key="widecard xml">This is defined in widecard.xml</entry>
</properties>
```

multilayer.properties

```
.include=xml.xml inclusion.properties,
multi=defined in multilayer.properties
```

loop.properties

```
.include=xml.xml inclusion.properties, multilayer.properties
multi=defined in multilayer.properties
```

3.9.3.2 Code

```java
public class PropertiesLoaderDemo {
    public static void main(String[] args) throws
InvalidPropertiesFormatException, IOException {
        PropertiesLoader l = new PropertiesLoader(PropertiesLoaderDemo.class);
        System.out.println(l.load("simple.properties"));
        System.out.println(l.load("xml.xml"));
        System.out.println(l.load("inclusion.properties"));
        System.out.println(l.load("widecard.*"));
        System.out.println(l.load("multilayer.*"));
        System.out.println(l.load("loop.*"));
    }
}
```

3.9.3.3 Output

```
{in_simple=this is defined in simple.properties}
```

```
{xml attribute=This is defined in xml.xml}
```

```
{in_simple=this is defined in simple.properties, xml attribute=This is defined in
xml.xml, inclusion=This is defined in inclusion.properties}
```

```
{widecard xml=This is defined in widecard.xml, widecard_properties=In
widecard.properties}
```

```
{in_simple=this is defined in simple.properties, xml attribute=This is defined in
xml.xml, multi=defined in multilayer.properties, inclusion=This is defined in
inclusion.properties}
```

```
Exception in thread "main" java.lang.IllegalArgumentException: Loop found when
loading properties file: inclusion.properties
```

```
      at net.sf.jabb.util.prop.PropertiesLoader.load(PropertiesLoader.java:139)
```

```
      at net.sf.jabb.util.prop.PropertiesLoader.load(PropertiesLoader.java:151)
```

```
      at net.sf.jabb.util.prop.PropertiesLoader.load(PropertiesLoader.java:151)
```

```
      at net.sf.jabb.util.prop.PropertiesLoader.load(PropertiesLoader.java:186)
```

```
      at
net.sf.jabb.util.prop.test.PropertiesLoaderDemo.main(PropertiesLoaderDemo.java:22)
```

3.10   Number Statistics

3.10.1   Related classes

- `public class  net.sf.jabb.util.stat.AtomicMinLong`

- `public class  net.sf.jabb.util.stat.AtomicMaxLong`

- `public class  net.sf.jabb.util.stat.BasicNumberStatistics`

3.10.2   Overview

If you have a lot of numbers coming from many concurrent threads, and you want to calculate the statistics of them, then you can try these classes.

They are multi-threads safe.

如果你有来自多个并发线程的一些数字，而且想要计算它们的一些统计信息，那么可以试试用这些类来实现。

它们都是多线程安全的。

For example, if there are 100 data files, hidden in each of the file there are hundreds

比如说，有 100 个数据文件，每个文件中藏有数十万个数字，你需要计算所有这些

of thousands of numbers, and you need to calculate the maximum, minimum and average of all these numbers. Then you can implement like the following:

- Create an instance of `BasicNumberStatistics`.

- Read in data files in 100 threads, each thread takes care of one file.

- In each of the threads, once a number is found, give it to the instance of `BasicNumberStatistics` via `put()` method.

- Once all the threads are finished, the instance of `BasicNumberStatistics` can provide you statistics information through `getMin()`, `getMax()`, `getSum()`, `getCount()`, `getAvg()`.

If only `getMin()` is needed, `AtomicMinLong` can be used instead; If only `getMax()` is needed, `AtomicMaxLong` can be used instead.

数字的最大值、最小值和平均值。那么可以这样实现：

- 创建一个 `BasicNumberStatistics` 的实例。

- 分别在 100 个线程中读取数据，每个线程负责一个文件。

- 在每一个线程中，一旦找到了一个数字，就向 `BasicNumberStatistics` 的这个实例去 `put()` 这个数字。

- 当所有这些线程都结束之后，就可以从 `BasicNumberStatistics` 的这个实例中 `getMin()`, `getMax()`, `getSum()`, `getCount()`, `getAvg()`。

如果只需要统计最小值，可以用 `AtomicMinLong`；如果只需要统计最大值，可以用 `AtomicMaxLong`。

3.11   Frequency Counters

3.11.1   Related classes

- `public abstract class net.sf.jabb.util.stat.FrequencyCounter`
- `public class net.sf.jabb.util.stat.BasicFrequencyCounter`
- `public class net.sf.jabb.util.stat.FrequencyCounterDefinition`
- `public class net.sf.jabb.util.stat.PackagedFrequencyCounter`

3.11.2   Overview

Below is the UML class diagram illustrating the architecture view.

下面这张 UML 类图展示了大致的架构。

BasicFrequencyCounter counts the occurrences of events in each of the segments/periods specified by you. For example, it can count hourly hits of a website, numbers of the emails you received within each size segments, and alarm messages generated in each minutes. You can specify the granularity, for example, every minute, every hour, every KB, etc. If the granularity is of a kind of time period, you may also want to specify how the old data get purged.

BasicFrequencyCounter 可以用来帮助你对事件发生的频次进行计数，比如说网站每小时的访问量、所收到的邮件的大小在各个区间的分布、每分钟产生的告警消息的数量等。你可以指定计数的颗粒度，比如每分钟、每小时、每 KB 等。对于时间性质的频次统计，你也可以考虑设置对过旧的数据进行自动删除。

If you want to count the same events in several different granularities, use PackagedFrequencyCounter. It enables you to count in different granularities, for example, every second, every minute, every hour and every day, at the same time.

In fact, all you need to do is to set the configurations to several FrequencyCounterDefinition objects and pass them to the constructor of PackagedFrequencyCounter. Then, within the newly created PackagedFrequencyCounter, a Map of

当你需要同时对不同颗粒度下的频次进行计数的时候，可以使用 PackagedFrequencyCounter，它使得你能够同时在比如每秒、每分钟、每小时、每天这几种不同的颗粒度下进行计数。

实际上，你只要把不同颗粒度的统计定义设置在多个 FrequencyCounterDefinition 里，然后把他们传递给 PackagedFrequencyCounter 的构造方法，它就会自动创建多个相应的 BasicFrequencyCounter 放到一个 Map

BasicFrequencyCounter will be created.   里来帮你做计数。

3.11.3   Examples

