链滴

Jodd 介绍

作者：zsr251

原文链接：https://ld246.com/article/1516634257785

来源网站：链滴

Jodd是一个普通开源Java包。你可以把Jodd想象成Java的"瑞士军刀",不仅小，锋利而且包含许多便的功能。

Jodd 提供的功能有：

1. 提供操作Java bean，
2. 可以从各种数据源加载Bean,
3. 简化JDBC的接连与代码，
4. 剖析SQL查询，
5. 处理时间与日期，
6. 操作与格式化String,
7. 搜索本地硬盘上的文件，
8. 帮助处理Servlet请求等。
9. Madvoc - 一个简单的MVC框架，用CoC和注解的方式
10. Petite：一个轻量级的DI（注入）框架
11. Proxetta：一个高效的动态代理框架
12. Db & DbOom：高效，轻量级的数据库处理框架
13. Paramo：在运行时，非常简单的获取方法和构造方法的参数
14. JTX：提供一个独立的，轻量级的事务管理器
15. VTor：实用的验证框架，可以针对任何Java 对象
16. Lagarto：高效，灵活的通用HTML解析器
17. Decora：基于模板的页面装饰框架。
18. Jerry：友好的jQuery java解析器，支持CSS选择器

Jodd = Tools + IoC + MVC + DB + AOP + TX + JSON + HTML < 1.5 Mb

Jodd 被分成众多模块，按需选择，其中

工具类模块有：

- jodd-core 一些工具类，包括 Printf、JDateTime、StringUtil、Fast buffers等等
- jodd-bean BeanUtil以及类型检查转换工具
- jodd-props 更强大的Java Properties替代
- jodd-email 更简单易用的e-mail收发
- jodd-upload 处理HTTP上传
- jodd-servlets 一些Servlet相关的工具类, 附带一套漂亮的JSP标签库
- jodd-http 轻巧的HTTP客户端

小型框架模块有：

- jodd-madvoc 一个MVC框架
- jodd-petite 一个依赖注入容器

原文链接：Jodd 介绍

- jodd-lagarto　HTML/XML解析器，包含Jerry和CSSelly，让你像jQuery一样筛选HTML节点
- jodd-lagarto-web Lagarto的Web扩展，包含　Decora、HtmlStapler等等
- jodd-proxetta　帮你实现动态代理，获取函数参数名
- jodd-dboom　数据库访问的轻量级封装，可看作一个简单的ORM
- jodd-json　JSON解析、序列化
- jodd-vtor　一个基于注解的字段验证框架

Jodd使用示例

```
 package org.xiaochen.test.jodd;
 import java.util.TimeZone;
 import org.apache.log4j.Logger;
 import org.junit.Test;
 import jodd.datetime.JDateTime;
/**
* JODD中的时间操作类
* @author DJZHOU
*
 */
 public class JDateTimeUtil {
 private Logger log = Logger.getLogger(JDateTimeUtil.class) ;
 @Test
 public void testConstructor()
 {
 /*
 * 构造函数的使用
 */
 JDateTime jdt = new JDateTime();       // set current date and time
 jdt = new JDateTime(2012, 12, 21);     // set 21st December 2012, midnight
 jdt = new JDateTime(System.currentTimeMillis());
 jdt = new JDateTime(2012, 12, 21, 11, 54, 22, 124); // set 21st December 2012, 11:54:22.124
 jdt = new JDateTime("2012-12-21 11:54:22.124");    // -//-
 jdt = new JDateTime("12/21/2012", "MM/DD/YYYY");   // set 21st December 2012, midnight
 }
 @Test
 public void testSet()
 {
 JDateTime jdt = new JDateTime();       // set current date and time
 /*
 * set方法的使用:设定日期时间
 */
 jdt.set(2012, 12, 21, 11, 54, 22, 124);    // set 21st December 2012, 11:54:22.124
 jdt.set(2012, 12, 21);                  // set 21st December 2012, midnight
 jdt.setDate(2012, 12, 21);                // change date to 21st December 2012, do not change te
time
 jdt.setCurrentTime();                   // set current date and time
 jdt.setYear(1973);                     // change the year
 jdt.setHour(22);                      // change the hour
 jdt.setTime(18, 00, 12, 853);
 }
 @Test
 public void testGet()
```

