| 组件 | 功能介绍 |
| --- | --- |
| BeanUtils | 提供了对于 JavaBean 进行各种操作，克隆对象,属性等等. |
| Betwixt | XML 与 Java 对象之间相互转换. |
| Codec | 处理常用的编码方法的工具类包 例如 DES、SHA1、MD5、Base64 等. |
| Collections | java 集合框架操作. |
| Compress | java 提供文件打包 压缩类库. |
| Configuration | 一个 java 应用程序的配置管理类库. |
| DBCP | 提供数据库连接池服务. |
| DbUtils | 提供对 jdbc 的操作封装来简化数据查询和记录读取操作. |
| Email | java 发送邮件 对 javamail 的封装. |
| FileUpload | 提供文件上传功能. |
| HttpClien | 提供 HTTP 客户端与服务器的各种通讯操作. 现在已改成 HttpComponents |
| IO | io 工具的封装. |
| Lang | Java 基本对象方法的工具类包 如：StringUtils,ArrayUtils 等等. |
| Logging | 提供的是一个 Java 的日志接口. |
| Validator | 提供了客户端和服务器端的数据验证框架. |

# 1、BeanUtils

提供了对于 JavaBean 进行各种操作， 比如对象,属性复制等等。

[java] view plaincopy

```
1.  //1、 克隆对象
2.  //  新创建一个普通 Java Bean，用来作为被克隆的对象
3.
4.      public class Person {
5.      private String name = "";
6.      private String email = "";
7.
8.      private int age;
9.      //省略 set,get 方法
10.     }
11.
12. //  再创建一个 Test 类，其中在 main 方法中代码如下：
13.     import java.lang.reflect.InvocationTargetException;
```

```
14.     import java.util.HashMap;
15.     import java.util.Map;
16.     import org.apache.commons.beanutils.BeanUtils;
17.     import org.apache.commons.beanutils.ConvertUtils;
18.     public class Test {
19.
20.     /**
21.
22.     * @param args
23.
24.     */
25.     public static void main(String[] args) {
26.     Person person = new Person();
27.     person.setName("tom");
28.     person.setAge(21);
29.     try {
30.             //克隆
31.         Person person2 =  (Person)BeanUtils.cloneBean(person);
32.         System.out.println(person2.getName()+">>"+person2.getAge());
33.     } catch (IllegalAccessException e) {
34.         e.printStackTrace();
35.     } catch (InstantiationException e) {
36.         e.printStackTrace();
37.     } catch (InvocationTargetException e) {
38.         e.printStackTrace();
39.     } catch (NoSuchMethodException e) {
40.         e.printStackTrace();
41.
42.     }
43.
44.     }
45.
46.     }
47.
48. //  原理也是通过 Java 的反射机制来做的。
49. //  2、 将一个 Map 对象转化为一个 Bean
50. //  这个 Map 对象的 key 必须与 Bean 的属性相对应。
51.     Map map = new HashMap();
52.     map.put("name","tom");
53.     map.put("email","tom@");
54.     map.put("age","21");
55.     //将 map 转化为一个 Person 对象
56.     Person person = new Person();
57.     BeanUtils.populate(person,map);
```

```
58. //  通过上面的一行代码，此时 person 的属性就已经具有了上面所赋的值了。
59. //  将一个 Bean 转化为一个 Map 对象了，如下：
60.     Map map = BeanUtils.describe(person)
```