```
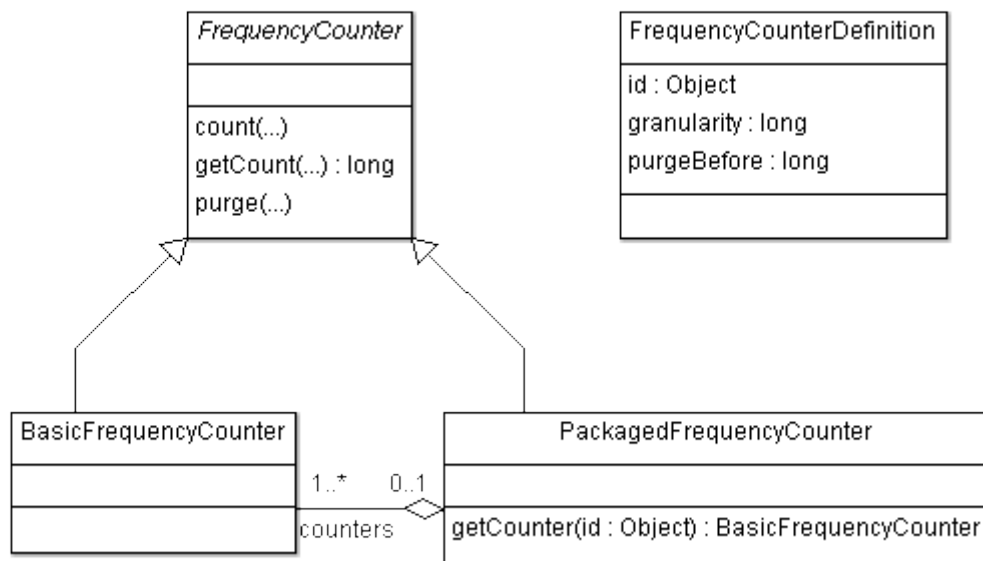PackagedFrequencyCounter counter;
FrequencyCounterDefinition d4 = new FrequencyCounterDefinition("5s",
            2, TimeUnit.SECONDS, 10, TimeUnit.MINUTES);
FrequencyCounterDefinition d1 = new FrequencyCounterDefinition("10S",
            10, TimeUnit.SECONDS, 50, TimeUnit.MINUTES);
FrequencyCounterDefinition d2 = new FrequencyCounterDefinition("30S",
            30, TimeUnit.SECONDS, 3, TimeUnit.HOURS);
FrequencyCounterDefinition d3 = new FrequencyCounterDefinition("1M",
            1, TimeUnit.MINUTES, 7, TimeUnit.HOURS);
counter = new PackagedFrequencyCounter(d1, d2, d3, d4);


…
counter.count();
```

3.12   Text formatting

3.12.1   Related classes

- public class net.sf.jabb.util.text.DurationFormatter
- public class net.sf.jabb.util.text.NameDeduplicator

3.12.2   Overview

DurationFormatter is an utility to format the length of time period to String, for example: 00:03:01.011 or 7d, 12:32:02:001.

DurationFormatter 是一个可以用来格式化时间段信息的工具类，其输出结果比如：00:03:01.011，或：7d, 12:32:02:001。

NameDeduplicator is an utility class that can rename names by appending numbers to avoid name duplication, and it is multi-thread safe.

NameDeduplicator 这个工具类可以通过给名称后面添加数字的方式来为避免重名而自动改名，它是多线程安全的。

3.12.3 Examples

3.12.3.1 DurationFormatter

3.12.3.1.1 Code

```java
long t0 = System.currentTimeMillis();

...

long t1 = System.currentTimeMillis();

...

