四川奔腾信息产业公司

**附件\_EC文档**

|  |  |
| --- | --- |
| 编 制 |  |
| 审 核 |  |
| 批 准 |  |

二〇〇九年七月十七日

修改历史

|  |  |  |  |
| --- | --- | --- | --- |
| 日 期 | 作者/修订者 | 版本 | 修改原因 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目 录

[前言 2](#_Toc235604493)

[Chapter 1. 配置 3](#_Toc235604494)

[1.1. 先决条件 3](#_Toc235604495)

[1.2. 安装 4](#_Toc235604496)

[1.3. 导出过滤器(可选) 5](#_Toc235604497)

[1.4. 安装测试 6](#_Toc235604498)

[Chapter 2. 概述 6](#_Toc235604499)

[2.1. 引言 6](#_Toc235604500)

[Chapter 3. TableTag 7](#_Toc235604501)

[3.1. 引言 7](#_Toc235604502)

[3.2. 显示图片 9](#_Toc235604503)

[3.3. 过滤、排序和动作（Action） 10](#_Toc235604504)

[3.4. 设定每页显示行数 10](#_Toc235604505)

[3.5. 样式 11](#_Toc235604506)

[3.6. 保存表的状态 11](#_Toc235604507)

[3.7. 其他属性 12](#_Toc235604508)

[3.8. 扩展属性 12](#_Toc235604509)

[Chapter 4. RowTag 13](#_Toc235604510)

[4.1. 引言 13](#_Toc235604511)

[4.2. 高亮显示行 14](#_Toc235604512)

[4.3. 样式 14](#_Toc235604513)

[4.4. Javascript属性 15](#_Toc235604514)

[4.5. 扩展属性 15](#_Toc235604515)

[Chapter 5. ColumnTag 15](#_Toc235604516)

[5.1. 引言 15](#_Toc235604517)

[5.2. Cell 18](#_Toc235604518)

[5.3. Filter Cell 20](#_Toc235604519)

[5.4. Header Cell 20](#_Toc235604520)

[5.5. 样式 21](#_Toc235604521)

[5.6. 解析和格式化 21](#_Toc235604522)

[5.7. 过滤和排序 22](#_Toc235604523)

[5.8. Calc 22](#_Toc235604524)

[5.9. 允许和不允许视图 22](#_Toc235604525)

[5.10. 其他属性 23](#_Toc235604526)

[5.11. 扩展属性 24](#_Toc235604527)

[Chapter 6. ParameterTag 25](#_Toc235604528)

[6.1. 引言 25](#_Toc235604529)

[Chapter 7. ExportTag 26](#_Toc235604530)

[7.1. 引言 26](#_Toc235604531)

[7.2. 扩展属性 27](#_Toc235604532)

[Chapter 8. Callbacks 29](#_Toc235604533)

[8.1. 引言 29](#_Toc235604534)

[8.2. RetrieveRowsCallback 29](#_Toc235604535)

[8.3. FilterRowsCallback 30](#_Toc235604536)

[8.4. SortRowsCallback 30](#_Toc235604537)

[Chapter 9. 视图 31](#_Toc235604538)

[9.1. 引言 31](#_Toc235604539)

[9.2. 表视图 31](#_Toc235604540)

[9.3. 导出视图 32](#_Toc235604541)

[Chapter 10. Preferences 32](#_Toc235604542)

[10.1. 引言 32](#_Toc235604543)

[10.2. TableTag 33](#_Toc235604544)

[10.3. ColumnTag 33](#_Toc235604545)

[10.4. Advanced Techniques 34](#_Toc235604546)

[Chapter 11. Messages 35](#_Toc235604547)

[11.1. 资源绑定 35](#_Toc235604548)

[11.2. 全局资源 36](#_Toc235604549)

[11.3. TableTag 37](#_Toc235604550)

[11.4. ColumnTag 37](#_Toc235604551)

[Chapter 12. Limit 38](#_Toc235604552)

[12.1. Limit指南 38](#_Toc235604553)

[12.2. 创建Limit 39](#_Toc235604554)

[12.3. Filter和Sort属性 39](#_Toc235604555)

[12.4. 设置页和行属性 40](#_Toc235604556)

[12.5. Setup 40](#_Toc235604557)

[Chapter 13. AutoGenerateColumns 41](#_Toc235604558)

[13.1. 引言 41](#_Toc235604559)

[13.2. ColumnsTag 41](#_Toc235604560)

[Chapter 14. Utilities 43](#_Toc235604561)

[14.1. 引言 43](#_Toc235604562)

[14.2. HtmlBuilder 43](#_Toc235604563)

[Chapter 15. 1.0.1-M4升级说明 43](#_Toc235604564)

[15.1. 变更概述 43](#_Toc235604565)

## 前言

eXtremeComponents是一系列提供高级显示的开源JSP定制标签。当前的包含 的组件为eXtremeTable，用于以表的形式显示数据。

本文档处于更新中。大部分章节我将仅仅描述如何使用eXtremeTable。当然， 为了使程序高效并具有更高的灵活性，源代码被再三重构。随后, 我认为阐述一下如 何做设计决定是值得的。我希望大家能知道使用extremeTable是多么容易，并且所 有的东西都是可配置。如果你有任何的要求和建议, 请及时通过论坛或者 extremecomponents@gmail.com和我联系。

我想感谢Chris Bauer(Hibernate开发小组)和Rod Johnson(Spring开发小组)。 Chris提供并修改了DocBook-XSL使它能生成Hibernate参考指南。Rod对于使用从Hibernate 得到的这个软件来生成Spring文档提供了支持。我现在用它来生成eXtremeComponents的文档。

译者注:如果大家有什么疑问和建议，可以通过xplucky@gmail.com和我联系。

## Chapter 1. 配置

### 1.1. 先决条件

\* Servlet 2.3或更高版本

\* JDK 1.3.1或更高版本

最小的Jars要求：

\* commons-beanutils 1.6

\* commons-collections 3.0

\* commons-lang 2.0

\* commons-logging 1.0.4

\* standard 1.0.2

PDF导出需要的Jars：

\* minimum jars (above)

\* avalon-framework 4.0

\* batik 1.5-fop-0.20-5

\* fop 0.20.5

\* xalan 2.5.1

\* xercesImpl 2.6.1

\* xml-apis 2.0.2

XLS导出需要的Jars：

\* minimum jars (above)

\* poi-2.5.1.jar

### 1.2. 安装

从[sourceforge](http://sourceforge.net/projects/extremecomp/) 下载发行包。

在压缩文件里你能找到开始使用需要的所有东西：

\* extremecomponents.jar

\* extremecomponents.tld

\* extremecomponents.css

\* 默认的一组图片

\* 源代码

\* test.jsp (用于确认安装是否正确)

将extremecomponents.jar文件拷贝到你的工程的/WEB-INF/lib目录下。

处理TLD文件有两种方式。 你可以把extremecomponents.tld文件放到WEB-INF目录下的任何地方。 不过，为了便于管理，我喜欢把我的TLD文件都放到/WEB-INF/tld目录下。你需要根据你的extremecomponents.tld 文件的位置来修改/WEB-INF/web.xml文件的标签映射。

<taglib>

<taglib-uri>/tld/extremecomponents</taglib-uri>

<taglib-location>/WEB-INF/tld/extremecomponents.tld</taglib-location>

</taglib>

随后，你需要向下面一样在你的JSP里把eXtremeTable包含进来：

<%@ taglib uri="/tld/extremecomponents" prefix="ec" %>

如果你的servlet容器支持JSP 1.2 (或更高版本)，它将能够自动发现TLD文件，那么你什么也不需要做。 当extremecomponents.jar被容器加载的时候，在它的META-INF目录下的extremecomponents.tld文件将被找到。 这时，你需要向下面一样在你的JSP里把eXtremeTable包含进来：

<%@ taglib uri="http://www.extremecomponents.org" prefix="ec" %>

为了使用eXtremeTable样式，从styles目录拷贝extremecomponents.css到你存放.css脚本的地方。 当然在JSP页面里，你需要提供一个到CSS的链接。就像我将我的样式表放在/styles目录下。

<%@ taglib uri="/tld/c" prefix="c" %>

<link rel="stylesheet" type="text/css" href="<c:url value="/styles/extremecomponents.css"/>">

### 1.3. 导出过滤器(可选)

为了使导出功能有效，你需要设置导出过滤器。这是一个仅用于导出功能的可选配置。

如下所示在/WEB-INF/web.xml里配置过滤器：

<filter>

<filter-name>eXtremeExport</filter-name>

<filter-class>org.extremecomponents.table.filter.ExportFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>eXtremeExport</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

过滤器还有一个可选的初始化参数，用于决定什么时候生成报头（headers）。我发现大多数的servlet容器 倾向于在调用过滤器的doFilter()方法后才设置响应报头（response headers）。然而，一些servlet容器只有在 调用过滤器的doFilter()方法前设置响应报头，过滤器才能正常工作。默认的方法是调用过滤器的doFilter()方法后 设置响应报头，你可以通过使用responseHeadersSetBeforeDoFilter这个初始化参数调整它。