## 2、Betwixt

XML 与 Java 对象之间相互转换。

[java] view plaincopy

```
1.  //1、 将 JavaBean 转为 XML 内容
2.      // 新创建一个 Person 类
3.      public class Person{
4.          private String name;
5.          private int age;
6.          /** Need to allow bean to be created via reflection */
7.          public PersonBean() {
8.          }
9.          public PersonBean(String name, int age) {
10.             this.name = name;
11.             this.age = age;
12.         }
13.         //省略 set, get 方法
14.         public String toString() {
15.             return "PersonBean[name='" + name + "',age='" + age + "']";
16.         }
17.     }
18.
19.     //再创建一个 WriteApp 类：
20.     import java.io.StringWriter;
21.     import org.apache.commons.betwixt.io.BeanWriter;
22.     public class WriteApp {
23.     /**
24.     * 创建一个例子 Bean，并将它转化为 XML.
25.     */
26.     public static final void main(String [] args) throws Exception {
27.         // 先创建一个 StringWriter，我们将把它写入为一个字符串
28.         StringWriter outputWriter = new StringWriter();
29.         // Betwixt 在这里仅仅是将 Bean 写入为一个片断
30.         // 所以如果要想完整的 XML 内容，我们应该写入头格式
```

```
31.         outputWriter.write("<?xml version='1.0' encoding='UTF-8' ?>\n");
32.         // 创建一个 BeanWriter，其将写入到我们预备的 stream 中
33.         BeanWriter beanWriter = new BeanWriter(outputWriter);
34.         // 配置 betwixt
35.         // 更多详情请参考 java docs 或最新的文档
36.         beanWriter.getXMLIntrospector().getConfiguration().setAttributesForP
    rimitives(false);
37.         beanWriter.getBindingConfiguration().setMapIDs(false);
38.         beanWriter.enablePrettyPrint();
39.         // 如果这个地方不传入 XML 的根节点名，Betwixt 将自己猜测是什么
40.         // 但是让我们将例子 Bean 名作为根节点吧
41.         beanWriter.write("person", new PersonBean("John Smith", 21));
42.         //输出结果
43.         System.out.println(outputWriter.toString());
44.         // Betwixt 写的是片断而不是一个文档，所以不要自动的关闭掉 writers 或者
    streams，
45.         //但这里仅仅是一个例子，不会做更多事情，所以可以关掉
46.         outputWriter.close();
47.         }
48.     }
49. //2、 将 XML 转化为 JavaBean
50.     import java.io.StringReader;
51.     import org.apache.commons.betwixt.io.BeanReader;
52.     public class ReadApp {
53.     public static final void main(String args[]) throws Exception{
54.         // 先创建一个 XML，由于这里仅是作为例子，所以我们硬编码了一段 XML 内容
55.         StringReader xmlReader = new StringReader(
56.         "<?xml version='1.0' encoding='UTF-
    8' ?> <person><age>25</age><name>James Smith</name></person>");
57.         //创建 BeanReader
58.         BeanReader beanReader  = new BeanReader();
59.         //配置 reader
60.         beanReader.getXMLIntrospector().getConfiguration().setAttributesForP
    rimitives(false);
61.         beanReader.getBindingConfiguration().setMapIDs(false);
62.         //注册 beans，以便 betwixt 知道 XML 将要被转化为一个什么 Bean
63.         beanReader.registerBeanClass("person", PersonBean.class);
64.         //现在我们对 XML 进行解析
65.         PersonBean person = (PersonBean) beanReader.parse(xmlReader);
66.         //输出结果
67.         System.out.println(person);
68.         }
69.     }
```

## 3、Codec

提供了一些公共的编解码实现，比如 Base64, Hex, MD5,Phonetic and URLs 等等。

[java] view plaincopy

```java
//Base64 编解码
private static String encodeTest(String str){
        Base64 base64 = new Base64();
        try {
            str = base64.encodeToString(str.getBytes("UTF-8"));
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
            System.out.println("Base64 编码后："+str);
        return str;
    }

    private static void decodeTest(String str){
        Base64 base64 = new Base64();
        //str = Arrays.toString(Base64.decodeBase64(str));
        str = new String(Base64.decodeBase64(str));
        System.out.println("Base64 解码后："+str);
    }
```

