xmlwebgui Documentation

by Lars Trieloff

Documentation for xmlwebgui, a validating xml editor

by Lars Trieloff

Table of Contents

List of Figures

List of Examples

Chapter 1. Installation

System Requirements

xmlwebgui requires the following systems to be already installed in your system:

- *Java Virtual Machine* A Java 1.2 or later compatible virtual machine must be present for usage of xmlwebgui

  ### Tip

  Note that all servlet engines require a JVM to run so if you are already using servlets you already have one installed.

- *Servlet Engine* A Servlet 2.2 compliant servlet engine must be present in order to support servlet operation and dynamic request handling.

If you don't have a servlet engine installed, well, stop right here and go to the Apache Tomcat project http://jakarta.apache.org/tomcat/ [http://jakarta.apache.org/tomcat/] then come back when you are done.

Getting xmlwebgui

You have three choices for getting xmlwebgui: You can eighter download a binary release, or you can download a source release or you get the latest development version from CVS . In the last two cases you have to build xmlwebgui on your own. Please read the appendix Building xmlwebgui from source code

Getting the binary release. You can simply download the binary distribution from the projects download page [http://sourceforge.net/project/showfiles.php?group_id=43829] at sourceforge. Select a file of the type `.zip` .

Getting the source release. You can simply download the source relase from the projects download page [http://sourceforge.net/project/showfiles.php?group_id=43829] at sourceforge. Select a file of the type `Source .zip` .

Getting the latest development version from CVS. xmlwebgui's SourceForge CVS repository can be checked out through anonymous (pserver) CVS with the following instruction set. When prompted for a password for anonymous, simply press the Enter key. Then type **`cvs -d:pserver:anonymous@cvs.xmlwebgui.sourceforge.net:/cvsroot/xmlwebgui login`** for loggin in to the CVS repository. For checking out the source code you will habe to type **`cvs -z3 -d:pserver:anonymous@cvs.xmlwebgui.sourceforge.net:/cvsroot/xmlwebgui co xmlwebgui`** .

Deploying the xmlwebgui web application

In most servlet engines, this is just a matter of copying the `xmlwebgui.war` file in a specific directory and the engine will take care of installing it when restarted. In some servlet engines you will have to change the xml parser used.

Deploying on Tomcat 3.2+. Tomcat currently uses a different version of the XML parser than xmlwebgui. To get xmlwebgui to work, you need to perform the following steps:

1. *Stop Tomcat* Go to the `tomcat/bin` directory, and run the shutdown script.
2. *Delete `tomcat/lib/jaxp.jar`* Tomcat's `jaxp.jar` is 'sealed', and since xerces contains its own implementation of the JAXP standard extension, Java will fail to load xerces and report a 'Package Sealing Violation' if both are in the `classpath`.
3. *Rename `tomcat/lib/parser.jar` to `tomcat/lib/zparser.jar`* Tomcat's `parser.jar` con-

tains older versions of some the same XML APIS that Xerces uses, and these will prevent Xerces from functioning properly if they appear before Xerces in the `classpath`. Since Tomcat's startup scripts automatically load all the `jar` files in `tomcat/lib` in name order, changing the name of the file causes it to be loaded last in the `classpath`.

4. Copy `wmlwebgui.war` to `xmlwebgui.zip` and unpack it to a directory called `xmlwebgui`

5. *Copy the `xmlwebgui/lib/xerces.jar` and `xmlwebgui/lib/xalan.jar` files to `tomcat/lib`* xmlwebgui will now be able to see and use the correct XML libraries.

6. *Copy `xmlwebgui.war` into `tomcat/webapps`*

7. *Start Tomcat* Go to the `tomcat/bin` directory, and run the startup script.

8. *Start using xmlwebgui* Access the URI http://localhost:8080/xmlwebgui/ with a DOM-conformant browser1. Tomcat will unpack the `xmlwebgui.war`.