<filter>

<filter-name>eXtremeExport</filter-name>

<filter-class>org.extremecomponents.table.filter.ExportFilter</filter-class>

<init-param>

<param-name>responseHeadersSetBeforeDoFilter</param-name>

<param-value>true</param-value>

</init-param>

</filter>

如果你使用了Sitemesh，你将需要包含SitemeshPageFilter。SitemeshPageFilter扩展了正常的 sitemesh的PageFilter，它使得正在进行导出的JSP页面不被修饰。

如下所示在/WEB-INF/web.xml里配置过滤器：

<filter>

<filter-name>Sitemesh</filter-name>

<filter-class>org.extremecomponents.table.filter.SitemeshPageFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>Sitemesh</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

### 1.4. 安装测试

使用发行包的test.jsp来测试安装。将test.jsp拷贝到web应用的最顶层， 默认的图片文件在 /images/table/子目录下。为了测试，创建相应的目录并将拷贝所有需要的图片。所有工作都完成后，你可以在浏览 器里运行test.jsp了。

提示: 我不提倡在JSP里使用脚本（scriplets），但为了不用使用框架而能进行快速测试，在test.jsp 使用脚本是唯一的办法。

## Chapter 2. 概述

### 2.1. 引言

eXtremeTable在给定的servlet范围（scope）外取得Beans或Maps的集合用于JSP页面显示。 servelet范围的搜索顺序是：page, request, session和application。通过设定TableTage的items 属性，eXtremeTable知道哪些需要在servlet范围外被保持。

集合里的Beans是pojo，如果使用maps那么它就是键值对。你可以认为每一个bean就是表中的一行数据。 在接下来的文档中，我将使用Beans集合来代替这两种集合。

实际表使用的最小语法如下所示：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

imagePath="${pageContext.request.contextPath}/images/\*.gif">

<ec:row>

<ec:column property="firstName"/>

<ec:column property="lastName"/>

<ec:column property="termDate"/>

</ec:row>

</ec:table>

这就是eXtremeTable的一个典型定义。通过这些定义，你将得到格式化的列和页头，工具栏的所有 特性也能很好工作。

## Chapter 3. TableTag

### 3.1. 引言

TableTag用来设定什么被显示并且如何进行显示。默认的eXtremeTable在servlet范围(按照page，request， session，applicaton的顺序)寻找具有名称和items属性设置相同的Beans集合（如前章所述它指Beans和Maps两种集合）。 表将遍历所有列，它使用var属性将当前行对应的bean从集合传到page范围，因此你可以从page范围中重新得到这些数据 进行操作。tableId用来唯一标识表，如果在JSP页面里包含两个或两个以上的表时需要设置它。

President bean定义如下:

public class President implements Serializable {

private String firstName;

private String lastName;

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

}

Beans集合需要被组装并传到servlet范围中。我喜欢使用Spring框架，因此示例将使用Spring框架的 控制器（Controller）。如果你正在使用Struts，它和Action的功能类似。如果你使用别的东西，比如直接使用 servlets，你只需要明白我所做的只是组装Beans集合并传到request范围中。

public class Presidents extends AbstractController {

protected ModelAndView handleRequestInternal(HttpServletRequest request,

HttpServletResponse response)

throws Exception {

List presidents = new ArrayList();

President president = new President();

president.setFirstName("George");

president.setLastName("Washington");

presidents.add(president);

president = new President();

president.setFirstName("John");

president.setLastName("Adams");

presidents.add(president);

request.setAttribute("presidents", presidents);

return new ModelAndView("/demo/presidents.jsp");

}

现在你可以构造表了：

<%@ taglib uri="/tld/extremecomponents" prefix="ec" %>

<ec:table

items="presidents"

var="pres"

imagePath="${pageContext.request.contextPath}/images/\*.gif"

action="${pageContext.request.contextPath}/presidents.run"

>

<ec:column property="firstName"/>

<ec:column property="lastName"/>

${pres.lastName}

</ec:column>

</ec:table>

从本示例中你应该知道我们将名为presidents的Beans集合以presidents为名称放到request中。 为了使表知道如何找到这个Beans集合，我们设置TableTag的items属性为presidents。同时我们定义 了两列：firstName和lastName。firstName列是最普通的用法：我们仅仅想让这列从当前bean中得到相应 firstName的值；lastName列示另外一种用法：明确取得值。

从一列中明确取得值非常有用，但是你需要理解表是如何构造行的。为了构造行，表需要对所有行进行 rowsDisplayed属性设定次数的迭代。每次迭代都从Beans里取得下一个bean并使用var属性设定的名称传入page 范围。也可以说每次迭代你都访问的是集合中当前行对应的bean。

### 3.2. 显示图片

为了显示图片需要设置imagePath属性：

<ec:table

items="presidents"

var="pres"

imagePath="${pageContext.request.contextPath}/images/\*.gif"

>

</ec:table>

eXtremeTable将找到一个目录下的所有图片并使用特殊的语法来定义他们是那类图片。 本示例中所有的图片都直接保存在web上下文的images目录下。\*.gif使eXtremeTable知道所 有的图片都是GIF格式的。在我们讨论preferences后，你将发现你可以你可以通过在 extremecomponents.properties文件中设定这个属性，而不用再整个应用的每个eXtremeTable 中包含它。

### 3.3. 过滤、排序和动作（Action）

eXtremeTable内嵌了过滤和排序功能，你只需要决定是否需要使用他们。你要使用的属性是 filterable和sortable，他们都是布尔值并且默认值是true。默认的所有特性都有效，你可以按照 需要来关掉一些特性。比如，如果你不想使用排序或过滤你可以把他们的属性设为false。

<ec:table

items="presidents"

var="pres"

imagePath="${pageContext.request.contextPath}/images/\*.gif"

action="${pageContext.request.contextPath}/presidents.run"

filterable="false"

sortable="false"

>

</ec:table>

如果你仍不确信，你可以来验证他们。首先，设置filterable和sortable为true，你将看到 eXtremeTable允许你输入关键词来过滤结果集，它也允许你通过在页头（header）上滚动鼠标来排序。 然后，设置filterable和sortable为false，你将发现所有这些特性都不允许使用。

本示例需要指出的是使用action属性，action被用来告诉eXtremeTable当过滤或排序时如何回 到当前的页面。本例中我通过Spring框架的controller（在这里是presidents.run）来得到Beans集合。 你不需要担心传参问题，eXtremeTable将保存所有的参数并将它们和过滤器、排序、分页一起传递给 Beans集合。更详细的信息请参考ParameterTag。

### 3.4. 设定每页显示行数

默认地eXtremeTable一页将显示15行。你可以通过设定rowsDisplayed属性为你想显示行数的数 值来改变它。rowsDisplayed也可以在extremecomponents.properties文件中设定。(参考Preferences)。

提示：如果你想在一页中显示所有行，只需要设置showPagination为false。

### 3.5. 样式

TableTag关联了很多样式属性：

<ec:table

cellspacing="0"

cellpadding="0"

border="0"

width="80%"

style=""

styleClass=""

/>

所有这些都是可选的。

### 3.6. 保存表的状态

表新增了两个属性：state和stateAttr。state属性参照State接口并能插接如何保存表的状态的不同实现。

State接口如下：

public interface State {

public void saveParameters(TableModel model, Map parameters);

public Map getParameters(TableModel model);

}

state属性使用预设的四种状态（default、notifyToDefault、persist和notifyToPersist）之一， 你也可以插接自己的实现。default状态不维持任何状态；persist状态没有任何参数传入，将一直维持表的状态； notifyToDefault状态将一直维持表的状态直到你传入参数告诉它回到default状态；notifyToPersist状态 将一直维持当前状态直到你传入参数告诉它维持persisted状态。

stateAttr为指定参数提供了一条途径，你也可以使用属性文件在全局范围内指定它。 为了向后兼容，默认参数一直为useSessionFilterSort。

如果你想state按照不同方式工作你只要实现State接口，然后使用TableTag的state属性来指定实现类的 全路径。

作为一条首要规则当使用state属性时，需要指定tableId。这是因为state使用tableId为名保存在session里。 如果tableId不唯一，eXtremeTable将覆盖另一个同名的内容。tableId默认值为ec。

### 3.7. 其他属性

为了保持一致性，所有的显示特性都命名为showXXXX。他们包括showPagination、showStatusBar、 showTooltips、和showExports。

title属性将在表的上方显示标题，标题的位置根据使用的视图不同而不同。当前默认视图中标题位于表的上方 工具条的左边。更详细的信息请参考View。

你会发现还有一些属性没有被探讨，因为他们将在其他章探讨。autoIncludeParameters在ParameterTag里被探讨； retrieveRowsCallback，sortRowsCallback和filterRowsCallback在Callbacks里被探讨。

### 3.8. 扩展属性

大多数标签包含一系列的固定属性，这样那些已经实现的功能能够被使用。然而，eXtremeTable具有一种更具弹性的架构， 你可以添加自己的标签属性实现更多的定制工作。此外，eXtremeTable提供了非常清晰的钩子（hooks）允许你得到那些定制的 标签属性来做一些你需要的工作。