## 4、Collections

对 java.util 的扩展封装，处理数据还是挺灵活的。

org.apache.commons.collections – Commons Collections 自定义的一组公用的接口和工具类

org.apache.commons.collections.bag – 实现 Bag 接口的一组类

org.apache.commons.collections.bidimap – 实现 BidiMap 系列接口的一组类

org.apache.commons.collections.buffer – 实现 Buffer 接口的一组类

org.apache.commons.collections.collection – 实现 java.util.Collection 接口的一组类

org.apache.commons.collections.comparators – 实现 java.util.Comparator 接口的一组类

org.apache.commons.collections.functors – Commons Collections 自定义的一组功能类

org.apache.commons.collections.iterators – 实现 java.util.Iterator 接口的一组类

org.apache.commons.collections.keyvalue – 实现集合和键/值映射相关的一组类

org.apache.commons.collections.list – 实现 java.util.List 接口的一组类

org.apache.commons.collections.map – 实现 Map 系列接口的一组类

org.apache.commons.collections.set – 实现 Set 系列接口的一组类

[java] view plaincopy

```java
/**
        * 得到集合里按顺序存放的 key 之后的某一 Key
        */
        OrderedMap map = new LinkedMap();
        map.put("FIVE", "5");
        map.put("SIX", "6");
        map.put("SEVEN", "7");
        map.firstKey(); // returns "FIVE"
        map.nextKey("FIVE"); // returns "SIX"
        map.nextKey("SIX"); // returns "SEVEN"

        /**
        * 通过 key 得到 value
        * 通过 value 得到 key
        * 将 map 里的 key 和 value 对调
        */

        BidiMap bidi = new TreeBidiMap();
        bidi.put("SIX", "6");
        bidi.get("SIX");  // returns "6"
        bidi.getKey("6");  // returns "SIX"
```

```
        //       bidi.removeValue("6");  // removes the mapping
        BidiMap inverse = bidi.inverseBidiMap();  // returns a map with keys a
nd values swapped
        System.out.println(inverse);

        /**
         * 得到两个集合中相同的元素
         */
        List<String> list1 = new ArrayList<String>();
        list1.add("1");
        list1.add("2");
        list1.add("3");
        List<String> list2 = new ArrayList<String>();
        list2.add("2");
        list2.add("3");
        list2.add("5");
        Collection c = CollectionUtils.retainAll(list1, list2);
        System.out.println(c);
```

## 5、Compress

commons compress 中的打包、压缩类库。

[java] view plaincopy

```
//创建压缩对象
    ZipArchiveEntry entry = new ZipArchiveEntry("CompressTest");
      //要压缩的文件
      File f=new File("e:\\test.pdf");
      FileInputStream fis=new FileInputStream(f);
      //输出的对象 压缩的文件
      ZipArchiveOutputStream zipOutput=new ZipArchiveOutputStream(new File("
e:\\test.zip"));
      zipOutput.putArchiveEntry(entry);
      int i=0,j;
      while((j=fis.read()) != -1)
      {
       zipOutput.write(j);
       i++;
       System.out.println(i);
```

```
        }
        zipOutput.closeArchiveEntry();
        zipOutput.close();
        fis.close();
```

## 6、Configuration

用来帮助处理配置文件的，支持很多种存储方式。

1. Properties files
2. XML documents
3. Property list files (.plist)
4. JNDI
5. JDBC Datasource
6. System properties
7. Applet parameters
8. Servlet parameters

[java] view plaincopy

```java
//举一个 Properties 的简单例子
# usergui.properties
colors.background = #FFFFFF
colors.foreground = #000080
window.width = 500
window.height = 300

PropertiesConfiguration config = new PropertiesConfiguration("usergui.properties");
config.setProperty("colors.background", "#000000);
config.save();

config.save("usergui.backup.properties);//save a copy
Integer integer = config.getInteger("window.width");
```

## 7、DBCP

(Database Connection Pool)是一个依赖 Jakarta commons-pool 对象池机制的数据库连接池,Tomcat 的数据源使用的就是 DBCP。