System.out.println(DurationFormatter.formatSince(t0));  // from t0 to current time

System.out.println(DurationFormatter.format(t1-t0)); // from t0 to t1


Calendar calendar = new GregorianCalendar();

calendar.set(2060, 1, 31, 8, 0);  // Jan 31 2060 8AM

System.out.println(DurationFormatter.formatSince(calendar.getTimeInMillis()));
```

3.12.3.1.2 Output

```
00:00:10.234

00:00:10.016

17710d, 20:09:00.000
```

3.12.3.2 NameDeduplicator

```java
        NameDeduplicator nd;

        for (int i = 0; i < 10; i ++){

                System.out.println(nd.deduplicate("The Name"));

        }
```

3.13   Text matching

3.13.1 Related classes

- public class net.sf.jabb.util.text.StringStartWithMatcher
- public class net.sf.jabb.util.text.UrlStartWithMatcher
- public class net.sf.jabb.util.text.StartWithMatcher
- public class net.sf.jabb.util.text.KeywordMatcher
- public class net.sf.jabb.util.text.RegExpSubstitution

3.13.2 Overview

All these classes depend on dk.brics.automaton.RunAutomaton – a regular expression matching library based on state machine model featuring "use more memory space for better performance".

这些类都依赖于 dk.brics.automaton.RunAutomaton——一个基于状态机模型的"以空间换性能"的正则表达式匹配库。

Both `StringStartWithMatcher` and `UrlStartWithMatcher` can be used to answer the "which pattern does it start with" question with high performance for huge amount of strings. The former normally handles text or phone numbers, while the latter targets URL. `StartWithMatcher` is their common parent class and does not need to be used directly. The matching is case sensitive. If one matching string starts with another, and the text string starts with them, then the longer one will be considered to be matched.

`StringStartWithMatcher` 与 `UrlStartWithMatcher` 都是用来快速地对海量字符串"是否以....开头"进行判断的。前者一般用来处理普通文本或者是电话号码，后者专门针对 URL。`StartWithMatcher` 是他们共同的父类，一般不需要用到。匹配时对大小写敏感。如果匹配字符串之间互相饱含，则匹配其中最长的。

`KeywordMatcher` can be used to check which keywords a text matches, and for each matched keyword how many occurrences are found. It is also suitable for processing huge amount of strings with high performance. The matching is case sensitive. If one matching string starts with another, and the text string starts with them, then the longer one will be considered to be matched.

`KeywordMatcher` 可以用来检查文本当中匹配了哪些关键词，以及每个被匹配到的关键词出现了多少次。它也是适合对海量字符串进行快速匹配处理的。匹配时对大小写敏感。如果匹配字符串之间互相饱含，则匹配其中最长的。

`RegExpSubstitution` is an utility that can substitute part of the String that matches specified regular expression(s) with another specified String.

An instance of this class can be used many times for substitution for different Strings. The performance overhead of each time of substitution is very small. Therefore it is very suitble to be used in occasions that the substitution criteria is fixed while the Strings to be substituted are of huge volume. Please

`RegExpSubstitution` 是一个工具类，它可以对一个或多个正则表达式所匹配到的字符串中的内容作替换。

这个类的一个实例可用来多次对不同的字符串进行替换，每次替换时的性能开销很小。因此它适合用在替换条件固定，但待替换字符串数量巨大的情形下。注意这里的正则表达式不支持"^"和"$"。

be aware that regular expressions here do not
support "^" and "$"

3.13.3   Examples

3.13.3.1   *StringStartWithMatcher*

3.13.3.1.1 Code

```java
Map<String, Object> heads = new HashMap<String, Object>();


///////////// 号段匹配 ///////////////
heads.clear();
heads.put("134", "134号段");
heads.put("135", "135号段");
heads.put("136", "136号段");
heads.put("1361", "1361号段");
heads.put("1362", "1362号段");
heads.put("137", "137号段");
heads.put("138", "138号段");
heads.put("13817", "13817号段");
heads.put("13817726996", "我的号码");
heads.put("138177269", "很接近我的号码");
heads.put("1381772", "有些接近我的号码");
heads.put("中华人民共和国", "中华人民共和国");
heads.put("中华人民", "中华人民");
heads.put("中华", "中华");
// 号段展开
    StringStartWithMatcher.expandNumberMatchingRange(heads, "1335000",
"1335999", "1335000~1335999");
    StringStartWithMatcher.expandNumberMatchingRange(heads, "1375010",
"1375039", "1375010~1375039");
    StringStartWithMatcher.expandNumberMatchingRange(heads, "13750632",
"13750641", "13750632~13750641");
    StringStartWithMatcher.expandNumberMatchingRange(heads, "130120",
"130139", "130120~130139");
    StringStartWithMatcher.expandNumberMatchingRange(heads, "130125",
"130129", "130125~130129");
    StringStartWithMatcher.expandNumberMatchingRange(heads, "1891",
"189299", "1891~189299");
    StringStartWithMatcher.expandNumberMatchingRange(heads, "1881991",
"1882", "1881991~1882");
```