通过addExtendedAttributes()方法将扩展属性包含到eXtremeTable里：

public void addExtendedAttributes(Table table);

<pre>

如果方法被覆盖TableTag将调用它。你需要做的就是扩展TableTag，覆盖addExtendedAttributes()方法，然后添加自己 的属性到表对象中。一个定制的TreeTag示例如下：

<pre>

public class TreeTag extends TableTag {

private String parentAttribute;

private String identifier;

public void setParentAttribute(String parentAttribute) {

this.parentAttribute = parentAttribute;

}

public void setIdentifier(String identifier) {

this.identifier = identifier;

}

public void addExtendedAttributes(Table table) {

table.addAttribute(TableConstants.PARENT\_ATTRIBUTE, TagUtils.evaluateExpressionAsString("parentAttribute",

parentAttribute, this, pageContext));

table.addAttribute(TableConstants.IDENTIFIER, TagUtils.evaluateExpressionAsString("identifier",

identifier, this, pageContext));

table.setFilterRowsCallback("org.extremecomponents.tree.ProcessTreeRowsCallback");

table.setSortRowsCallback("org.extremecomponents.tree.ProcessTreeRowsCallback");

}

}

现在你添加了属性值到table对象。

另外，你也可以定制自己的标签和自己的TLD文件。你不需要修改extremecomponents.tld文件。 你能象使用eXtremeTable里的标签一样使用自己的标签，除了使用你自己标签的参照。假如你的标签参照为mycompany 并且标签为customTable，你可以像下面一样使用他们：

<mycompany:customTable

items="presidents"

action="${pageContext.request.contextPath}/public/demo/presidents.jsp"

title="Presidents"

>

<ec:row>

<ec:column property="nickName"/>

</ec:row>

</mycompany:customTable>

## Chapter 4. RowTag

### 4.1. 引言

RowTag用来定义表中的行。

一个简单RowTag示例如下：

<ec:table

items="presidents"

var="pres"

action="${pageContext.request.contextPath}/presidents.run"

>

<ec:row>

<ec:column property="name"/>

<ec:column property="term"/>

</ec:row>

</ec:table>

### 4.2. 高亮显示行

使用highlightRow属性可以设置行的高亮显示，它的值为true或false，默认值为false。

<ec:table

items="presidents"

var="pres"

action="${pageContext.request.contextPath}/presidents.run"

>

<ec:row highlightRow="true">

<ec:column property="name"/>

<ec:column property="term"/>

</ec:row>

</ec:table>

设置highlightRow属性后，它将插入设置行css类的javascript脚本，默认的css 类为highlight。你可以使用highlightClass来使用定制的css类。

### 4.3. 样式

RowTag关联了很多样式属性：

<ec:row

style=""

styleClass=""

highlightClass=""

/>

所有这些都是可选的。highlightClass设置行高亮显示时的css类； style属性定义行内联的样式；styleClass允许你定义一个行显示的css类。

### 4.4. Javascript属性

为了便于你对于行数据进行动态交互处理，提供了onclick、onmouseover和 onmouseout属性。

<ec:row

onclick=""

onmouseover=""

onmouseout=""

/>

### 4.5. 扩展属性

大多数标签包含一系列的固定属性，这样那些已经实现的功能能够被使用。然而，eXtremeTable具有一种更具弹性的架构， 你可以添加自己的标签属性实现更多的定制工作。此外，eXtremeTable提供了非常清晰的钩子（hooks）允许你得到那些定制的 标签属性来做一些你需要的工作。

通过addExtendedAttributes()方法将扩展属性包含到eXtremeTable里：

public void addExtendedAttributes(Row row);

## Chapter 5. ColumnTag

### 5.1. 引言

ColumnTag用来定义表中的列。

示例President Bean：

public class President implements Serializable {

private String firstName;

private String lastName;

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

}

下例生成firstName和lastName列：

<ec:table

items="presidents"

var="pres"

action="${pageContext.request.contextPath}/presidents.run"

/>

<ec:row>

<ec:column property="firstName"/>

<ec:column property="lastName">

${pres.lastName}

</ec:column>

</ec:row>

</ec:table>

通过对TableTag的讨论，已经知道列可以通过动态或精确的方式得到他们的值。 firstName列动态地取得相应的值，列找到当前的bean并调用相应的getFirstName()取得值。 lastName列明确地从当前bean取得值，它要求你自己取得相应的值。如下例：

<ec:table

items="presidents"

var="pres"

action="${pageContext.request.contextPath}/presidents.run"

>

<ec:row>

<ec:column property="lastName">

${pageScope.pres.lastName}

</ec:column>

</ec:row>

</ec:table>

从page范围中取得名为pres的bean并得到它对应的lastName属性值。如果你正使用 Beans集合请确认具有对应的getter方法；如果使用Maps集合则不需要任何别的动作， eXtremeTable能够通过属性名从Map中得到对应的值。

提供这种可选取值方法的主要原因是使你能够对其他类型的html标签提供动作支持，例如显示 一幅图片或者通过定义href使该列成为到其它页的一个链接。

<ec:table

items="presidents"

var="pres"

action="${pageContext.request.contextPath}/presidents.run"

>

<ec:row>

<ec:column property="lastName">

<a href="http://goto.president.detail">${pageScope.pres.lastName}</a>

</ec:column>

</ec:row>

</ec:table>

切记bean中所有的属性都是可访问的，因此你甚至可以通过firstName属性 来显示下一页。请注意firstName属性是如何作为URL字符串传输的。

<ec:table

items="presidents"

var="pres"

action="${pageContext.request.contextPath}/presidents.run"

/>

<ec:row>

<ec:column property="lastName">

<a href="http://goto.president.detail?firstName=${pageScope.pres.firstName}">

${pageScope.presidents.lastName}

</a>

</ec:column>

</ec:row>

</ec:table>

我将不再在任何示例中强调pageScope。JSP标签总是最先在pageScope中寻找任何对像, 因此我们总是能安全地返回正确的bean。

### 5.2. Cell

每一列总是被实现Cell接口的对象修饰，你可以认为Cell是一个为了html显示或导出而返回格式化值的对象。 发行包包含的Cell有DisplayCell、DateCell、 NumberCell和RowCountCell。 DisplayCell是仅仅显示列值的默认cell；DateCell使用parse属性（可选）和format属性来格式化对应的属性值； NumberCell使用format属性来格式化对应的属性值；RowCountCell显示当前行。

提示：为了避免混乱并提高灵活性Cell接口已经被修改。而且对于区别 如何处理html和导出显示值也不是很清晰。以前列值作为html显示，列的propertyValue作为导出使用。 另外列值和propertyValue已经重写，他们以前在view中是不能被访问的。

cell现在是singleton并且不再线程安全，改变的原因是为了Cell接口能更简单地被使用。 init()和destroy()方法作为singleton更灵活但是处于一种混乱的状态。

Cell接口如下：

public interface Cell {

/\*\*

\* The display that will be used for the exports.

\*/

public String getExportDisplay(TableModel model, Column column);

/\*\*

\* The html that will be displayed in the table.

\*/

public String getHtmlDisplay(TableModel model, Column column);

}

现在得到导出和html显示存在明显的区别。更重要的，需要返回字符串。列值和属性值不再 需要设置。

DisplayCell是扩展AbstractCell的最简单的Cell。AbstractCell定义 的虚拟方法getCellValue用来返回cell的值。虽然AbstractCell在一些情况下是有用的， 但更多情况下只需要直接实现Cell接口。

DisplayCell：

public class DisplayCell extends AbstractCell {

public String getExportDisplay(TableModel model, Column column) {

return column.getPropertyValueAsString();

}

protected String getCellValue(TableModel model, Column column) {

return column.getValueAsString();

}

}

AbstractCell：

public abstract class AbstractCell implements Cell {

public String getExportDisplay(TableModel model, Column column) {

return getCellValue(model, column);

}

public String getHtmlDisplay(TableModel model, Column column) {

HtmlBuilder html = new HtmlBuilder();

CellBuilder.tdStart(html, column);

CellBuilder.tdBody(html, getCellValue(model, column));

CellBuilder.tdEnd(html);

return html.toString();

}

/\*\*

\* A convenience method to get the display value.