[java] view plaincopy

```java
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;

import org.apache.commons.pool.ObjectPool;
import org.apache.commons.pool.impl.GenericObjectPool;
import org.apache.commons.dbcp.ConnectionFactory;
import org.apache.commons.dbcp.PoolingDataSource;
import org.apache.commons.dbcp.PoolableConnectionFactory;
import org.apache.commons.dbcp.DriverManagerConnectionFactory;
//官方示例
public class PoolingDataSources {

    public static void main(String[] args) {
        System.out.println("加载 jdbc 驱动");
        try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        } catch (ClassNotFoundException e) {
        e.printStackTrace();
        }
        System.out.println("Done.");
        //
        System.out.println("设置数据源");
        DataSource dataSource = setupDataSource("jdbc:oracle:thin:@localhost:1521:test");
        System.out.println("Done.");

        //
        Connection conn = null;
        Statement stmt = null;
        ResultSet rset = null;

        try {
        System.out.println("Creating connection.");
```

```
36.         conn = dataSource.getConnection();
37.         System.out.println("Creating statement.");
38.         stmt = conn.createStatement();
39.         System.out.println("Executing statement.");
40.         rset = stmt.executeQuery("select * from person");
41.         System.out.println("Results:");
42.         int numcols = rset.getMetaData().getColumnCount();
43.         while(rset.next()) {
44.         for(int i=0;i<=numcols;i++) {
45.         System.out.print("\t" + rset.getString(i));
46.         }
47.         System.out.println("");
48.         }
49.         } catch(SQLException e) {
50.         e.printStackTrace();
51.         } finally {
52.         try { if (rset != null) rset.close(); } catch(Exception e) { }
53.         try { if (stmt != null) stmt.close(); } catch(Exception e) { }
54.         try { if (conn != null) conn.close(); } catch(Exception e) { }
55.         }
56.         }
57.
58.     public static DataSource setupDataSource(String connectURI) {
59.         //设置连接地址
60.         ConnectionFactory connectionFactory = new DriverManagerConnectionFac
    tory(
61.                 connectURI, null);
62.
63.         // 创建连接工厂
64.         PoolableConnectionFactory poolableConnectionFactory = new PoolableCo
    nnectionFactory(
65.                 connectionFactory);
66.
67.         //获取 GenericObjectPool 连接的实例
68.         ObjectPool connectionPool = new GenericObjectPool(
69.                 poolableConnectionFactory);
70.
71.         // 创建 PoolingDriver
72.         PoolingDataSource dataSource = new PoolingDataSource(connectionPool);


73.
74.         return dataSource;
75.     }
76. }
```

## 8、DbUtils

Apache 组织提供的一个资源 JDBC 工具类库，它是对 JDBC 的简单封装，对传统操作数据库的类进行二次封装，可以把结果集转化成 List。，同时也不影响程序的性能。