9. *Edit `xmlwebgui/WEB-INF/web.xml`* In this file you will find a parameter that must be customized to make xmlwebgui run. Find the following part of the file

```
<context-param>
                                          <param-name>rootDir</param-name>
                <!-- Enter the base path for templates and results here -->
                 <param-value>C:/java/tomcat401/webapps/xmlwebgui/</param-value>
</context-param>
<context-param>
                                       <param-name>applicationDir</param-name>
                 <!-- Enter the base path for the application here -->
                 <param-value>C:/java/tomcat401/webapps/xmlwebgui/</param-value>
</context-param>
```

   Edit the paths that it points to your xmlwebgui installation directory.

Deploying on Tomcat 4.x. Tomcat 4.x is a really straight-forward installation

1. *Stop Tomcat* Go to the `tomcat/bin` directory, and run the shutdown script.

2. *Copy `xmlwebgui.war` into `tomcat/webapps`*

3. *Start Tomcat* Go to the `tomcat/bin` directory, and run the startup script.

4. *Start using xmlwebgui* Access the URI http://localhost:8080/xmlwebgui/ with a DOM-conform browser1. Tomcat will unpack the `xmlwebgui.war`.

5. *Edit `xmlwebgui/WEB-INF/web.xml`* In this file you will find a parameter that must be customized to make xmlwebgui run. Find the following part of the file

```
<context-param>
                                          <param-name>rootDir</param-name>
                <!-- Enter the base path for templates and results here -->
                 <param-value>C:/java/tomcat401/webapps/xmlwebgui/</param-value>
</context-param>
<context-param>
                                       <param-name>applicationDir</param-name>
                 <!-- Enter the base path for the application here -->
                 <param-value>C:/java/tomcat401/webapps/xmlwebgui/</param-value>
</context-param>
```

   Edit the path that it points to your xmlwebgui installation directory.

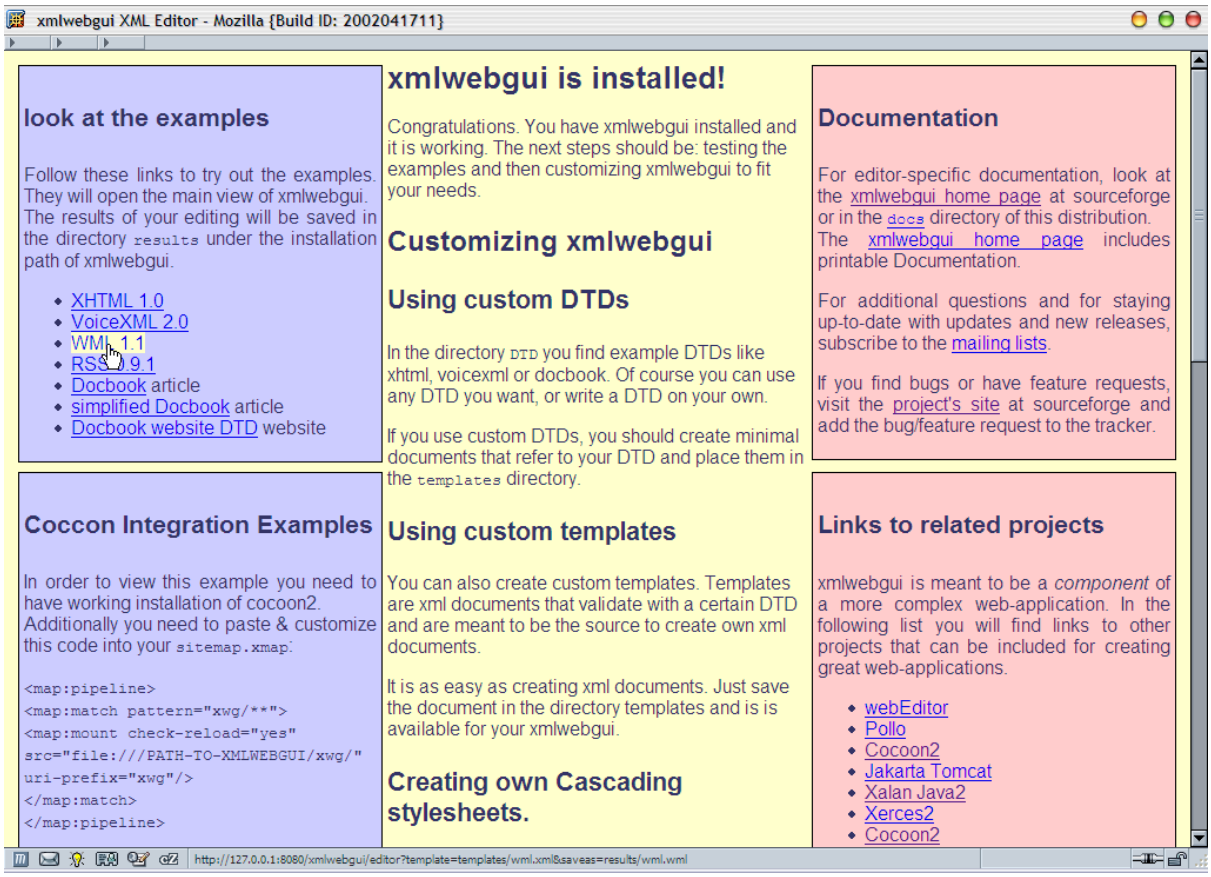
Deploying xmlwebgui on another servlet engine. Please refer to your servlet engine's documentation. FIXME [http://info.astrian.net/jargon/terms/f/FIXME.html]