\*/

protected abstract String getCellValue(TableModel model, Column column);

}

现在你应该知道Cell是多么简单。只需通过实现Cell接口或扩展AbstractCell来定制你自己的Cell， 并设置列标签的Cell属性为类的全路径。例如: 如果你定制了一个名为MyCell的Cell，那么你可以像下面一样使用它：

<ec:column property="firstName" cell="com.mycompany.cell.MyCell"/>

如果你改变列的数据，那么过滤或排序可能没有意义。切记我的意思是如果你人为地改变数据， 而不是使用样式对它进行包装或作为<a href>包含。 如果你的定制cell显示数据的树状视图，或者是一幅图片， 那么过滤和排序等一切逻辑操作都是没有意义的。

### 5.3. Filter Cell

列的filterCell属性控制过滤器如何显示，它和cell属性非常相像并且也是实现Cell接口。 已经定义了两个过滤器cells：默认的和droplist。默认的是一个输入框元素，除非你确信你需要使这列可以进行过滤， 否则你不需要做任何事。

你可以像下面一样使用droplist过滤器Cell：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

>

<ec:row>

<ec:column property="status" filterCell="droplist"/>

</ec:row>

</ec:table>

filterCell也允许你定义定制的过滤器，所有你必须做的就是实现Cell接口或者扩展AbstractCell， 并设置列标签的Cell属性为类的全路径。例如，如果你定制了一个名为MyCell的Cell，那么你可以像下面一样使用它：

<ec:column property="firstName" filterCell="com.mycompany.cell.MyFilterCell"/>

参阅Cell节了解如何创建你自己定制Cells的更多信息。

### 5.4. Header Cell

headerCell属性控制headers如何显示，它和cell属性非常相像并且也是实现Cell接口。 默认header cell作为文本显示，包含排序逻辑。

headerCell也允许你定义定制的过滤器，所有你必须做的就是实现Cell接口或者扩展AbstractCell， 并设置列标签的Cell属性为类的全路径。例如，如果你定制了一个名为MyCell的Cell，那么你可以像下面一样使用它：

<ec:column property="firstName" headerCell="com.mycompany.cell.MyHeaderCell"/>

参阅Cell节了解如何创建你自己定制Cells的更多信息。

### 5.5. 样式

ColumnTag关联了很多样式属性：

<ec:column

width=""

style=""

styleClass=""

headerStyle=""

headerClass=""

filterStyle=""

filterClass=""

/>

所有这些都是可选的。style属性定义列内联的样式；styleClass允许你定义一个列显示的css类； headerClass属性允许你改变header列的css类；filterClass属性允许你改变filter列的css类。

### 5.6. 解析和格式化

解析和格式化属性被用在日期和货币的显示上。

和date交互的工作依赖于你的bean属性是否是一个字符串或者是一个Date对象。 如果是一个字符串那么你就需要确定parse属性，parse属性是按照模板定义来解析一个字符串为 一个日期对象。如果bean中的属性是日期型对象则不需要添加parse属性。不论如何你都需要设置format属性。 format属性按你提供的模板对值进行格式化。

本示例中使用MM/dd/yyyy模板格式化日期型值。因为bean中的born属性值为字符串，所以我们需要 使用parse属性来将它转换成日期型数值。

<ec:column property="born" cell="date" parse="yyyy-MM-dd" format="MM/dd/yyyy"/>

对于货币只需要设置format属性：

<ec:column property="payroll" cell="currency" format="###,###,##0.00"/>

很多时候在extremeTable中，你使用同样的模版来解析和格式化日期和货币值。 所以便利的方法是在你自己的extremecomponents.properties文件中定义解析和格式化属性。 参阅Preferences章了解更多信息。

### 5.7. 过滤和排序

你可能记得在TableTag中看见过filterable和sortable属性，ColumnTag中也有相同的属性。 列的filterable和sortable属性将覆盖表的filterable和sortable属性设置。当你需要除了对表中的一、两列之外的 所有列进行过滤和排序时，十分便利。

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

>

<ec:row>

<ec:column property="firstName" filterable="false"/>

<ec:column property="lastName" sortable="false"/>

</ec:row>

</ec:table>

### 5.8. Calc

列新增了两个属性：calc和calcTitle：

<ec:column property="data" calc="total" calcTitle="Total:" />

calc属性实现具有唯一方法的Calc接口：

public interface Calc {

public Number getCalcResult(TableModel model, Column column);

}

它传入model和column，并返回一个Number型的值。默认的实现为总计和平均值。

为了使用定制的Calc，只需要使用ColumnTag的calc属性来指定实现Calc接口的实现类的 全路径。

Calc为singleton并且不是线程安全的，因此不要定义任何类变量。

### 5.9. 允许和不允许视图

viewsAllowed属性制定类允许使用的视图。视图包括：html、pdf、xls、csv，以及任何定制的视图。 如果你指定一个或几个视图，那么列仅能使用这些指定的视图。例如：你指定viewsAllowed="pdf"，这意味着 这列只允许PDF导出，而不能进行其他格式的导出或html视图。

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

>

<ec:row>

<ec:column property="firstName"/>

<ec:column property="lastName" viewsAllowed="pdf"/>

</ec:row>

</ec:table>

viewsDenied属性制定类不允许使用的视图。视图包括：html、pdf、xls、csv，以及任何定制的视图。 如果你指定一个或几个视图，那么列仅这些指定的视图不能被使用。例如：你指定viewsDenied="html"，这意味着 这列不允许使用html试图，但能进行任何形式的导出。

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

>

<ec:row>

<ec:column property="firstName"/>

<ec:column property="lastName" viewsDenied="html"/>

</ec:row>

</ec:table>

### 5.10. 其他属性

title属性用来为header设定一个描述性的名称。如果你不定义title那么列将使用属性名。 如果你不想显示任何title，你只需要设置title属性值为一个空白（whitespace）。

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

title="Presidents"

>

<ec:row>

<ec:column property="firstName"/> //title shows as First Name

<ec:column property="firstName" title="First Name"/> //title shows as First Name

<ec:column property="firstName" title=" "/> //no title shows up

</ec:row>

</ec:table>

### 5.11. 扩展属性

大多数标签包含一系列的固定属性，这样那些已经实现的功能能够被使用。然而，eXtremeTable具有一种更具弹性的架构， 你可以添加自己的标签属性实现更多的定制工作。此外，eXtremeTable提供了非常清晰的钩子（hooks）允许你得到那些定制的 标签属性来做一些你需要的工作。

通过addExtendedAttributes()方法将扩展属性包含到eXtremeTable里：

public void addExtendedAttributes(Column column);

如果方法被覆盖ColumnTag将调用它。你需要做的就是扩展ColumnTag，覆盖addExtendedAttributes()方法，然后添加自己 的属性到列对象中。一个定制的CustomTag示例如下：

public class MyCustomTag extends ColumnTag {

private String customAttributeOne;

public String getCustomAttributeOne() {

return customAttributeOne;

}

public void setCustomAttributeOne(String customAttributeOne) {

this.customAttributeOne = customAttributeOne;

}

public void addExtendedAttributes(Column column) {

column.addAttribute("customAttributeOne", customAttributeOne);

}

}

现在你添加了属性值到Column对象，现在你可以像下例一样来定制cell：

public class MyCustomCell implements Cell {

public String getHtmlDisplay(TableModel model, Column column) {

Object customAttributeOne = column.getAttribute("customAttributeOne")

String customAttributeOne = column.getAttributeAsString("customAttributeOne")

}

}

另外，你也可以定制自己的标签和自己的TLD文件。你不需要修改extremecomponents.tld文件。 你能象使用eXtremeTable里的标签一样使用自己的标签，除了使用你自己标签的参照。假如你的标签参照为mycompany 并且标签为customColumn，你可以像下面一样使用他们：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/public/demo/presidents.jsp"

title="Presidents"

>

<ec:row>

<mycompany:customColumn

property="hello"

cell="com.mycompany.cell.MyCustomCell"

customAttributeOne="Hello World"/>

</ec:row>

</ec:table>

## Chapter 6. ParameterTag

### 6.1. 引言

eXtremeTable能够指定是否处理所有的参数。默认为处理所有的参数，这意味着当你进行 过滤、排序、分页时，所有的参数都被eXtremeTable保存并传到JSP中。通常这是你需要的功能， 然而，有时候你需要只允许一些特定的参数保存到eXtremeTable中。我喜欢把它看作锁定extremeTable， 可以通过设置表的autoIncludeParameters属性值为false来到达目的：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

autoIncludeParameters=”false”

>

</ec:table>

甚至当表被锁定时，你仍然可以通过简单地使用parameter标签来包含一些你想使用的参数。下例 包含了foo参数：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

autoIncludeParameters=”false”

>

<ec:parameter name=”foo” value=”${param.foo}”/>

</ec:table>

现在所有的eXtremeTable参数都被锁定。但过滤、排序和翻页时，foo参数仍将被传递。

## Chapter 7. ExportTag

### 7.1. 引言

eXtremeTable具有导出不同格式文件的功能，导出的数据为过滤和排序后的所有结果集， 分页不会影响返回的结果集。换句话说，如果表数据分多页显示，那么所有页的数据都将被导出。 导出的格式为Microsoft Excel (OpenOffice Calc)、PDF和CSV。

使用ExportXlsTag导出Microsoft Excel (OpenOffice Calc)：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

/>

<ec:exportXls

fileName="presidents.xls"

tooltip="Export Excel"/>

</ec:table>

使用ExportPdfTag导出PDF。所有要做的就是指定fileName属性和一些样式属性：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

/>

<ec:exportPdf

fileName="presidents.pdf"

tooltip="Export PDF"

headerColor="blue"

headerBackgroundColor="red"

headerTitle="Presidents"/>

</ec:table>

使用ExportCsvTag导出CSV。当使用CSV导出是默认的分隔符为‘,’(comma)。你可以使用 delimiter属性来指定为其他的符号。下面为指定‘|’（pipe）为CSV分隔符的示例：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