DbUtils 类：启动类

ResultSetHandler 接口：转换类型接口

MapListHandler 类：实现类，把记录转化成 List

BeanListHandler 类：实现类，把记录转化成 List，使记录为 JavaBean 类型的对象

Qrery Runner 类：执行 SQL 语句的类

[java] view plaincopy

```java
import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.handlers.BeanListHandler;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.List;
//转换成 list
public class BeanLists {
    public static void main(String[] args) {
        Connection conn = null;
        String url = "jdbc:mysql://localhost:3306/ptest";
        String jdbcDriver = "com.mysql.jdbc.Driver";
        String user = "root";
        String password = "ptest";

        DbUtils.loadDriver(jdbcDriver);
        try {
            conn = DriverManager.getConnection(url, user, password);
            QueryRunner qr = new QueryRunner();
            List results = (List) qr.query(conn, "select id,name from person
", new BeanListHandler(Person.class));
            for (int i = 0; i < results.size(); i++) {
                Person p = (Person) results.get(i);
```

```
                System.out.println("id:" + p.getId() + ",name:" + p.getName())
;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            DbUtils.closeQuietly(conn);
        }
    }
}

public class Person{
    private Integer id;
    private String name;

   //省略 set, get 方法
}

import org.apache.commons.dbutils.DbUtils;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.handlers.MapListHandler;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import java.util.List;
import java.util.Map;
//转换成 map
public class MapLists {
    public static void main(String[] args) {
        Connection conn = null;
        String url = "jdbc:mysql://localhost:3306/ptest";
        String jdbcDriver = "com.mysql.jdbc.Driver";
        String user = "root";
        String password = "ptest";

        DbUtils.loadDriver(jdbcDriver);
        try {
            conn = DriverManager.getConnection(url, user, password);
            QueryRunner qr = new QueryRunner();
            List results = (List) qr.query(conn, "select id,name from person
", new MapListHandler());
            for (int i = 0; i < results.size(); i++) {
```

```
                Map map = (Map) results.get(i);
                System.out.println("id:" + map.get("id") + ",name:" + map.get("name"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            DbUtils.closeQuietly(conn);
        }
    }
}
```

## 9、Email

提供的一个开源的 API，是对 javamail 的封装。

[java] view plaincopy

```
//用 commons email 发送邮件
public static void main(String args[]){
        Email email = new SimpleEmail();
        email.setHostName("smtp.googlemail.com");
        email.setSmtpPort(465);
        email.setAuthenticator(new DefaultAuthenticator("username", "password"));
        email.setSSLOnConnect(true);
        email.setFrom("user@gmail.com");
        email.setSubject("TestMail");
        email.setMsg("This is a test mail ... :-)");
        email.addTo("foo@bar.com");
        email.send();
    }
```

## 10、FileUpload

java web 文件上传功能。

[java] view plaincopy

```
//官方示例：
//* 检查请求是否含有上传文件
    // Check that we have a file upload request
    boolean isMultipart = ServletFileUpload.isMultipartContent(request);
```

```
5.
6.      //现在我们得到了 items 的列表
7.
8.      //如果你的应用近于最简单的情况，上面的处理就够了。但我们有时候还是需要更多的控制。
9.      //下面提供了几种控制选择：
10.     // Create a factory for disk-based file items
11.     DiskFileItemFactory factory = new DiskFileItemFactory();
12.
13.     // Set factory constraints
14.     factory.setSizeThreshold(yourMaxMemorySize);
15.     factory.setRepository(yourTempDirectory);
16.
17.     // Create a new file upload handler
18.     ServletFileUpload upload = new ServletFileUpload(factory);
19.
20.     // 设置最大上传大小
21.     upload.setSizeMax(yourMaxRequestSize);
22.
23.     // 解析所有请求
24.     List /* FileItem */ items = upload.parseRequest(request);
25.
26.     // Create a factory for disk-based file items
27.     DiskFileItemFactory factory = new DiskFileItemFactory(
28.             yourMaxMemorySize, yourTempDirectory);
29.
30.     //一旦解析完成，你需要进一步处理 item 的列表。
31.     // Process the uploaded items
32.     Iterator iter = items.iterator();
33.     while (iter.hasNext()) {
34.         FileItem item = (FileItem) iter.next();
35.
36.         if (item.isFormField()) {
37.             processFormField(item);
38.         } else {
39.             processUploadedFile(item);
40.         }
41.     }
42.
43.     //区分数据是否为简单的表单数据，如果是简单的数据：
44.     // processFormField
45.     if (item.isFormField()) {
46.         String name = item.getFieldName();
47.         String value = item.getString();
```

```
48.         //...省略步骤
49.     }
50.
51.     //如果是提交的文件：
52.     // processUploadedFile
53.     if (!item.isFormField()) {
54.         String fieldName = item.getFieldName();
55.         String fileName = item.getName();
56.         String contentType = item.getContentType();
57.         boolean isInMemory = item.isInMemory();
58.         long sizeInBytes = item.getSize();
59.         //...省略步骤
60.     }
61.
62.     //对于这些 item，我们通常要把它们写入文件，或转为一个流
63.     // Process a file upload
64.     if (writeToFile) {
65.         File uploadedFile = new File(...);
66.         item.write(uploadedFile);
67.     } else {
68.         InputStream uploadedStream = item.getInputStream();
69.         //...省略步骤
70.         uploadedStream.close();
71.     }
72.
73.     //或转为字节数组保存在内存中：
74.     // Process a file upload in memory
75.     byte[] data = item.get();
76.     //...省略步骤
77.     //如果这个文件真的很大，你可能会希望向用户报告到底传了多少到服务端，让用户了解
    上传的过程
78.     //Create a progress listener
79.     ProgressListener progressListener = new ProgressListener(){
80.        public void update(long pBytesRead, long pContentLength, int pItems)
    {
81.            System.out.println("We are currently reading item " + pItems);
82.            if (pContentLength == -1) {
83.                System.out.println("So far, " + pBytesRead + " bytes have bee
    n read.");
84.            } else {
85.                System.out.println("So far, " + pBytesRead + " of " + pConten
    tLength
86.                                   + " bytes have been read.");
87.            }
```

```
88.         }
89.     };
90.     upload.setProgressListener(progressListener);
```