```
        System.out.println("\t*** 展开后的匹配对应表 *******");

        SortedSet<String> ss = new TreeSet<String>(heads.keySet());

        for (String pattern: ss){

                System.out.format("\t %-15s ---> %s\n", pattern,
heads.get(pattern));

        }

        System.out.println();


        m = new StringStartWithMatcher(heads);


        System.out.println("  **** 匹配结果 *******");

        for (String s: new String[] {

                        "1376726637", "13717726996", "1340898394",

                        "18", "138", "1385", "13817", "13817899633", "1381772",
"13817726",

                        "13817726997", "13817726996",

                        "138177269977", "138177269967", "138177269", "1381772699",

                        "13817726997744", "138177269967343",

                        "133500166", "1335010", "133501",

                        "1891234", "18923434", "1892",

                        "1301213243", "130138090", "13009090", "13012689892",

                        "中间位置", "中国人民", "中华大地", "中华民国", "中华人民共有的财
产", "中华人民共和国", "中华人民共和国成立了",

                        "1360172", "13610238", "1362834", "2137138139",
"134136137138139"

                }){

            Object o = m.match(s);

            List<Object> lo = m.matchAll(s);

            System.out.format("\t %-15s ===> %s  |%s\n", s, (o!=null ?
o.toString() : "null"),

                            (lo!=null ? lo.toString() : "null"));

        }

        System.out.println();
```

3.13.3.1.2 Output

```
    *** 展开后的匹配对应表 *******
    13012           ---> 130120~130139
    130125          ---> 130125~130129
    130126          ---> 130125~130129
```

```
130127              ---> 130125~130129
130128              ---> 130125~130129
130129              ---> 130125~130129
13013               ---> 130120~130139
1335                ---> 1335000~1335999
134                 ---> 134号段
135                 ---> 135号段
136                 ---> 136号段
1361                ---> 1361号段
1362                ---> 1362号段
137                 ---> 137号段
137501              ---> 1375010~1375039
137502              ---> 1375010~1375039
137503              ---> 1375010~1375039
13750632            ---> 13750632~13750641
13750633            ---> 13750632~13750641
13750634            ---> 13750632~13750641
13750635            ---> 13750632~13750641
13750636            ---> 13750632~13750641
13750637            ---> 13750632~13750641
13750638            ---> 13750632~13750641
13750639            ---> 13750632~13750641
13750640            ---> 13750632~13750641
13750641            ---> 13750632~13750641
138                 ---> 138号段
13817               ---> 13817号段
1381772             ---> 有些接近我的号码
138177269           ---> 很接近我的号码
13817726996         ---> 我的号码
1881991             ---> 1881991~1882
1881992             ---> 1881991~1882
...
1882997             ---> 1881991~1882
1882998             ---> 1881991~1882
1882999             ---> 1881991~1882
1891                ---> 1891~189299
1892                ---> 1891~189299
```

中华   ---> 中华

中华人民   ---> 中华人民

中华人民共和国   ---> 中华人民共和国

**** 匹配结果 *******

    1376726637   ===> 137号段 |[137号段]

    13717726996   ===> 137号段 |[137号段]

    1340898394   ===> 134号段 |[134号段]

    18   ===> null |null

    138   ===> 138号段 |[138号段]

    1385   ===> 138号段 |[138号段]

    13817   ===> 13817号段 |[138号段, 13817号段]

    13817899633   ===> 13817号段 |[138号段, 13817号段]

    1381772   ===> 有些接近我的号码 |[138号段, 13817号段, 有些接近我的号码]

    13817726   ===> 有些接近我的号码 |[138号段, 13817号段, 有些接近我的号码]

    13817726997   ===> 很接近我的号码 |[138号段, 13817号段, 有些接近我的号码, 很接
近我的号码]

    13817726996   ===> 我的号码 |[138号段, 13817号段, 有些接近我的号码, 很接近我的
    号码, 我的号码]

    138177269977   ===> 很接近我的号码 |[138号段, 13817号段, 有些接近我的号码, 很接
近我的号码]

    138177269967   ===> 我的号码 |[138号段, 13817号段, 有些接近我的号码, 很接近我的
    号码, 我的号码]

    138177269   ===> 很接近我的号码 |[138号段, 13817号段, 有些接近我的号码, 很接
近我的号码]

    1381772699   ===> 很接近我的号码 |[138号段, 13817号段, 有些接近我的号码, 很接
近我的号码]

    13817726997744   ===> 很接近我的号码 |[138号段, 13817号段, 有些接近我的号码, 很接
近我的号码]

    138177269967343   ===> 我的号码 |[138号段, 13817号段, 有些接近我的号码, 很接近我的
号码, 我的号码]

    133500166   ===> 1335000~1335999 |[1335000~1335999]

    1335010   ===> 1335000~1335999 |[1335000~1335999]

    133501   ===> 1335000~1335999 |[1335000~1335999]

    1891234   ===> 1891~189299 |[1891~189299]

    18923434   ===> 1891~189299 |[1891~189299]

    1892   ===> 1891~189299 |[1891~189299]