/>

<ec:exportCsv

fileName="presidents.txt"

tooltip="Export CSV"

delimiter="|"/>

</ec:table>

你可以通过指定view属性来导出其他文件格式。eXtremeTable视图实现View接口并是 可插接的。参阅View章了解更多信息。

### 7.2. 扩展属性

大多数标签包含一系列的固定属性，这样那些已经实现的功能能够被使用。然而，eXtremeTable具有一种更具弹性的架构， 你可以添加自己的标签属性实现更多的定制工作。此外，eXtremeTable提供了非常清晰的钩子（hooks）允许你得到那些定制的 标签属性来做一些你需要的工作。

通过addExtendedAttributes()方法将扩展属性包含到eXtremeTable里：

public void addExtendedAttributes(Export export);

如果方法被覆盖ExportTag将调用它。你需要做的就是扩展ExportTag，覆盖addExtendedAttributes()方法，然后添加自己 的属性到导出对象中。

一个定制的ExportCsvTag示例如下:

public class ExportCsvTag extends ExportTag {

private String delimiter;

public String getDelimiter() {

return delimiter;

}

public void setDelimiter(String delimiter) {

this.delimiter = delimiter;

}

public void addExtendedAttributes(Export export) {

String view = export.getView();

if (StringUtils.isBlank(view)) {

export.setView(TableConstants.VIEW\_CSV);

export.setImageName(TableConstants.VIEW\_CSV);

}

export.addAttribute(CsvView.DELIMITER, getDelimiter());

}

}

现在你添加了属性值到Export对象，下面是CsvView实现的一部分：

public class CsvView implements View {

public void body(TableModel model, Column column) {

Export export = model.getExportHandler().getCurrentExport();

}

}

另外，你也可以定制自己的标签和自己的TLD文件。你不需要修改extremecomponents.tld文件。 你能象使用eXtremeTable里的标签一样使用自己的标签，除了使用你自己标签的参照。假如你的标签参照为mycompany 并且标签为customExport，你可以像下面一样使用他们：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/public/demo/presidents.jsp"

title="Presidents"

>

<mycompany:customExport fileName="presidents.txt" delimiter="|"/>

</ec:table>

## Chapter 8. Callbacks

### 8.1. 引言

Callbacks被用于重新得到（retrieve）、过滤和排序行数据。eXtremeTable为每个callback提供了一个定制实现。 首先，载入所有的元数据，元数据为所有eXtremeTable标签的所有属性；接着在eXtremeTable的model中调用 execute方法。eXtremeTable使用execute方法决定如何通过调用每个callback的接口来重新得到、过滤和排序行数据。 这三个callback的接口是：RetrieveRowsCallback、FilterRowsCallback和SortRowsCallback。

callbacks为singleton并且不是线程安全的，因此不要定义任何类变量。呵呵

### 8.2. RetrieveRowsCallback

RetrieveRowsCallback的默认实现在servlet范围内寻找具有名称和TableTag 的items属性设置相同的Beans集合。为了使用定制的callback，只要实现RetrieveRowsCallback接口， 然后使用retrieveRowsCallback属性来指定实现类的全路径：

<ec:table

var="pres"

action="${pageContext.request.contextPath}/presidents.run"

retrieveRowsCallback="com.mycompany.callback.MyCustomCallback"

/>

RetrieveRowsCallback接口如下所示：

public interface RetrieveRowsCallback {

public Collection retrieveRows(TableModel model) throws Exception;

}

只有一个方法需要实现，传入TableModel并返回一个集合，集合为Beans或Maps集合。 通过得到TableModel，就拥有了TableTag的所有元数据并能访问Context。能够访问Context非常重要， 这意味着你访问web容器的任何东西。

### 8.3. FilterRowsCallback

FilterRowsCallback的默认实现得到Beans集合，通过实现jakarta Predicate接口进行过滤， 过滤值从eXtremeTable的filter输入框中取得。为了使用定制的callback，只要实现FilterRowsCallback接口， 然后使用filterRowsCallback属性来指定实现类的全路径：

<ec:table

var="pres"

action="${pageContext.request.contextPath}/presidents.run"

filterRowsCallback="com.mycompany.callback.MyCustomCallback"

/>

FilterRowsCallback接口如下所示：

public interface FilterRowsCallback {

public Collection filterRows(TableModel model, Collection rows) throws Exception;

}

只有一个方法需要实现，传入TableModel并返回一个集合。你只需像eXtremeTable对 每个callback的默认实现一样来定制自己的callback。

### 8.4. SortRowsCallback

SortRowsCallback的默认实现得到Beans集合，使用jakarta BeanComparator进行排序， 排序值当用户点击列头时取得。为了使用定制的callback，只要实现SortRowsCallback接口， 然后使用sortRowsCallback属性来指定实现类的全路径：

<ec:table

var="pres"

action="${pageContext.request.contextPath}/presidents.run"

sortRowsCallback="com.mycompany.callback.MyCustomCallback"

/>

SortRowsCallback接口如下所示：

public interface SortRowsCallback {

public Collection sortRows(TableModel model, Collection rows) throws Exception;

}

只有一个方法需要实现，传入TableModel并返回一个集合。你只需像eXtremeTable对 每个callback的默认实现一样来定制自己的callback。

## Chapter 9. 视图

### 9.1. 引言

eXtremeTable里视图是可插接的，这意味着html很容易改变，或者一类新的导出能够被实现。 所有需要做的就是实现View接口并在TableTag或ExportTag中设置view属性。首先，让我们看一下View接口：

public interface View {

public void beforeBody(TableModel model);

public void body(TableModel model, Column column);

public Object afterBody(TableModel model);

}

实现View接口的类有三次插入内容的机会。beforeBody()方法会被立刻调用； body()方法在每一行的每一列处理的时候调用；afterBody()方法是被eXtremeTable调用的 最后方法，它将返回代表视图的一个对象，通常它是一个字符串。例如：在HTML视图类中为的html 标签（markup），当然它可以是任何东西。最主要的原因是定制导出时,你应该返回一些其他的对象。

### 9.2. 表视图

eXtremeTable的所有标签（markup）在两个地方生成：View或Cell。 组合使用他们， 能为你提供一种可插接的表示内容的解决方案。为了使用定制的View， 只需要使用TableTag的view属性来指定实现View接口的实现类的 全路径：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

view="com.mycompany.view.MyCustomView"

>

...

</ec:table>

### 9.3. 导出视图

与表视图不同，export的所有标签都在View中生成。为了使用定制的View， 只需要使用ExportTag的view属性来指定实现View接口的实现类的 全路径：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

>

<ec:export

fileName="custom.file"

tooltip="Export Custom"

view="com.mycompany.view.MyCustomExportView"/>

...

</ec:table>

## Chapter 10. Preferences

### 10.1. 引言

为了替代硬编码eXtremeTable使用的默认属性值，我在属性文件中配置所有用到的属性。 如果你需要覆盖任何默认的设置，你可以创建自己的extremecomponents.properties文件 并设置你想改变的值。

为了设置属性文件，你应该如下例所示在/WEB-INF/web.xml文件中声明一个context-param，并 指定你的属性文件的路径：

<context-param>

<param-name>extremecomponentsPreferencesLocation</param-name>

<param-value>/org/extremesite/resource/extremecomponents.properties</param-value>

</context-param>

你可以认为属性文件为你提供了一个对所有的eXtremeTables声明全局设置的一个方法。 创建属性文件的最大好处就是避免在标签中复制、粘贴相同的属性。典型的extremecomponents.properties文件如下所示：

table.imagePath=/extremesite/images/\*.gif

table.rowsDisplayed=12

column.parse.date=yyyy-MM-dd

column.format.date=MM/dd/yyyy

column.format.currency=$###,###,##0.00

### 10.2. TableTag

在属性文件定义的TableTag使用最多的两个属性是：imagePath和rowsDisplayed。如果你不在属性文件中声明 这些属性，你需要在每个eXtremeTable中添加他们。典型的表如下所示：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

imagePath="${pageContext.request.contextPath}/images/\*.gif"

rowsDisplayed="12"

title="Presidents"

>

...

</ec:table>

如果在属性文件声明imagePath和rowsDisplayed，则表如下所示：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/presidents.run"

title="Presidents"

>

...