## 11、HttpClient

基于 HttpCore 实 现的一个 HTTP/1.1 兼容的 HTTP 客户端，它提供了一系列可重用的客户端身份验证、HTTP 状态保持、HTTP 连接管理 module。

[java] view plaincopy

```
1.  //GET 方法
2.  import java.io.IOException;
3.  import org.apache.commons.httpclient.*;
4.  import org.apache.commons.httpclient.methods.GetMethod;
5.  import org.apache.commons.httpclient.params.HttpMethodParams;
6.
7.  public class GetSample{
8.      public static void main(String[] args) {
9.          // 构造 HttpClient 的实例
10.         HttpClient httpClient = new HttpClient();
11.         // 创建 GET 方法的实例
12.         GetMethod getMethod = new GetMethod("http://www.ibm.com");
13.         // 使用系统提供的默认的恢复策略
14.         getMethod.getParams().setParameter(HttpMethodParams.RETRY_HANDLER,
15.                 new DefaultHttpMethodRetryHandler());
16.         try {
17.             // 执行 getMethod
18.             int statusCode = httpClient.executeMethod(getMethod);
19.             if (statusCode != HttpStatus.SC_OK) {
20.                 System.err.println("Method failed: "
21.                         + getMethod.getStatusLine());
22.             }
23.             // 读取内容
24.             byte[] responseBody = getMethod.getResponseBody();
25.             // 处理内容
26.             System.out.println(new String(responseBody));
27.         } catch (HttpException e) {
28.             // 发生致命的异常，可能是协议不对或者返回的内容有问题
29.             System.out.println("Please check your provided http address!");
30.             e.printStackTrace();
```

```
        } catch (IOException e) {
            // 发生网络异常
            e.printStackTrace();
        } finally {
            // 释放连接
            getMethod.releaseConnection();
        }
    }
}

//POST 方法
import java.io.IOException;
import org.apache.commons.httpclient.*;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.params.HttpMethodParams;

public class PostSample{
    public static void main(String[] args) {
        // 构造 HttpClient 的实例
        HttpClient httpClient = new HttpClient();
        // 创建 POST 方法的实例
        String url = "http://www.oracle.com/";
        PostMethod postMethod = new PostMethod(url);
        // 填入各个表单域的值
        NameValuePair[] data = { new NameValuePair("id", "youUserName"),
        new NameValuePair("passwd", "yourPwd") };
        // 将表单的值放入 postMethod 中
        postMethod.setRequestBody(data);
        // 执行 postMethod
        int statusCode = httpClient.executeMethod(postMethod);
        // HttpClient 对于要求接受后继服务的请求，象 POST 和 PUT 等不能自动处理转发

        // 301 或者 302
        if (statusCode == HttpStatus.SC_MOVED_PERMANENTLY ||
        statusCode == HttpStatus.SC_MOVED_TEMPORARILY) {
            // 从头中取出转向的地址
            Header locationHeader = postMethod.getResponseHeader("location");

            String location = null;
            if (locationHeader != null) {
             location = locationHeader.getValue();
             System.out.println("The page was redirected to:" + location);

            } else {
```

```
72.              System.err.println("Location field value is null.");
73.             }
74.             return;
75.         }
76.     }
77. }
```