Chapter 2. Running the examples

Im this chapter we are going to take a look at the example xml templates provided with xmlwebgui. This includes opening and saving of documents, modifying of elements, attributes and cdata sections.

The welcome screen. After having installed xmlwebgui point your DOM-conform bowser [1] to http://localhost:8080/xmlwebgui/ and you will se the following screen

Figure 2.1. The welcome screen



This screen includes links to various examples an to this documentation. This time we will use the wml example, so click on the corresponding link.

selecting an element. Now you are looking at the editing screen of xmlwebgui. This screen contains of the editingwindow at the right side, where you can see the various elements and their text contents. In this example you see a wml element [2] containing a card element, which contains a p element, wich stands for paragraph. At the right side there are four menus, create Child Elements [3] , insert Child Elements [4] , Options and Text/CData [5] . Each element has six menu items. These are select element [6] , edit attributes [7], remove element [8], cut element

1 xmlwebgui makes use of DOM 1 methods and properties as described in Peter Paul Koch's DOM Compatibility Table [http://www.xs4all.nl/~ppk/js/version5.html]. Theese are supported by Mozilla [http://www.mozilla.org], Netscape 6 [http://home.netscape.com/computing/download/] or Microsoft Internet Explorer 5 [http://www.microsoft.com/windows/ie/] or higher.
2 wml is the root element of this document. Every document has exactly one root element.
3 This menu is currently empty, but there will be entries, when you select an element, where child elements can be appended. The corresponding submenu buttons will be enabled automatically by xmlwebgui. When clicking one of this buttons, the element will be appended after the last child of the selected element.
4 This menu is currently empty, but there will be entries, when you select an element, where child elements can be inserted. The corresponding submenu buttons will be enabled automatically by xmlwebgui. When clicking one of this buttons, the element will be inserted berfore the selected element.
5 This menu allows you to insert text to elements.
6 selects the clicked element
8 removes the element from the document

9, copy element 10 and an indicator for the content model 11.