原文链接：Jodd 介绍

```
{
JDateTime jdt = new JDateTime();       // set current date and time
/*
* get方法的使用:读取日期和时间
*/
jdt.getYear();
jdt.getDateTimeStamp();
log.warn(jdt.getDateTimeStamp());//获取当前时间
log.warn(jdt.getJulianDate());
log.warn(jdt.getDay()) ;
log.warn(jdt.getDayOfMonth()) ;
log.warn(jdt.getDayOfWeek()) ;
log.warn(jdt.getDayOfYear()) ;
log.warn(jdt.getEra()) ;
log.warn(jdt.getFirstDayOfWeek()) ;
log.warn(jdt.getFormat()) ;
}
@Test
public void testAdd()
{
JDateTime jdt = new JDateTime();       // set current date and time
jdt.add(1, 2, 3, 4, 5, 6, 7);   // add 1 year, 2 months, 3 days, 4 hours...
jdt.add(4, 2, 0);          // add 4 years and 2 months
jdt.addMonth(-120);        // go back 120 months
jdt.subYear(1);            // go back one year
jdt.addHour(1234);         // add 1234 hours
}
@Test
public void testAdd2()
{
JDateTime jdt = new JDateTime() ;
log.warn(jdt.toString("YYYY-MM-DD")) ;
jdt.addDay(-20) ;
log.warn(jdt.toString("YYYY-MM-DD")) ;
jdt.addDay(10, true) ;
log.warn(jdt.toString("YYYY-MM-DD")) ;
jdt.addYear(1);
log.warn(jdt.toString("YYYY-MM-DD")) ;
}
@Test
public void testFormat()
{
JDateTime jdt = new JDateTime();       // set current date and time
/**
* 转换说明
YYYY   年
MM     月
DD     日
D      一周中的第几天 从周一算起
MML    月,长型
MMS    月,短行
DL     一周中的第几天 长型 从周一算起
DS     一周中的第几天 短型 从周一算起
hh     小时
```

```
mm     分钟
ss     秒
mss    毫秒
 DDD    一年中的第几天
 WW     一年中的第几周
 WWW    一年中的第几周并用W标识
 W      一个月中的第几周
 E      年代,公元前还是公元后
 TZL    时间长型
 TZS    时间短型
 *
 */
 log.warn(jdt.convertToDate()) ;
 log.warn(jdt.toString("YYYY-MM-DD"));
 log.warn(jdt.toString("YYYY-MM-DD hh:mm:ss"));//转换成字符串
 log.warn(jdt.toString("WW"));//本年度第几周
 log.warn(jdt.toString("YYYY")) ;
 }
 }
```

JODD操作properties文件

```
 package org.xiaochen.test.jodd;
 import java.io.File;
 import java.io.IOException;
 import org.apache.log4j.Logger;
 import org.junit.Test;
 import jodd.props.Props;
 /**
 * JODD操作properties文件
 * @author DJZHOU
 *
 */
 public class PropUtil {
 private static Logger log = Logger.getLogger(PropUtil.class) ;
 @Test
 public void propExam(){
 /*
 * 读取prop文件中的属性值
 */
 Props p = new Props();
 log.warn(URLUtil.getClassPath(this.getClass())+"/a.properties") ;
 try {
 p.load(new File(URLUtil.getClassPath(this.getClass())+"/a.properties"));
 } catch (IOException e) {
 e.printStackTrace();
 }
 String story = p.getValue("a");
 log.warn(p.getBaseValue("a")) ;
 log.warn(story);
 log.warn(null == p.getValue("a")) ;
 log.warn(p.toString()) ;
 p.setValue("c", "cc") ;
 }
```