```
1301213243       ===> 130120~130139  |[130120~130139]
130138090        ===> 130120~130139  |[130120~130139]
13009090         ===> null  |null
13012689892      ===> 130125~130129  |[130120~130139, 130125~130129]
```

中间位置   ===> null |null

中国人民   ===> null |null

中华大地   ===> 中华 |[中华]

中华民国   ===> 中华 |[中华]

中华人民共有的财产   ===> 中华人民 |[中华，中华人民]

中华人民共和国   ===> 中华人民共和国 |[中华，中华人民，中华人民共和国]

中华人民共和国成立了   ===> 中华人民共和国 |[中华，中华人民，中华人民共和国]

1360172   ===> 136号段 |[136号段]

13610238   ===> 1361号段 |[136号段，1361号段]

1362834   ===> 1362号段 |[136号段，1362号段]

2137138139   ===> null |null

134136137138139 ===> 134号段 |[134号段]

3.13.3.2   UrlStartWithMatcher

3.13.3.2.1 Code

```java
Map<String, Object> heads = new HashMap<String, Object>();

/////////////  URL匹配  ///////////////
heads.clear();
for (String s: new String[] {
            "news.sina.com.cn",
            "*.sina.com.cn",
            "*.sina.com.cn/z",
            "*.sina.com.cn/images",
            "*.baidu.cn",
            "*.baidu.com",
            "*.baidu.com/se",
            "*.baidu.com/set",
            "www.baidu.com/search/",
            "www.baidu.com/",
            "*.sina.com.cn/news/daily/a.jpg"
    }){
    heads.put(s, s);
}
```

```
        System.out.println("   ------- 匹配对应表 -------");
        for (String pattern: heads.keySet()){
            System.out.format("   %-35s ---> %s\n", pattern,
heads.get(pattern));
        }
        System.out.println();


        m = new UrlStartWithMatcher(heads);


        System.out.println("   ------- 匹配结果 -------");
        for (String s: new String[] {
                "http://new-s.sina.com.cn",
                "http://news.sina.com.cn",
                "https://news_sina3com4cn",
                "https://news.sina.com.cn/z/2010chunyun/index.shtml",
                "http://h3.news.sina.com.cn/z/2010chunyun/index.shtml",
                "ent.sina.com.cn/entertainment/x/3/a.html",
                "news.sina.com.cn",
                "news_sina3com4cn",
                "news.sina.com.cn/z/2010chunyun/index.shtml",
                "ent.sina.com.cn/entertainment/x/3/a.html",
                "hollywood.sina.com.cn/entertainment/news/today.html",
                "www.baidu.com/search?key=3&a=b",
                "mp3.baidu.com/download/song=1",
                "video.baidu.com/screen/video=2",
                "www.baidu.com/news/a.html",
                "www.baidu.com/setup/",
                "www.baidu.com/search/abcd.jpg",
                "www.baidu.com/search/mp3/test.jpg",
                "www.baidu.com/search/video/flash/new.jpg",
                "www.baidu.com/news/daily/headline.jpg",
                "www.baidu.com/news/daily/common/headline.jpg"
                }){
            Object o = m.match(s);
            System.out.format("   %-54s ===> %s\n", s, (o!=null ?
o.toString() : "null"));
        }
```

3.13.3.2.2 Output

```
------- 匹配对应表 -------
*.baidu.com/set                          ---> *.baidu.com/set
*.sina.com.cn                            ---> *.sina.com.cn
*.baidu.com                              ---> *.baidu.com
*.sina.com.cn/images                     ---> *.sina.com.cn/images
news.sina.com.cn                         ---> news.sina.com.cn
www.baidu.com/search/                    ---> www.baidu.com/search/
*.baidu.cn                               ---> *.baidu.cn
*.sina.com.cn/news/daily/a.jpg           ---> *.sina.com.cn/news/daily/a.jpg
*.sina.com.cn/z                          ---> *.sina.com.cn/z
*.baidu.com/se                           ---> *.baidu.com/se
www.baidu.com/                           ---> www.baidu.com/