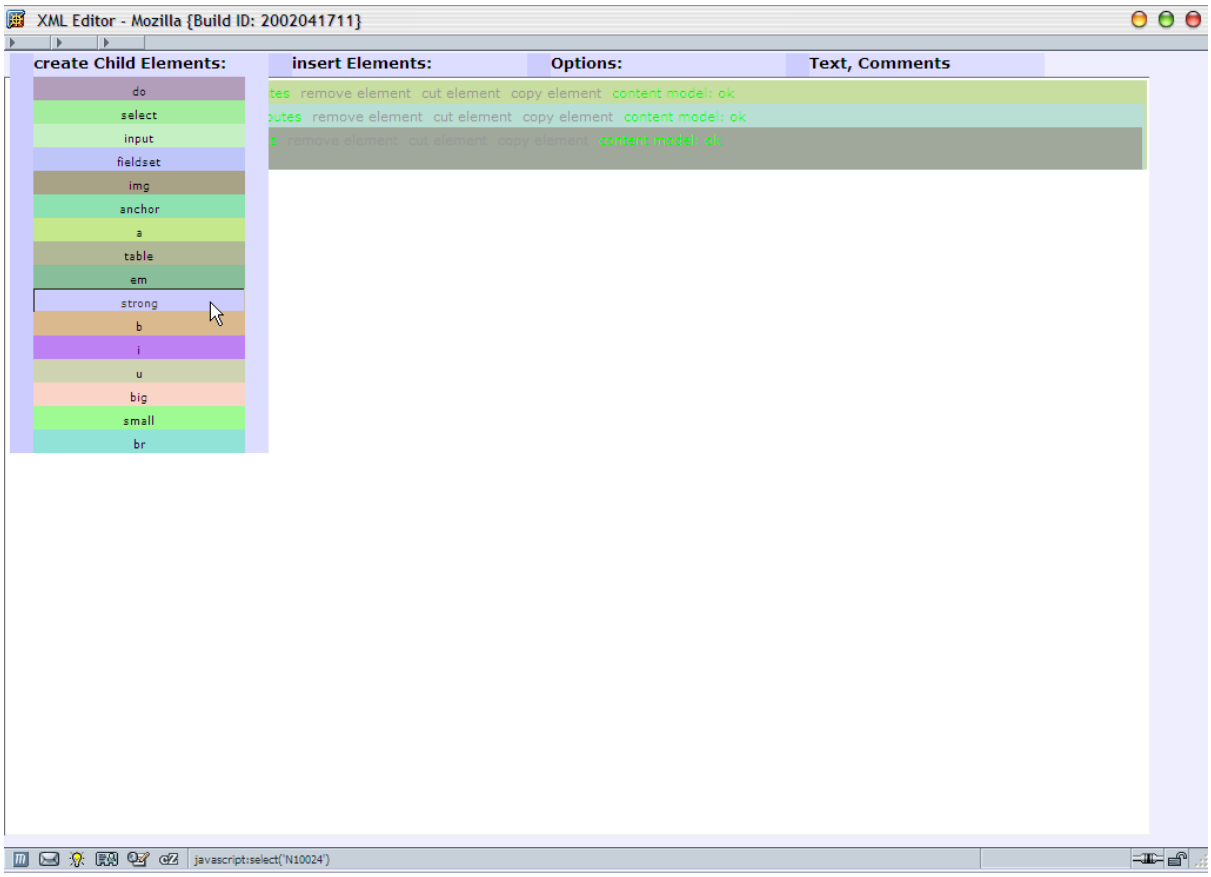
Figure 2.2. selecting an element

selecting the p element

Click on the select element link of the p element to select it.

inserting a child element. After selecting an element, in the menu create Child Elements appear all types of elements, which can be inserted as child elements of the selected p. Click on the button strong to insert a strong element.