</ec:table>

正如你所见，属性文件避免了重复编码。

### 10.3. ColumnTag

在属性文件定义的ColumnTag使用最多的两个属性是：parse和format。如果你不在属性文件中声明 这些属性，你需要在每个eXtremeTable中添加他们。典型的列使用日期cell如下所示：

<ec:column property="dateOfBirth" cell=”date” parse=”yyyy-MM-dd” format=”MM/dd/yyyy”/>

如果在属性文件声明parse和format，则列如下所示：

<ec:column property="dateOfBirth" cell=”date”/>

当然你仍然可以定义parse和format属性来覆盖全局设置，但是大多数工程对于日期使用一致的parse 和format。需要注意属性文件中parse.date和format.date的声明语法。

下例为使用货币cell的典型列：

<ec:column property="salary" cell=”currency” format=”$###,###,##0.00”/>

如果在属性文件声明format，则列如下所示：

<ec:column property="salary" cell=”currency”/>

另外，你可以声明一个定制的format并在列中通过使用列的basis来使用它，我把这想象为named属性。因此如果你的 extremecomponents.properties文件如下所示：

table.format.myCustomDate=yy-MM-dd

那么列可以如下使用定制的format：

<ec:column property="dateOfBirth" cell="date" format=”myCustomDate”>

### 10.4. Advanced Techniques

使用named属性是我定义其他不同属性默认值时经常使用的方法。你可能对我 使用cell="date"来指定日期cell、使用cell="currency"来指定货币cell或使用view="xls."来指定xls导出感到疑惑。 如果我给你展示extremetable.properties文件的一些片断，这些就将非常清晰了。 extremetable.properties是eXtremeTable声明默认设置的属性文件，你可以通过使用 extremecomponents.properties文件来覆盖它。

column.cell.date=org.extremecomponents.table.cell.DateCell

column.cell.currency=org.extremecomponents.table.cell.NumberCell

column.filterCell.droplist=org.extremecomponents.table.cell.FilterDroplistCell

table.view.xls=org.extremecomponents.table.view.XlsView

当你在列上定义cell="date"时，eXtremeTable寻找到column.cell. 属性并将你定义的cell属性值拼接上。 换句话说cell="date"关联到column.cell.date=org.extremecomponents.table.cell.DateCell这条属性。使用属性文件 真正强大的地方在于你能够在extremecomponents.properties文件中声明一个定制的cell，并在ColumnTag中通过 名称来使用它。

再使用一个实例来阐明这一点，是否记得ColumnTag章Cell节中如何调用一个名为MyCell的定制cell：

<ec:column property="firstName" cell="com.mycompany.cell.MyCell"/>

cell使用的更好方式是在属性文件中声明并通过名称使用它。首先，更新extremecomponents.properties文件：

table.imagePath=/extremesite/images/\*.gif

table.rowsDisplayed=12

table.cellspacing=2

column.parse.date=yyyy-MM-dd

column.format.date=MM/dd/yyyy

column.format.currency=$###,###,##0.00

column.cell.myCell=com.mycompany.cell.MyCell

现在可以通过名称调用MyCell：

<ec:column property="firstName" cell="myCell"/>

正如你所见的这能帮助保持代码清洁，并且这些都在一个地方定义。如果你的定制cell声明 需要改变你只需要修改属性文件。

## Chapter 11. Messages

### 11.1. 资源绑定

为了设置资源绑定，你应该如下例所示在/WEB-INF/web.xml文件中声明一个context-param，并 指定你的资源文件的路径：

<context-param>

<param-name>extremecomponentsMessagesLocation</param-name>

<param-value>org/extremesite/resource/extremecomponentsResourceBundle</param-value>

</context-param>

本示例中资源文件为extremecomponentsResourceBundle，它可以为任何名或者使用已经存在的资源文件。

如果你不指定locale，则它将根据你的servlet request来决定使用哪个资源文件。 在eXtremeTable中可以通过使用TableTag的locale属性来设置它。

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/public/demo/locale.jsp"

title="table.title.president"

locale="de\_DE"

>

...

</ec:table>

在这里eXtremeTable将寻找德文资源文件。

### 11.2. 全局资源

eXtremeTable使用一些全局的keys来与用户交互，包括：状态栏的文本信息，Rows Displayed droplist和不同的tooltips。如果你足够幸运，eXtremeTable已经提供了相应的语言支持 的话，那么你什么也不用担心。否则的话，你需要申明下列keys：

statusbar.resultsFound={0} results found, displaying {1} to {2}

statusbar.noResultsFound=There were no results found.

toolbar.firstPageTooltip=First Page

toolbar.lastPageTooltip=Last Page

toolbar.prevPageTooltip=Previous Page

toolbar.nextPageTooltip=Next Page

toolbar.filterTooltip=Filter

toolbar.clearTooltip=Clear

toolbar.clearText=Clear

toolbar.firstPageText=First

toolbar.lastPageText=Last

toolbar.nextPageText=Next

toolbar.prevPageText=Prev

toolbar.filterText=Filter

column.headercell.sortTooltip=Sort By

column.calc.total=Total

column.calc.average=Average

现在仅支持英语和德语。如果你使用其他语言的话，并能提供相应的翻译的话我将不胜感激。你可以通过 extremecomponents@gmail.com发送给我。

译者注:我已经提供了中文和日文的资源文件。

### 11.3. TableTag

TableTag属性中能够使用locale方式指定的是：imagePath和title。

在eXtremeTable中，imagePath属性有一个特定的key：table.imagePath。你可以在你的资源文件中 设置这个key为特定语言的目录结构。例如：德文图片可能放在de文件夹下，那么你可以在相应的资源文件中 进行如下设置：

table.imagePath=/extremesite/images/table/de/\*.gif

title有一点不同，如果你指定的title属性值包含dot (.)并且你定义了一个资源文件，那么 eXtremeTable将寻找匹配的key。例如，如果你像下例一样在表中指定属性title="table.title.president"：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/public/demo/locale.jsp"

title="table.title.president"

>

...

</ec:table>

那么eXtremeTable将在属性文件中寻找匹配的key：

table.title.president=US Präsidenten

### 11.4. ColumnTag

ColumnTag属性中能够使用locale方式指定的是：format和title。

在eXtremeTable中，format属性有一个特定的key：table.fomat.type。参考属性文件的讨论 来了解更多的细节，他们具有同样的概念。日期和货币的format类型定义可能如下所示：

column.format.date=MM/dd/yyyy

column.format.currency=$###,###,##0.00

title有一点不同，如果你指定的title属性值包含dot (.)并且你定义了一个资源文件，那么 eXtremeTable将寻找匹配的key。例如，如果你像下例一样在列中指定属性title="table.column.nickName"：

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/public/demo/locale.jsp"

title="table.title.president"

>

<ec:row>

<ec:column property="nickName" title="table.column.nickName" />

</ec:row>

</ec:table>

那么eXtremeTable将在属性文件中寻找匹配的key：

table.column.nickName=Spitzname

## Chapter 12. Limit

### 12.1. Limit指南

默认的情况下eXtremeTable取得所有的结果集然后处理Beans集合，这样的好处是 你可以随意进行排序、过滤和分页操作。你只需要组装Beans集合并让eXtremeTable知道如何 引用它。这样的操作对于小到中等数据量的结果集非常有效，当结果集很大时这将非常糟糕。 这是一个判断，但我更喜欢描述如何做出我的技术决定。如果您认为在性能上有问题， 那么最好是使用一个profiler工具记录并查看它。有许多开源和商业的profiler工具可以帮助 你做出最好的判断。因此，假设我们发现了性能上存在问题，需要我们自己来处理分页。

手动处理分页意味着你一次只想取得一页显示需要的结果集。同时，你需要自己处理排序、过滤和分页。 下面的讨论是基于我假设你从数据库中取得集合，当然同样的原理能应用到任何地方。

这是一个重要的部分。为了得到较小的结果集，你可以创建一个普通的查询语句，但是limit你得到的结果集。 在Sybase和SQLServer中你可以使用rowcount命令，在MySql中你可以使用limit命令。 我不知道其他数据库怎么使用，但我确信每个数据库都有相似的功能。 也就是说当用户浏览第一页是得到第一页需要的 结果集，当用户浏览下一页时，再得到下一页需要的结果集。

使用Sybase的开发人员可能会说:rowcount命令总是从第一条开始，那么当我到第二页时我也必须 从第一条数据开始。 是的，你现在得到的是两页的结果集，而不需要得到所有的结果集。当你到第三页时，你只需要得到三页的结果集。。。。。。 其他数据库比如MySQL，允许你精确地得到你想要的那段数据，这样你就可以只得到当前页面显示需要的结果集。

为了知道用户想如何排序和过滤，他们想浏览哪一页，一页需要显示几条结果，eXtremeTable有一个使用LimitFactory 创建的名为Limit的简便接口：

### 12.2. 创建Limit

首先你需要通过LimitFactory得到一个Limit实例：

Context context = new HttpServletRequestContext(request);

LimitFactory limitFactory = new TableLimitFactory(context, tableId);

Limit limit = new TableLimit(limitFactory);

Limit对象定义了limit结果集的所有方法。

TableLimitFactory具有另外一个构造函数，如果没有指定tableId的话默认的tableId将为ec。

Context context = new HttpServletRequestContext(request);

LimitFactory limitFactory = new TableLimitFactory(context);

Limit limit = new TableLimit(limitFactory);

### 12.3. Filter和Sort属性

当你对Limit实例化时，实例化对象包含两个对象：FilterSet和Sort。

private FilterSet filterSet;

private Sort sort;