------- 匹配结果 -------
http://new-s.sina.com.cn                                    ===> *.sina.com.cn
http://news.sina.com.cn                                     ===> news.sina.com.cn
https://news_sina3com4cn                                    ===> null
https://news.sina.com.cn/z/2010chunyun/index.shtml          ===> *.sina.com.cn/z
http://h3.news.sina.com.cn/z/2010chunyun/index.shtml        ===> *.sina.com.cn/z
ent.sina.com.cn/entertainment/x/3/a.html                    ===> *.sina.com.cn
news.sina.com.cn                                            ===> news.sina.com.cn
news_sina3com4cn                                            ===> null
news.sina.com.cn/z/2010chunyun/index.shtml                  ===> *.sina.com.cn/z
ent.sina.com.cn/entertainment/x/3/a.html                    ===> *.sina.com.cn
hollywood.sina.com.cn/entertainment/news/today.html         ===> *.sina.com.cn
www.baidu.com/search?key=3&a=b                              ===> *.baidu.com/se
mp3.baidu.com/download/song=1                               ===> *.baidu.com
video.baidu.com/screen/video=2                              ===> *.baidu.com
www.baidu.com/news/a.html                                   ===> www.baidu.com/
www.baidu.com/setup/                                        ===> *.baidu.com/set
www.baidu.com/search/abcd.jpg                               ===>
www.baidu.com/search/
www.baidu.com/search/mp3/test.jpg                           ===>
www.baidu.com/search/
www.baidu.com/search/video/flash/new.jpg                    ===>
www.baidu.com/search/
www.baidu.com/news/daily/headline.jpg                       ===> www.baidu.com/
www.baidu.com/news/daily/common/headline.jpg                ===> www.baidu.com/
```

3.13.3.3   KeywordMatcher

3.13.3.3.1 Code

```java
Map<String, Object> keywords = new HashMap<String, Object>();

keywords.clear();
keywords.put("中国", "中国");
keywords.put("中国人", "中国人");
keywords.put("中华人民共和国", "中华人民共和国");
keywords.put("毛泽东", "毛泽东");
keywords.put("江泽民", "江泽民");
keywords.put("天安门", "天安门");
keywords.put("年", "年");
keywords.put("北京", "北京");
keywords.put("上海", "上海");
keywords.put("，", "逗号");

System.out.println("*** 关键词表 *******");
for (String w: keywords.keySet()){
    System.out.format("\t %-15s ---> %s\n", w, keywords.get(w));
}
System.out.println();


m = new KeywordMatcher(keywords);

String s ="1949年10月1日，在北京天安门上，毛泽东庄严宣布，\n"
    + "中华人民共和国成立了，从此，中国人民站起来了。这是全中国人民的节日，\n"
    + "北京、上海等地的人民欢呼雀跃。";
System.out.println("*** 文本 *******");
System.out.println(s);
System.out.println();


Map<Object, MutableInt> result = m.match(s);


System.out.println("*** 结果 *******");
for (Object o: result.keySet()){
    System.out.format("\t %-15s ===> %d\n", o,
                        result.get(o).intValue());
}
System.out.println();
```

3.13.3.3.2 Output

```
***  关键词表  *******

    上海              --->  上海

    北京              --->  北京

    天安门             --->  天安门

    中国人             --->  中国人

    江泽民             --->  江泽民

    中华人民共和国          --->  中华人民共和国

    年               --->  年

    ,               --->  逗号

    毛泽东             --->  毛泽东

    中国              --->  中国


***  文本  *******

1949年10月1日，在北京天安门上，毛泽东庄严宣布，

中华人民共和国成立了，从此，中国人民站起来了。这是全中国人民的节日，

北京、上海等地的人民欢呼雀跃。


***  结果  *******

    上海              ===>  1

    北京              ===>  2

    中国人             ===>  2

    天安门             ===>  1

    中华人民共和国          ===>  1

    年               ===>  1

    毛泽东             ===>  1

    逗号              ===>  6
```

3.14   Sequencers

3.14.1.1   Related classes

- public class net.sf.jabb.util.thread.Sequencer

- public class net.sf.jabb.util.thread.RangedSequencer

3.14.1.2   Overview

These sequence number generators are multi-threads safe and of high performance thanks to the `AtomicLong` class of JDK.

`Sequencer` generates incremental numbers from `0` to `Long.MAX_VALUE`, while `RangedSequencer` allow you to specify a range.

利用 JDK 中的 `AtomicLong`，这些序列值生成器可以同时做到多线程安全与高性能。

`Sequencer` 在 `0` 到 `Long.MAX_VALUE` 区间生成渐增的数值，而 `RangedSequencer` 允许你自己指定区间。

3.14.1.3   Examples

```
Sequencer stringEncoderSeq = new Sequencer();

RangedSequencer rangedSeq = new RangedSequencer(21,100);

...