Figure 2.3. inserting a child element

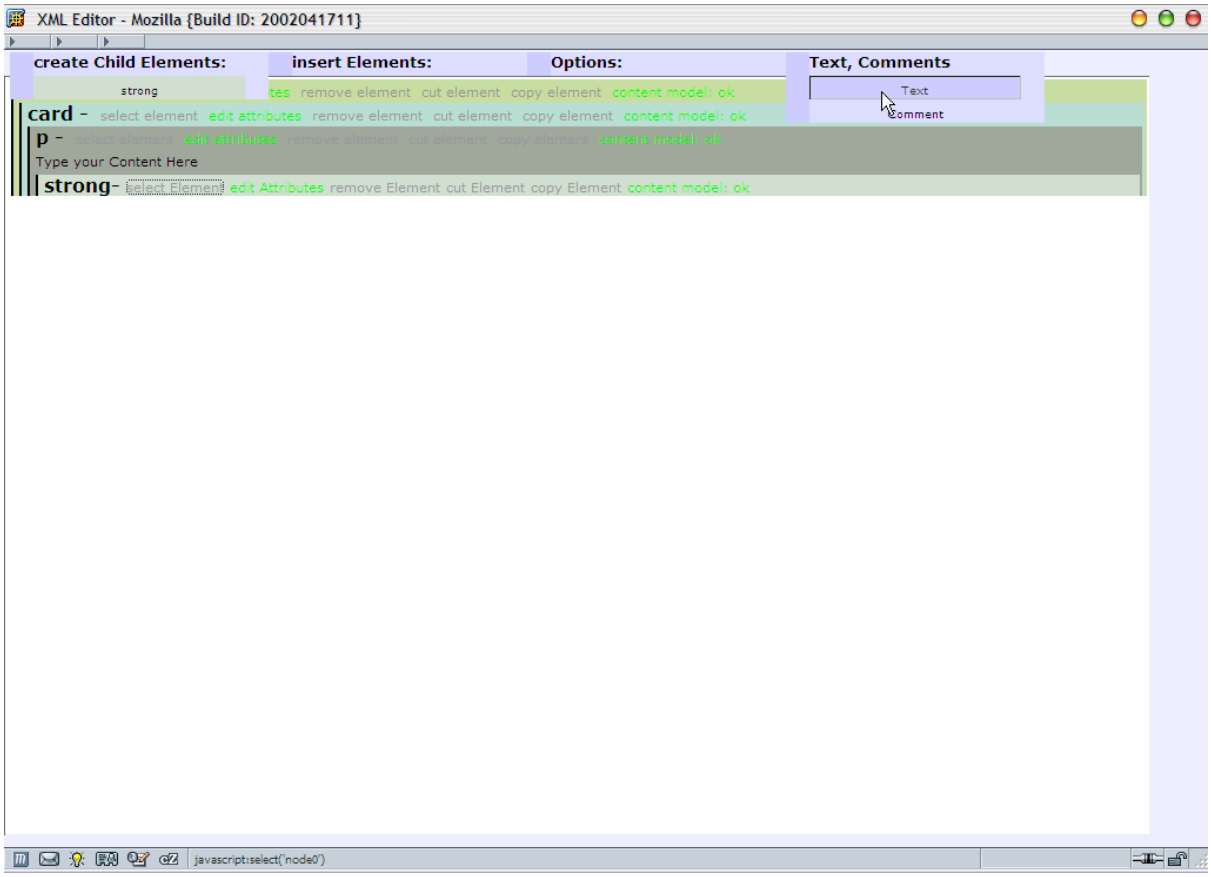


After clicking, in the p element will appear a strong element. This strong element has a red marked content model indicator. This means that you will have to insert more elements or a textnode. This is what is done in the next step.