FilterSet包含一个过滤动作（Action）和一个过滤器对象数组。 动作为TableConstants.FILTER\_ACTION或TableConstants.CLEAR\_ACTION。 一个过滤器包含一个property和这个过滤器的值。

private final String action;

private final Filter[] filters;

Sort对象包含property和sortOrder。sortOrder为 TableConstants.SORT\_ASC或TableConstants.SORT\_DESC：

private Sort sort;

### 12.4. 设置页和行属性

设置行属性：

limit.setRowAttributes(totalRows, DEFAULT\_ROWS\_DISPLAYED);

下面是设置行属性可能用到的信息：

private int rowStart;

private int rowEnd;

private int currentRowsDisplayed;

private int page;

private int totalRows;

每个变量都有一个getter方法,我将不深入讲解属性的细节。

### 12.5. Setup

在你完成所有的定制工作：排序、过滤.....定制的Controller（Spring）或者Action（Struts）或者其他类似的框架后， 另外你需要创建一个callback，eXtremeTable已经提供了一个名为LimitCallback的实现。为了使用你只需要设置表 属性：retrieveRowsCallback、filterRowsCallback和sortRowsCallback：

<ec:table

items="presidents"

retrieveRowsCallback="limit"

filterRowsCallback="limit"

sortRowsCallback="limit"

action="${pageContext.request.contextPath}/limit.run"

title="Presidents"

>

<ec:row>

<ec:column property="fullName" title="Name"/>

<ec:column property="nickName" />

<ec:column property="term" />

<ec:column property="born" cell="date"/>

<ec:column property="died" cell="date"/>

<ec:column property="career" />

</ec:row>

</ec:table>

使用callback需要做的唯一事情是传输集合到request，同时传输totalRows属性。 totalRows表示总行数，使用PaginationCallback.TOTAL\_ROWS静态变量将易于维护。 如果JSP页面使用了两个（以上）eXtremeTable的话你可以利用tableId分别传输totalRows。 例如如果tableId名为pres，你可以如下处理：

request.setAttribute("pres", presidents);

request.setAttribute("pres\_totalRows", new Integer(""+totalRows));

译者注:关于limit使用的更详细信息，请参考《Limit指南》

## Chapter 13. AutoGenerateColumns

### 13.1. 引言

大多数情况下你按照你需要的列来设计数据库表。但是，有时候需要运行时动态生成一些列。 为了实现这点，eXtremeTable需要使用ColumnsTag并设置autoGenerateColumns属性。

AutoGenerateColumns为singleton并且不是线程安全的，因此不要定义任何类变量。

### 13.2. ColumnsTag

ColumnsTag只有autoGenerateColumns这一个属性。所有你必须做的就是实现AutoGenerateColumns接口， 并设置autoGenerateColumns属性为类的全路径。

<ec:table

items="presidents"

action="${pageContext.request.contextPath}/autoGenerateColumns.run"

title="Presidents"

>

<ec:columns autoGenerateColumns="org.extremesite.controller.AutoGenerateColumnsImpl"/>

</ec:table>

AutoGenerateColumns接口只有一个方法：

public void addColumns(TableModel model);

你需要做的就是添加列（columns）到model里。最简单的示例如下：

public class AutoGenerateColumnsImpl implements AutoGenerateColumns {

public void addColumns(TableModel model) {

Iterator iterator = columnsToAdd().iterator();

while (iterator.hasNext()) {

Map columnToAdd = (Map) iterator.next();

Column column = new Column(model);

column.setProperty((String) columnToAdd.get(PROPERTY));

column.setCell((String) columnToAdd.get(CELL));

model.getColumnHandler().addAutoGenerateColumn(column);

}

}

}

示例中columnsToAdd()方法简单返回一个包含生成列（columns）需要的所有信息的集合。 作为参考，下面是我在eXtremeComponents网站实例中使用的columnsToAdd()方法：

private List columnsToAdd() {

List columns = new ArrayList();

columns.add(columnToAdd("fullName", "display"));

columns.add(columnToAdd("nickName", "display"));

columns.add(columnToAdd("term", "display"));

columns.add(columnToAdd("born", "date"));

columns.add(columnToAdd("died", "date"));

columns.add(columnToAdd("career", "display"));

return columns;

}

private Map columnToAdd(String property, String cell) {

Map column = new HashMap();

column.put(Column.PROPERTY, property);

column.put(Column.CELL, cell);

return column;

}

另外，我想声明的是只创建列一次。eXtremeTable为了高效，不会每行创建一列， 而是通过循环持续插入新列值到已经存在的列。记住TableModel能够访问Context，因此 你可以在Controller（Spring）或Action（Struts）中定义样式（look like）并通过request传输集合。 所以你得AutoGenerateColumns实现只需要构建列（Columns）并添加到model.columns里。

## Chapter 14. Utilities

### 14.1. 引言

eXtremeTable包含许多Utility类。在这里我只简单地讨论一些，你可以通过javadocs 得到更多的信息。

### 14.2. HtmlBuilder

封装了所有html语法的类。这个简单类的使你能够写出更干净的html代码，而不用担心null或空字符串。 一个span标签的示例代码如下：

HtmlBuilder html = new HtmlBuilder();

html.span().styleClass(FORM\_BUTTONS).close();

html.append(formButtons);

html.spanEnd();

return html.toString();

## Chapter 15. 1.0.1-M4升级说明

### 15.1. 变更概述

下面的特性已经被更改：

\* HtmlView基于原来代码的实现已经deprecated

\* Cell接口更简单

\* 增加RowTag

\* AutoGenerateColumns变为singleton，更易添加列属性

\* Extended Attributes方法名变更

\* TableTag的collection属性被删除

\* BaseModel更名为TableModel

\* Properties和ResourceBundle现在为Preferences和Messages

\* pageContext被Context接口代替

\* Limit和LimitFactory的语法变更，更易于使用

\* TableTag的saveFilterSort属性被state属性代替

\* ColumnTag的showTotal属性被calc属性代替

\* search图片的名称变为filter

\* FormTag/InputTag为deprecated

\* RetrieveRowsCallbacks、FilterRowsCallback、SortRowsCallback都变为singletons

#### 15.1.1. HtmlView

我把和旧的view相关的代码：原始的view、cell和相关代码放到deprecated文件夹。 原因是新的view代码非常成功，所以没有必要使用旧的代码。使用新代码构建定制view请参考 view包中的HtmlView或CompactView。

#### 15.1.2. Cell

Cell接口已经改变，原因是想结束混乱以提高灵活性。以前对于如何处理区分html和export显示值 不是十分明显。现在Column值设置html，Column的propertyValue设置export。另外因为Column值和 propertyValue值被重写。现在他们在view中是不可见的。

cell现在是singleton并且不再线程安全，因此不要定义任何类变量。改变的原因是为 了Cell接口能更简单地被使用。init()和destroy()方法作为singleton更灵活但是处于一种混乱的状态。

Cell接口如下：

public interface Cell {

/\*\*

\* The display that will be used for the exports.

\*/

public String getExportDisplay(TableModel model, Column column);

/\*\*

\* The html that will be displayed in the table.

\*/

public String getHtmlDisplay(TableModel model, Column column);

}

现在得到导出和html显示存在明显的区别。更重要的，需要返回字符串。列值和属性值不再 需要设置。另一个细微的区别是：BaseModel已经被TableModel取代。这种改变是的不再需要一个 基础包（base package），这意味着不再需要BaseModel。

cell变为singleton不会导致使用复杂，如果你定义了任何类变量只需要把他们放到正确的 方法那么他们就能被任何其他方法使用。

BaseCell被删除因为不再需要添加任何值。替代的是AbstractCell，虚拟方法 getCellValue被用来返回cell的值。这种方法非常容易使用并不需要关心markup。查看 AbstractCell也是有意义的，你会发现这代码实现的多么简单。然而，很多时候需要做的仅仅是 实现Cell接口：

DisplayCell：

public class DisplayCell extends AbstractCell {

public String getExportDisplay(TableModel model, Column column) {

return column.getPropertyValueAsString();

}

protected String getCellValue(TableModel model, Column column) {

return column.getValueAsString();

}

}

AbstractCell：

public abstract class AbstractCell implements Cell {

public String getExportDisplay(TableModel model, Column column) {

return getCellValue(model, column);

}

public String getHtmlDisplay(TableModel model, Column column) {

HtmlBuilder html = new HtmlBuilder();

CellBuilder.tdStart(html, column);

CellBuilder.tdBody(html, getCellValue(model, column));

CellBuilder.tdEnd(html);

return html.toString();

}

/\*\*

\* A convenience method to get the display value.