String stringEncoderName = "stringEncoder" + stringEncoderSeq.next();

long l = rangedSeq.next();
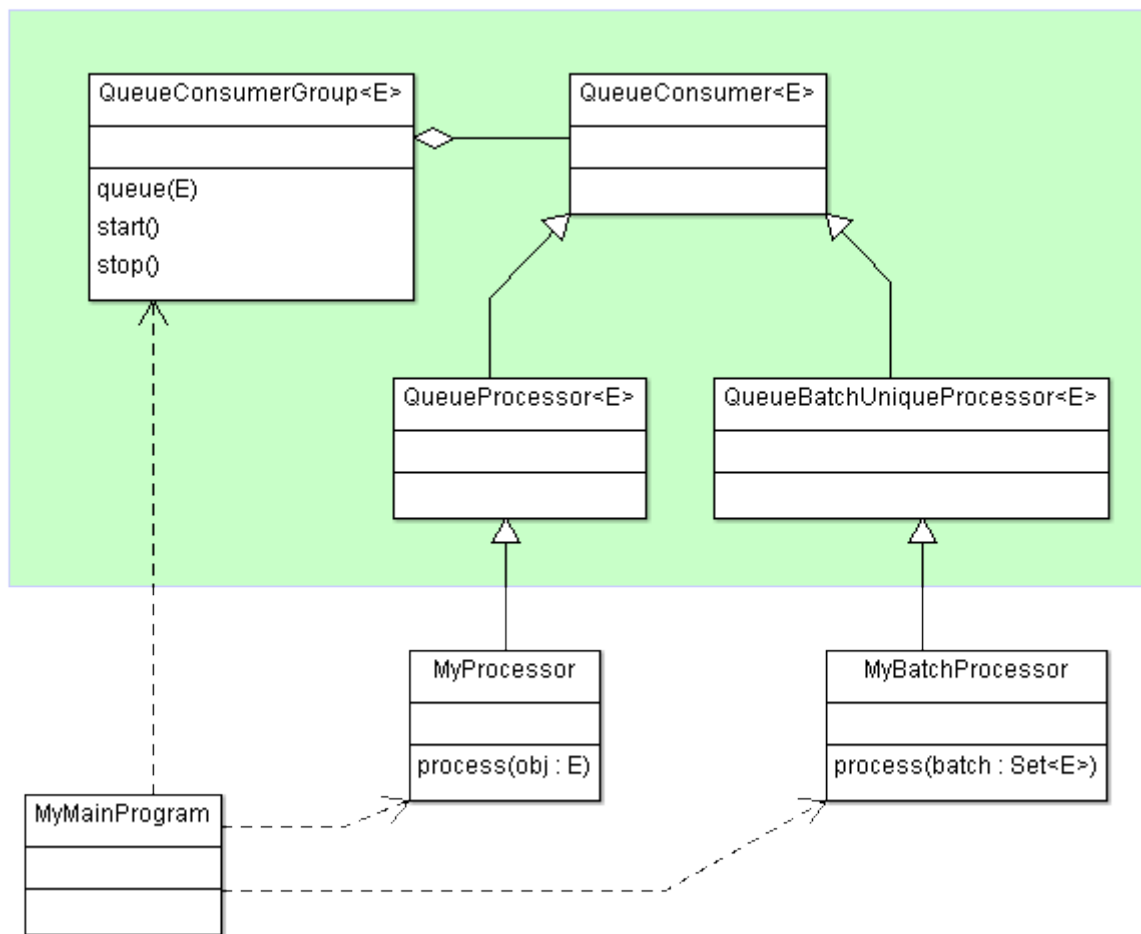```

# 3.15    Concurrent data processing

### 3.15.1.1    Related classes

- public abstract class net.sf.jabb.util.thread.QueueConsumer

- public abstract class net.sf.jabb.util.thread.QueueProcessor

- public abstract class net.sf.jabb.util.thread.QueueBatchUniqueProcessor

- public class net.sf.jabb.util.thread.QueueConsumerGroup

### 3.15.1.2    Overview

Below is the UML class diagram illustrating the architecture view.

下面这张 UML 类图展示了大致的架构。

These classes formed a tiny framework to ease the programming of multi-threaded concurrent data processing.

All you need to do are:

1. Create your own processor class by extending either `QueueProcessor` or `QueueBatchUniqueProcessor` and implement the `process()` method of it.

2. Instantiate a `BlockingQueue` to hold the data for processing.

3. Instantiate as many instances of your processor class as you want. Every instance will have a corresponding working thread created automatically whenever needed.

这些类构成了一个小型的框架，能够简化多线程并行数据处理的程序开发。

用起来很简单，只要这样：

1. 创建你自己的处理实现类，继承 `QueueProcessor` 或者 `QueueBatchUniqueProcessor`，实现其中的 `process()` 方法。

2. 用 `BlockingQueue` 的实例来存放待处理的数据

3. 对于你自己的处理实现类，创建足够多的实例。每个实例都会有一个对应的工作线程在需要的时候被创建。

4. 创建一个 `QueueConsumerGroup` 的实例来管理和控制整个多线程数

4. Create an instance of `QueueConsumerGroup` to manage and control the  multi-threaded processing.

据处理过程。

QueueProcessor is suitable for occasions that data need to be processed one by one.

QueueBatchUniqueProcessor is suitable for occasions that data need to be processed in batches. And it can de-duplicate data in the same batch.