原文链接：Jodd 介绍

```
}
```

JODD操作email类

```
package org.xiaochen.test.jodd;
import java.io.File;
import java.io.IOException;
import org.junit.Test;
import jodd.io.FileUtil;
import jodd.mail.Email;
import jodd.mail.EmailAttachment;
import jodd.mail.EmailMessage;
import jodd.mail.SendMailSession;
 import jodd.mail.SmtpSslServer;
 import jodd.mail.att.ByteArrayAttachment;
 import jodd.mail.att.FileAttachment;
 import jodd.util.MimeTypes;
 /**
 * JODD操作email类
 *
 * @author DJZHOU
 *
 */
 public class EmailUtil
 {
 public static void main(String[] args)
 {
 Email email = Email.create();
 email.addMessage(new EmailMessage("消息"));
 email.addText("邮件内容");
 email.embedFile(new File("d:/console.txt"));
 email.from("771842634@qq.com").to("771842634@qq.com");
 email.subject("主题");
 SendMailSession mailSession = new SmtpSslServer("smtp.qq.com//发送端邮箱服务器协议", "
送端QQ邮箱", "发送端QQ邮箱密码").createSession();
 mailSession.open();
 mailSession.sendMail(email);
 mailSession.close();
 System.out.println("发送成功!...");
 }
 @Test
 public void test() throws IOException
 {
 Email email = new Email();
 email.setFrom("infoxxx@jodd.org");
 email.setTo("igorxxxx@gmail.com");
 email.setSubject("test7");
 EmailMessage textMessage = new EmailMessage("Hello!", MimeTypes.MIME_TEXT_PLAIN);
 email.addMessage(textMessage);
 EmailMessage htmlMessage = new EmailMessage(
 "" +
 "Hey!jodd使用示例Hay!",
 MimeTypes.MIME_TEXT_HTML);
 email.addMessage(htmlMessage);
```

```
 EmailAttachment embeddedAttachment =
 new ByteArrayAttachment(FileUtil.readBytes("d:\\c.png"), "image/png", "c.png", "c.png");
 email.attach(embeddedAttachment);
 EmailAttachment attachment = new FileAttachment(new File("d:\\b.jpg"), "b.jpg", "image/jp
g");
 email.attach(attachment);
 }
 }
```