7 shows or hides the list of this element's attributes. This is also an indicator, which is green, if all attributes that are required have a value, and which is red, if not all attributes which need to have a value assigned, really have one assigned.
9 cuts the element, it can be inserted again with Paste Before (inserts before the selected element) or Paste Child (inserts as last child of the selected element) from the Options menu.
10 copies the element, it can be inserted again with Paste Before (inserts before the selected element) or Paste Child (inserts as last child of the selected element) from the Options menu.
11 If this indicator is green, everything is ok, the element has the right count and order of child elements. If it has a red background, you will have to take a look at the documentation for this kind of xml files, which child elements are allowed and wich are required inside this element.

inserting text to an element. In this step you will insert a textnode in the newly-created strong element. Therefore you have to select the strong element by clicking on it's select element link.

Figure 2.4. inserting text



Now click on the button PCDATA in the Text/CDATA menu. This will insert a small paragraph containing the text *cdata* in the strong element. In the next step you will change the content of the paragraphs textnode.

changing textual content. This is very simple. Just click on the text you want to change, it will turn from a html- paragraph to a html-textarea, you enter the text of your choice and press the Ready-button to save your changes.

Figure 2.5. selecting text to be changed

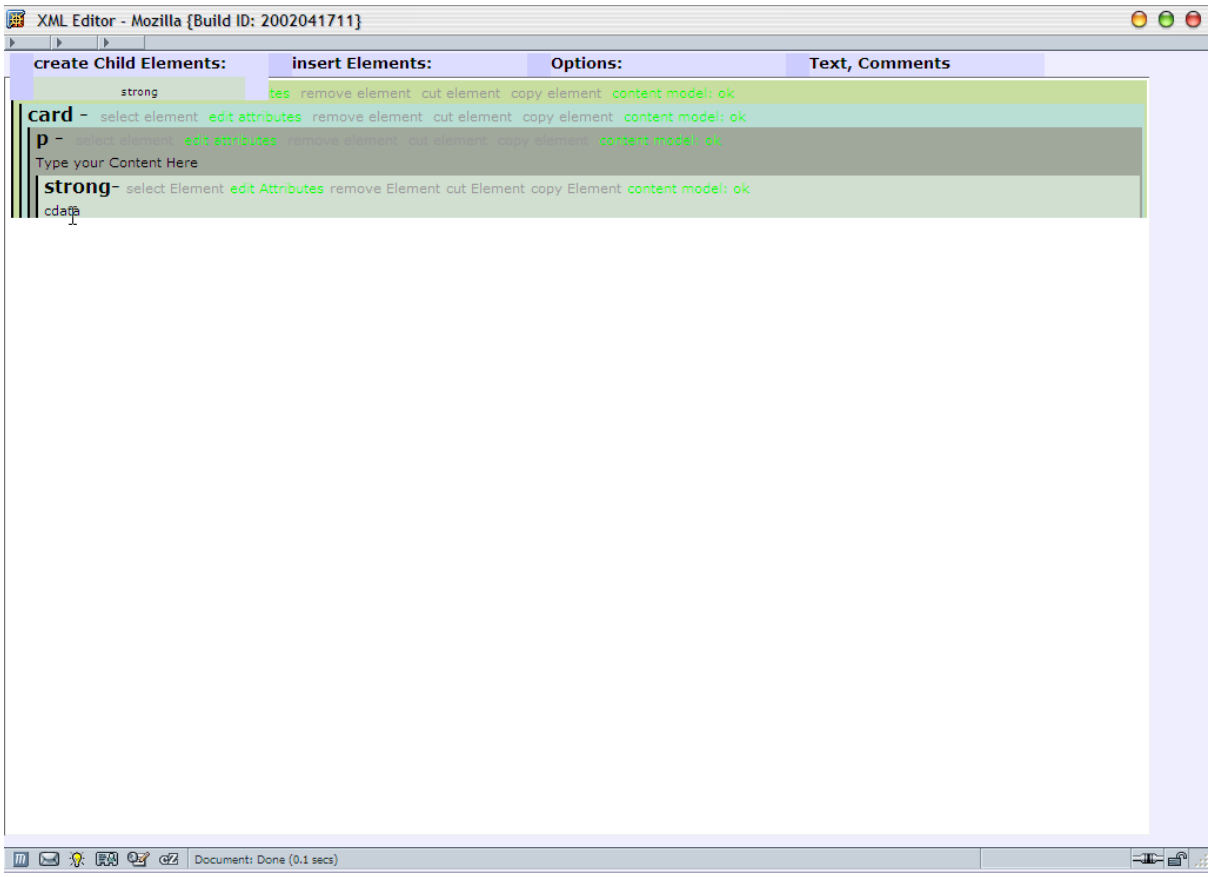
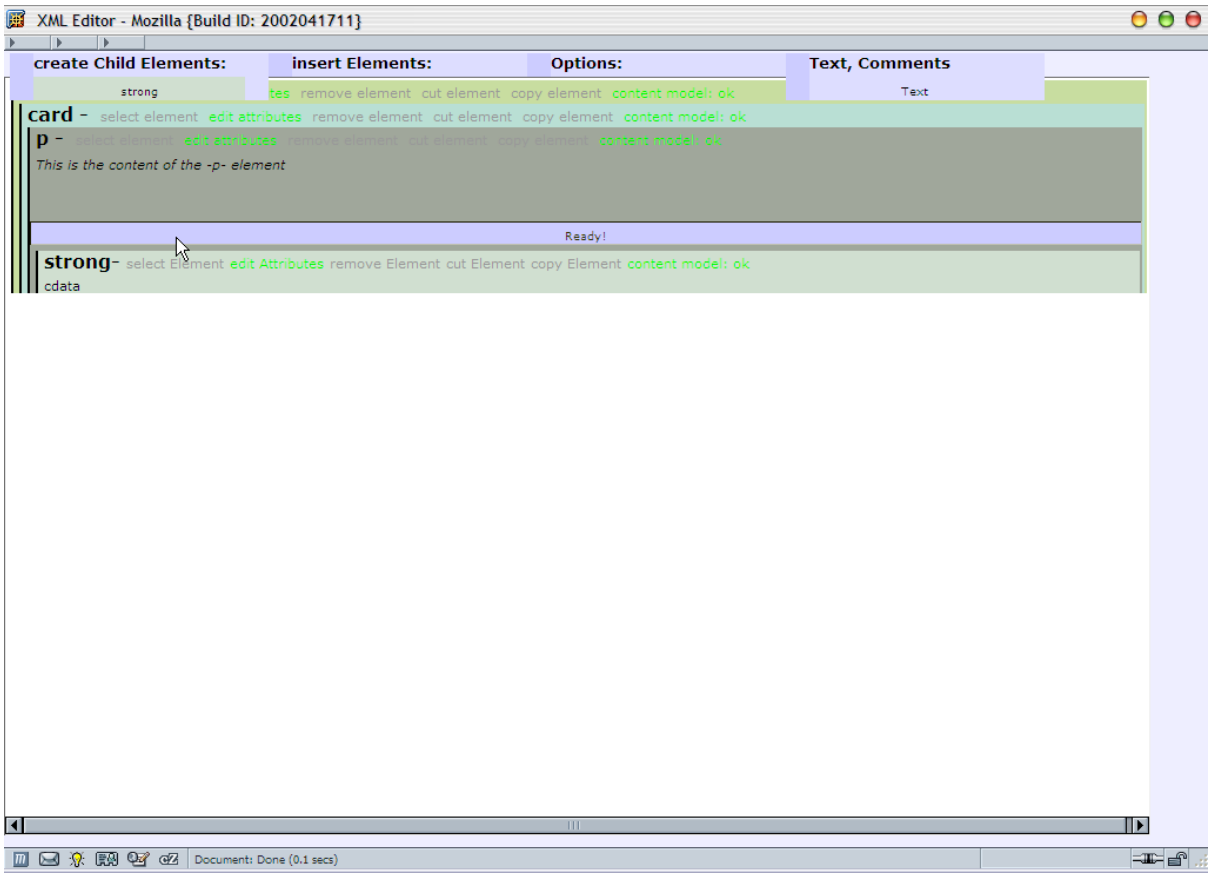