QueueProcessor 适合于需要对数据进行一个一个地处理的场合。

QueueBatchUniqueProcessor 适合于需要对数据进行分批处理的场合。而且它能够在每批中去除掉重复的数据。

### 3.15.1.3    *Examples*

```java
public class TestStringProcessor extends QueueProcessor<String> {
    public TestStringProcessor(){
        super();
    }


    @Override
    public void process(String obj) {
        try {
            Thread.sleep(800);
        } catch (InterruptedException e) {
            // do nothing
        }
        System.out.println("Processed by " + this.getName() + " : " + obj);
    }
}
```

…

```java
        ArrayList<QueueConsumer<String>> processors =
                new ArrayList<QueueConsumer<String>>(20);
        for (int i = 0; i < 20; i ++){
            processors.add(new TestStringProcessor());
        }
        QueueConsumerGroup<String> group =
```

```java
                new QueueConsumerGroup<String>(200, processors);


        group.start();
        for (int i = 0; i < 250; i ++){
                String s = "This is a string_" + i;
                try {
                        group.queue(s);
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                System.out.println("Queued: " + s);
        }
        group.stop();
        System.out.println("All finished!");
```

4   Change Log

Version 1.0.2	
Updated or added external jar files that jabb depends on: • automaton • camel-core • camel-netty • commons-collections • commons-logging • commons-pool • freemarker • hibernate-commons-annotations • hibernate-core • javaassist • javax.servlet.jsp • jboss-logging • jetty-server • jetty-util • log4j • netty • ognl • org.springframework.context • servlet-api • slf4j-api • slf4j-log4j • struts2-core • xwork-core	更新或增加了 jabb 所依赖的外部 jar 文件。 （请看左边的列表）
The Eclipse project of jabb's source code no longer exports jar files it depends.	Jabb 源代码的 Eclipse 项目不再对外输出它所依赖的那些.jar 文件。

net.sf.jabb.util.db.HibernateConnectionProvider now depends on Hibernate 4 (But this class is still unfinished)	net.sf.jabb.util.db.HibernateConnectionProvider 现在依赖于 Hibernate 4 （但这个类仍旧是未完成）
Added: net.sf.jabb.jetty.ServerHelper. It helps putting Jetty server into Spring container as a bean that can be started and stopped.	增加了：net.sf.jabb.jetty.ServerHelper，用于帮助将 Jetty 服务器嵌入 Spring 容器中进行启动、停止。
Added: net.sf.jabb.util.bean.DoubleValueBean, KeyValueBean and JQueryGridData. The former two are just normal Java beans, while the latter encapsulates the JSON data structure requred by jqGrid.	增加了：net.sf.jabb.util.bean.DoubleValueBean, KeyValueBean 以及 JQueryGridData。前两者只是简单的 Bean，后者封装了 jqGrid 所需的 JSON 数据结构。
Added: net.sf.jabb.util.bean.jstree.* They encapsulate the JSON data structure required by jsTree.	增加了：net.sf.jabb.util.bean.jstree.* 它们封装了 jsTree 所需要的 JSON 数据结构。
Added: net.sf.jabb.util.col.MapGetter. This class makes it easy to try to retrive value from Maps by several keys in predefined order.	增加了：net.sf.jabb.util.col.MapGetter。这个类可用于从 Map 中按照预先设定好的次序逐个尝试一系列的 key，看看能不能取到对应的 value。
Added: net.sf.jabb.util.db.StartAndStopSQL. This class can be defined as Spring bean. It can execute SQLs when the Spring container starts and stops.	增加了: net.sf.jabb.util.db.StartAndStopSQL。它可以被定义为 Spring 的 bean，从而在 Spring 容器启动和停止的时候执行你设定的 SQL。
Added: net.sf.jabb.util.net.SocketUtility. This is an utility class for socket related tasks, such as checking whether a port is currently in use or not.	增加了：net.sf.jabb.util.net.SocketUtility。这是关于 socket 的一个工具类，比如可以用来检查某个端口是否当前正被占用。
Added: net.sf.jabb.util.web.WebApplicationConfiguration. It encapsulates common configuration data that a web application normally needs.	增加了：net.sf.jabb.util.web.WebApplicationConfiguration。它封装了一个普通的 web 应用所可能需要的常见配置数据。
Added: net.sf.jabb.util.bean.StringKeyValueBean	增加了：net.sf.jabb.util.bean.StringKeyValueBean
Added: net.sf.jabb.util.web.CharsetFilter It allows you to alter the charset attribute in http response.	增加了：net.sf.jabb.util.web.CharsetFilter 它使你可以修改 http 响应中的 charset 属性。
Renamed PutOnGetMap to PutIfAbsentMap. The	把 PutOnGetMap 改名为了 PutIfAbsentMap，

formal still remains in jabb, but is now depreciated.	前者仍旧保留在 JABB 中，但是已经不推荐使用了。
Version 1.0.1	
First official release.	第一个正式发布版本。