String字符串的操作工具类

```
 package org.xiaochen.test.jodd;
  import org.junit.Test;
  import jodd.util.StringUtil;
  /**
  * String字符串的操作工具类,太强大以至于我要发疯
  *
  * @author DJZHOU
  *
  */
   public class StringExamUtil
   {
   @Test
   public void stringExam()
   {
   String exam = "abcdefg10101010abcdefg";
   String result = "";
   /*
   * replace 字符替换
   */
   // 将字符串exam中的a替换成b
   result = StringUtil.replace(exam, "a", "b");
   // 将字符串exam中的a替换成8,b替换成9
   result = StringUtil.replace(exam, new String[] { "a", "b" }, new String[] { "8", "9" });
   // 将字符串exam中的a替换成b 这里是替换字符
   result = StringUtil.replaceChar(exam, 'a', 'b');
   // 将字符串exam中的a替换成8,b替换成9 这里是替换字符
   result = StringUtil.replaceChars(exam, new char[] { 'a', 'b' }, new char[] { '8', '9' });
   // 将字符串exam中的第一个a替换成b
   result = StringUtil.replaceFirst(exam, "a", "b");
   // 将字符串exam中的第一个a替换成b 这里是替换字符
   result = StringUtil.replaceFirst(exam, 'a', 'b');
   // 将字符串exam中的最后一个a替换成b
   result = StringUtil.replaceLast(exam, "a", "b");
   // 将字符串exam中的最后一个a替换成b 这里是替换字符
   result = StringUtil.replaceLast(exam, 'a', 'b');
   // 将字符串exam中的a和A替换成FF b和B替换成gg 即忽略大小写
   result = StringUtil.replaceIgnoreCase(exam, new String[] { "a", "b" }, new String[] { "FF", "gg
});
   /*
   * remove 字符移除
   */
   // 将字符串exam中的a移除
   result = StringUtil.remove(exam, "a");
```

```
    // 将字符串exam中的a移除 移除的是字符
    result = StringUtil.remove(exam, 'a');
    // 将字符串exam中的a b移除 移除的是字符 最后一个参数为无限参数
    result = StringUtil.removeChars(exam, 'a', 'b');
    // 将字符串exam中的a移除
    result = StringUtil.removeChars(exam, "a");
    /*
    * 判断字符串是否为空
    */
    // 判断字符串exam是否为空
    System.out.println(StringUtil.isEmpty(exam));
    // 判断字符串exam是否不为空
    System.out.println(StringUtil.isNotEmpty(exam));
    // 判断字符串exam是否为空 这里的空为去掉空格之后
    System.out.println(StringUtil.isBlank("  "));
    // 判断字符串exam是否不为空 这里的空为去掉空格之后
    System.out.println(StringUtil.isNotBlank("  "));
    // 判断字符串(无限参数)是否都为空 他们之间的关系为并且
    System.out.println(StringUtil.isAllEmpty(exam, " ", "null"));
    // 判断字符串(无限参数)是否都为空 这里的空为去掉空格之后 他们之间的关系为并且
    System.out.println(StringUtil.isAllBlank(exam, " ", "null"));
    // 对比字符串exam中的第4索引的字符是不是d
    System.out.println(StringUtil.isCharAtEqual(exam, 4, 'd'));
    // 对比字符串exam中的第4索引的字符是不是 不是d
    System.out.println(StringUtil.isCharAtEscaped(exam, 4, 'd'));
    /*
    * equals安全的字符串对比是否相等 不需要考虑null.equals等问题
    */
    // 判断字符串exam与aaa是否相等
    System.out.println(StringUtil.equals(exam, "aaa"));
    // 判断两个数组是否相等
    System.out.println(StringUtil.equals(new String[] { "aaa" }, new String[] { "aaa", "bbb" }));
    // 判断两个数组是否相等 且忽略大小写
    System.out.println(StringUtil.equalsIgnoreCase(new String[] { "aaa" }, new String[] { "aaa", "
bb" }));
    // 获取字符串bbb在数组中的索引
    System.out.println(StringUtil.equalsOne("bbb", new String[] { "aaa", "bbb" }));
    // 获取字符串bbb在数组中的索引 且忽略大小写
    System.out.println(StringUtil.equalsOneIgnoreCase("bbb", new String[] { "aaa", "bbb" }));
    /*
    * 首字母的更改
    */
    // 首字母大写
    result = StringUtil.capitalize(exam);
    // 首字母小写
    result = StringUtil.uncapitalize(exam);
    /*
    * split字符串分割
    */
    // 将字符串按 , 分割
    String[] array = StringUtil.split("1,2,3,4,5,6,7,8", ",");
    /*
    * indexOf 获取字符串中的字符索引
    */
```

```
/*
* Strips, crops, trims and cuts
*/
// 若这个字符串以a为开头,则去掉a
 result = StringUtil.stripLeadingChar(exam, 'a');
 // 若这个字符串以g为结尾,则去掉g
 result = StringUtil.stripTrailingChar(exam, 'g');
 // 若该字符串为"" 则返回null 若不是则返回字符串
 result = StringUtil.crop("");
 // 裁剪数组 将""变成null
 StringUtil.cropAll(new String[] { "", " " });
 // 去掉字符串两边的空格
 result = StringUtil.trimDown(" aa ");
 // 去掉字符串左边的空格
 result = StringUtil.trimLeft(" aa ");
 // 去掉字符串右边的空格
 result = StringUtil.trimRight(" aa ");
 // 去掉字符串右边的空格
 String[] array2 = new String[] { " aa ", " b b" };
 /*
 * 去掉数组内空格
 */
 StringUtil.trimAll(array2);
 StringUtil.trimDownAll(array2);
 for (String string : array2)
 {
 System.out.println(string);
 }
 /*
 * 切割字符串
 */
 // 从字符串的f字符开始切割字符串 保留f
 result = StringUtil.cutFromIndexOf(exam, 'f');
 // 从字符串的fg字符串开始切割字符串 保留fg
 result = StringUtil.cutFromIndexOf(exam, "fg");
 // 检查字符串是否为abc开头,若为此开头,则切割掉abc
 result = StringUtil.cutPrefix(exam, "abc");
 // 检查字符串是否为efg结尾,若为此结尾,则切割掉efg
 result = StringUtil.cutSuffix(exam, "efg");
 // 检查字符串是否为efg开头或结尾,若为此开头或结尾,则切割掉efg
 result = StringUtil.cutSurrounding(exam, "efg");
 // 检查字符串是否为abc开头efg结尾,若为为abc开头efg结尾,则切割掉
 result = StringUtil.cutSurrounding(exam, "abc", "efg");
 // 截取到字符串的f字符开始切割字符串 不保留f
 result = StringUtil.cutToIndexOf(exam, 'f');
 // 截取到字符串的fg字符串开始切割字符串 不保留fg
 result = StringUtil.cutToIndexOf(exam, "fg");
 /*
 * 其他很多小巧的方法,可以自行研究
 */
 System.out.println(result);
 }
 }
```