## 12、IO

对 java.io 的扩展 操作文件非常方便。

[java] view plaincopy

```
1.  //1. 读取 Stream
2.
3.  //标准代码:
4.  InputStream in = new URL( "http://jakarta.apache.org" ).openStream();
5.  try {
6.          InputStreamReader inR = new InputStreamReader( in );
7.          BufferedReader buf = new BufferedReader( inR );
8.          String line;
9.          while ( ( line = buf.readLine() ) != null ) {
10.            System.out.println( line );
11.         }
12.   } finally {
13.     in.close();
14.   }
15.
16. //使用 IOUtils
17.
18. InputStream in = new URL( "http://jakarta.apache.org" ).openStream();
19. try {
20.     System.out.println( IOUtils.toString( in ) );
21. } finally {
22.     IOUtils.closeQuietly(in);
23. }
24.
25. //2. 读取文件
26. File file = new File("/commons/io/project.properties");
27. List lines = FileUtils.readLines(file, "UTF-8");
28. //3. 察看剩余空间
29. long freeSpace = FileSystemUtils.freeSpace("C:/");
```

## 13、Lang

主要是一些公共的工具集合，比如对字符、数组的操作等等。

Packages
org.apache.commons.lang
org.apache.commons.lang.builder
org.apache.commons.lang.enum
org.apache.commons.lang.enums
org.apache.commons.lang.exception
org.apache.commons.lang.math
org.apache.commons.lang.mutable
org.apache.commons.lang.reflect
org.apache.commons.lang.text
org.apache.commons.lang.time

[java] view plaincopy

```java
// 1 合并两个数组: org.apache.commons.lang. ArrayUtils
    // 有时我们需要将两个数组合并为一个数组，用 ArrayUtils 就非常方便，示例如下：

    private static void testArr() {
        String[] s1 = new String[] { "1", "2", "3" };
        String[] s2 = new String[] { "a", "b", "c" };
        String[] s = (String[]) ArrayUtils.addAll(s1, s2);
        for (int i = 0; i < s.length; i++) {
            System.out.println(s[i]);
        }
        String str = ArrayUtils.toString(s);
        str = str.substring(1, str.length() - 1);
        System.out.println(str + ">>" + str.length());

    }
    //2 截取从 from 开始字符串
    StringUtils.substringAfter("SELECT * FROM PERSON ", "from");
    //3 判断该字符串是不是为数字(0~9)组成，如果是，返回 true 但该方法不识别有小数点和 请注意。
    StringUtils.isNumeric("454534"); //返回 true
    //4.取得类名
       System.out.println(ClassUtils.getShortClassName(Test.class));
       //取得其包名
       System.out.println(ClassUtils.getPackageName(Test.class));