\*/

protected abstract String getCellValue(TableModel model, Column column);

}

#### 15.1.3. RowTag

RowTag<ec:row> 现在被需要，它被用来替代columns。 现在看来它一直被需要。它不知道表中到底有多少列，最近重构的时候我通过 Table -> Row -> Column使结构固定来合并得到更好的灵活性。将来我可能提供更多的 特性，因为我知道eXtremeTable有着清晰的架构。

典型的eXtremeTable如下：

<ec:table

items="presidents"

var="pres"

action="${pageContext.request.contextPath}/presidents.run"

>

<ec:row>

<ec:column property="name"/>

<ec:column property="term"/>

</ec:row>

</ec:table>

#### 15.1.4. AutoGenerateColumns

AutoGenerateColumns得到了很大的提高，现在你只需要设置你需要的属性。 当你添加列到ColumnHandler使，defaults将别自动调用。

AutoGenerateColumns为singleton并且不是线程安全的，因此不要定义任何类变量。

现在它的实现可能如下：

public class AutoGenerateColumnsImpl implements AutoGenerateColumns {

public void addColumns(TableModel model) {

Iterator iterator = columnsToAdd().iterator();

while (iterator.hasNext()) {

Map columnToAdd = (Map) iterator.next();

Column column = new Column(model);

column.setProperty((String) columnToAdd.get(PROPERTY));

column.setCell((String) columnToAdd.get(CELL));

model.getColumnHandler().addAutoGenerateColumn(column);

}

}

}

#### 15.1.5. Extended Attributes

addExtendedAttributes方法重命名使得如何使用这个特性更清晰。 因此RowTag的addExtendedAttributes现在变为addRowAttributes，ColumnTag变 为addColumnAttributes，TableTag变为addTableAttributes，ExportTag变 为addExportAttributes。另外你参考正确的model bean（它的实现更清晰）， 将知道如何添加属性到你的cell、view.....

使用ExportCsvTag的示例如下：

public void addExportAttributes(Export export) {

String view = export.getView();

if (StringUtils.isBlank(view)) {

export.setView(TableConstants.CSV);

export.setImageName(TableConstants.CSV);

}

export.addAttribute(CsvView.DELIMITER, getDelimiter());

}

为了得到delimiter属性值你只需要从Export bean中get它：

Export export = model.getExportHandler().getCurrentExport();

String delimiter = export.getAttributeAsString(DELIMITER);

现在你需要在ExportTag中覆盖它，你只需要调用setter方法。如果是新的属性，那么使用 可以和前一版一样使用addAttribute()方法。为了得到值，首先从ExportHandler得到Export， 然后调用需要的getter方法。这和使用其它tags一样。

在RowTag和ColumnTag中增加了两个新的callback方法：modifyRowAttributes和 modifyColumnAttributes，因此你可以在rows/columns被处理时改变属性值。

#### 15.1.6. TableTag的collection属性

TableTag的collection属性被删除，现在变为三个新属性：tableId、items和var。 因为我按照标准的JSTL命名，你应该能够根据这些名称知道他们的作用。tableId属性 被用来作为表的唯一标识，items属性用来表示从各种servlet的scopes里取得的集合， var属性表示你将使用EL编写脚本的名称。

依赖你的需要来决定如何使用新的属性，tableId用来唯一标识表。如果你的页面上只使用了 一个eXtremeTable并且不使用Limit特性，那么你根本不需要定义它。默认的表示名为'ec'；如果使用 Limit特性你也可以使用'ec'这个名称。然而，如果的一个JSP页面上同时使用两个eXtremeTables你就 需要使用tableId来唯一标识他们。var属性被用来你将使用EL编写脚本的名称。items属性用来表示从 各种servlet的scopes里取得的集合，而且现在非常健壮。你现在能nest集合到另外的对象，或者nest你想的 深度。

取得集合的语法如下：

<ec:table items="command.myObject.myCol" />

本示例将从命令对象（command object）取得myCol集合。

<ec:table items="myCol" />

本示例中将根据名称自动取得集合，就像以前版本的collection属性一样。

#### 15.1.7. BaseModel

BaseModel被重命名为TableModel，这是因为已经不需要一个基础包（base package）了，这也 意味着不再需要BaseModel了。

#### 15.1.8. Properties和ResourceBundle

web.xml文件中使用extremecomponentsPreferencesLocation属性取代extremecomponentsPropertiesLocation属性， properties现在被Preferences接口控制。在以后的版本，我将提供可选的xml文件配置。

eXtremeTable不再默认在顶层类路径（top level classpath）寻找 extremecomponents.properties文件。你应该在web.xml中使用设置context-param属性的 值为extremecomponentsPreferencesLocation，这将更为通用。

web.xml文件中使用extremecomponentsMessagesLocation属性取代extremecomponentsResourceBundleLocation属性， internationalization现在被Messages接口控制。

Properties和ResourceBundle属性根据table、row、column、export和filterare区分开。 在以前版本中每个属性都以'table'开头，现在每个属性关联到被使用的标签。另外poperty不再需要使用奇怪的下划线语法， 而是使用dot来强调。

下面为properties文件的示例：

table.imagePath=/extremesite/images/\*.gif

table.rowsDisplayed=12

column.parse.date=yyyy-MM-dd

column.format.date=MM/dd/yyyy

column.format.currency=$###,###,##0.00

[英文的属性文件示例如下：]

statusbar.resultsFound={0} results found, displaying {1} to {2}

statusbar.noResultsFound=There were no results found.

toolbar.firstPageTooltip=First Page

toolbar.lastPageTooltip=Last Page

toolbar.prevPageTooltip=Previous Page

toolbar.nextPageTooltip=Next Page

toolbar.filterTooltip=Filter

toolbar.clearTooltip=Clear

toolbar.clearText=Clear

toolbar.firstPageText=First

toolbar.lastPageText=Last

toolbar.nextPageText=Next

toolbar.prevPageText=Prev

toolbar.filterText=Filter

column.headercell.sortTooltip=Sort By

column.calc.total=Total

column.calc.average=Average

#### 15.1.9. pageContext

TableModel （以前的BaseModel）不再直接访问pageContext，取而代之的是使用Context接口， 默认的被pageContext支持。直接访问pageContext是一个不好的实现，Context提供你需要从不同servlet scopes中 取得需要属性值的所有方法。然而，如果你需要直接访问背后的对象，可以使用getContextObject()方法。

#### 15.1.10. Limit和LimitFactory

Limit和LimitFactory现在都是接口，以前版本的实现不如我想象的简单。然而， Limit对象的方法命名和以前版本的一样，因此你以前的代码也能很好的工作。

我两个Limit实现重构为一个，但是仍有两个LimitFactory实现， 从coding的观点用户只要使用一个 实现，但是它必须兼容以前的版本。现在只有一个Limit实现我重命名为TableLimit。同时，因为Limit特性 依赖Context而不是request，我重命名工厂类（TableLimitFactory）来reflect它。

Limit在导出时正确地显示行信息，Limit具有一个isExported()方法。

使用Limit和LimitFactory的示例如下：

Context context = new HttpServletRequestContext(request);

LimitFactory limitFactory = new TableLimitFactory(context, tableId);

Limit limit = new TableLimit(limitFactory);

设置row属性，仅设置totalRows和默认的行显示：

limit.setRowAttributes(totalRows, DEFAULT\_ROWS\_DISPLAYED);

RequestLimitFactory具有另一个如果没有指定tableId将设置为ec的构造函数：

Context context = new HttpServletRequestContext(request);

LimitFactory limitFactory = new TableLimitFactory(context);

#### 15.1.11. TableTag的saveFilterSort属性

saveFilterSort属性被state属性取代，state属性参照State接口并能插接不通的关于 如何保存表状态的实现。

State接口如下：

public interface State {

public void saveParameters(TableModel model, Map parameters);

public Map getParameters(TableModel model);

}

表新增了两个属性：state和stateAttr。state属性使用预设的四种 状态（default、notifyToDefault、persist和notifyToPersist）之一， 你也可以插接自己的实现。default状态不维持任何状态；persist状态没有任何参数传入，将一直维持表的状态； notifyToDefault状态将一直维持表的状态直到你传入参数告诉它回到default状态；notifyToPersist状态 将一直维持当前状态直到你传入参数告诉它维持persisted状态。stateAttr为指定参数提供了一条途径，你 也可以使用属性文件在全局范围内指定它。为了向后兼容，默认参数一直为useSessionFilterSort。

如果你想state按照不同方式工作你只要实现State接口，然后使用TableTag的state属性来指定实现类的 全路径。

#### 15.1.12. ColumnTag的showTotal属性

列新增了两个属性：calc和calcTitle：

<ec:column property="data" calc="total" calcTitle="Total:" />

calc属性实现具有唯一方法的Calc接口：

public interface Calc {

public Number getCalcResult(TableModel model, Column column);

}

它传入model和column，并返回一个Number型的值。默认的实现为总计和平均值。

为了使用定制的Calc，只需要使用ColumnTag的calc属性来指定实现Calc接口的实现类的 全路径。

Calc为singleton并且不是线程安全的，因此不要定义任何类变量。

showTotal因为不再适用在新版中被删除，我也删除了表中的totalTitle。

#### 15.1.13. Image名

search图片名从search变为filter。

#### 15.1.14. FormTag / InputTag Deprecated

FormTag和InputTag现在为deprecated。他们在新的html视图（v iew）中不再被使用。

#### 15.1.15. RetrieveRowsCallbacks、FilterRowsCallback、SortRowsCallback

Callbacks为singleton并且不是线程安全的，因此不要定义任何类变量。