```

```
        //5.NumberUtils
        System.out.println(NumberUtils.stringToInt("6"));
        //6.五位的随机字母和数字
        System.out.println(RandomStringUtils.randomAlphanumeric(5));
        //7.StringEscapeUtils
        System.out.println(StringEscapeUtils.escapeHtml("<html>"));
        //输出结果为<html>
        System.out.println(StringEscapeUtils.escapeJava("String"));

        //8.StringUtils,判断是否是空格字符
        System.out.println(StringUtils.isBlank("    "));
        //将数组中的内容以,分隔
        System.out.println(StringUtils.join(test,","));
        //在右边加下字符,使之总长度为 6
        System.out.println(StringUtils.rightPad("abc", 6, 'T'));
        //首字母大写
        System.out.println(StringUtils.capitalize("abc"));
        //Deletes all whitespaces from a String 删除所有空格
        System.out.println( StringUtils.deleteWhitespace("   ab  c  "));
        //判断是否包含这个字符
        System.out.println( StringUtils.contains("abc", "ba"));
        //表示左边两个字符
        System.out.println( StringUtils.left("abc", 2));
        System.out.println(NumberUtils.stringToInt("33"));
```

## 14、Logging

提供的是一个 Java 的日志接口,同时兼顾轻量级和不依赖于具体的日志实现工具。

[java] view plaincopy

```
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

    public class CommonLogTest {
     private static Log log = LogFactory.getLog(CommonLogTest.class);
     //日志打印
     public static void main(String[] args) {
         log.error("ERROR");
         log.debug("DEBUG");
         log.warn("WARN");
```

```
11.          log.info("INFO");
12.          log.trace("TRACE");
13.       System.out.println(log.getClass());
14.      }
15.
16.    }
```

## 15、Validator

通用验证系统，该组件提供了客户端和服务器端的数据验证框架。

验证日期

[java] view plaincopy

```
1.  // 获取日期验证
2.        DateValidator validator = DateValidator.getInstance();
3.
4.        // 验证/转换日期
5.        Date fooDate = validator.validate(fooString, "dd/MM/yyyy");
6.        if (fooDate == null) {
7.            // 错误 不是日期
8.            return;
9.        }
```

表达式验证

[java] view plaincopy

```
1.  // 设置参数
2.        boolean caseSensitive = false;
3.        String regex1    = "^([A-Z]*)(?:\\-)([A-Z]*)*$"
4.        String regex2    = "^([A-Z]*)$";
5.        String[] regexs = new String[] {regex1, regex1};
6.
7.        // 创建验证
```

```
        RegexValidator validator = new RegexValidator(regexs, caseSensitive);


        // 验证返回 boolean
        boolean valid = validator.isValid("abc-def");

        // 验证返回字符串
        String result = validator.validate("abc-def");

        // 验证返回数组
        String[] groups = validator.match("abc-def");
```

配置文件中使用验证

[html] view plaincopy

```
<form-validation>
    <global>
        <validator name="required"
            classname="org.apache.commons.validator.TestValidator"
            method="validateRequired"
            methodParams="java.lang.Object, org.apache.commons.validator.Field
"/>
    </global>
    <formset>
    </formset>
</form-validation>

添加姓名验证.

<form-validation>
    <global>
        <validator name="required"
            classname="org.apache.commons.validator.TestValidator"
            method="validateRequired"
            methodParams="java.lang.Object, org.apache.commons.validator.Field
"/>
    </global>
    <formset>
        <form name="nameForm">
            <field property="firstName" depends="required">
                <arg0 key="nameForm.firstname.displayname"/>
            </field>
            <field property="lastName" depends="required">
```

```
            <arg0 key="nameForm.lastname.displayname"/>
          </field>
        </form>
    </formset>
</form-validation>
```

验证类

[java] view plaincopy

```
 Excerpts from org.apache.commons.validator.RequiredNameTest
//加载验证配置文件
InputStream in = this.getClass().getResourceAsStream("validator-name-required.xml");

ValidatorResources resources = new ValidatorResources(in);
//这个是自己创建的 bean 我这里省略了
Name name = new Name();

Validator validator = new Validator(resources, "nameForm");
//设置参数
validator.setParameter(Validator.BEAN_PARAM, name);


Map results = null;
//验证
results = validator.validate();

if (results.get("firstName") == null) {
    //验证成功
} else {
    //有错误     int errors = ((Integer)results.get("firstName")).intValue();
```