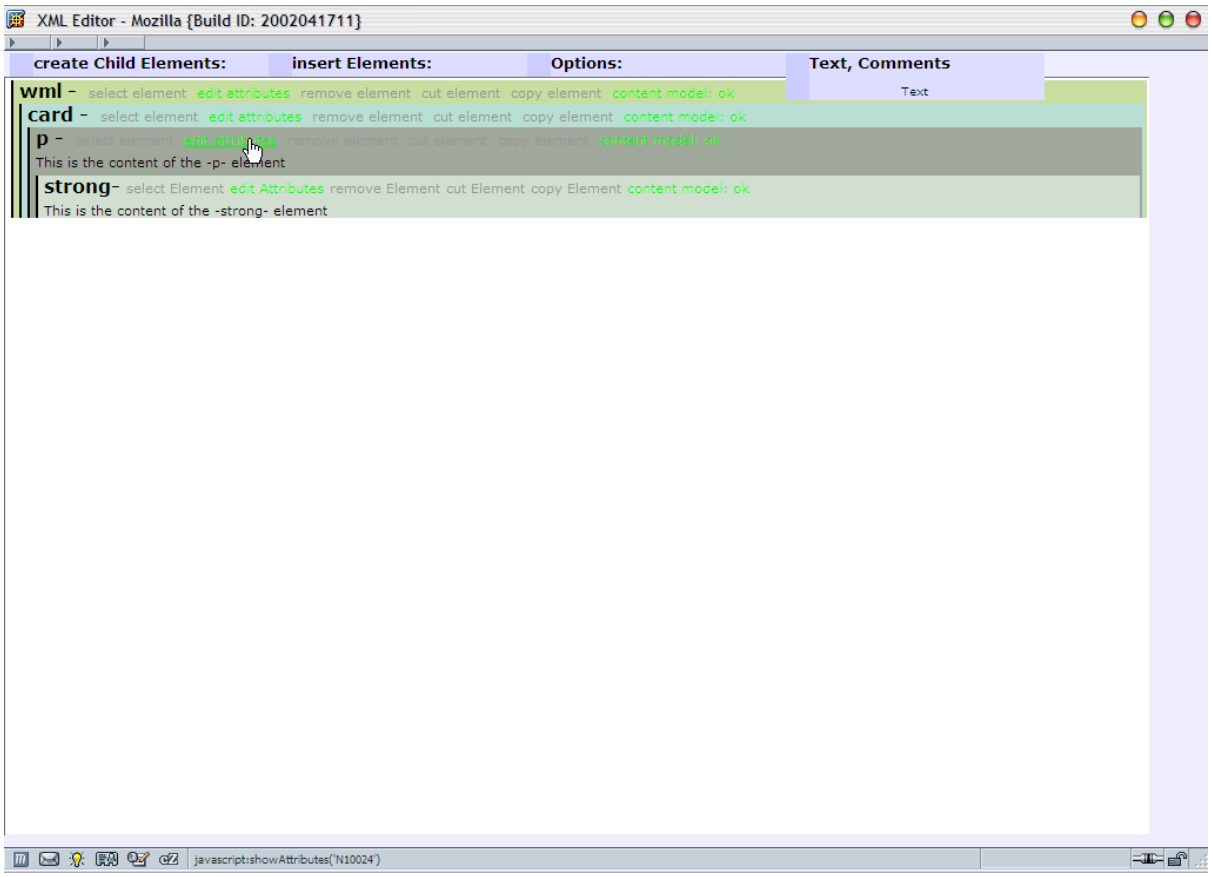
Figure 2.6. saving changes to the text



If you delete the whole text in the textarea, the textnode will be removed from the document. You can change the text of the strong element the same way.

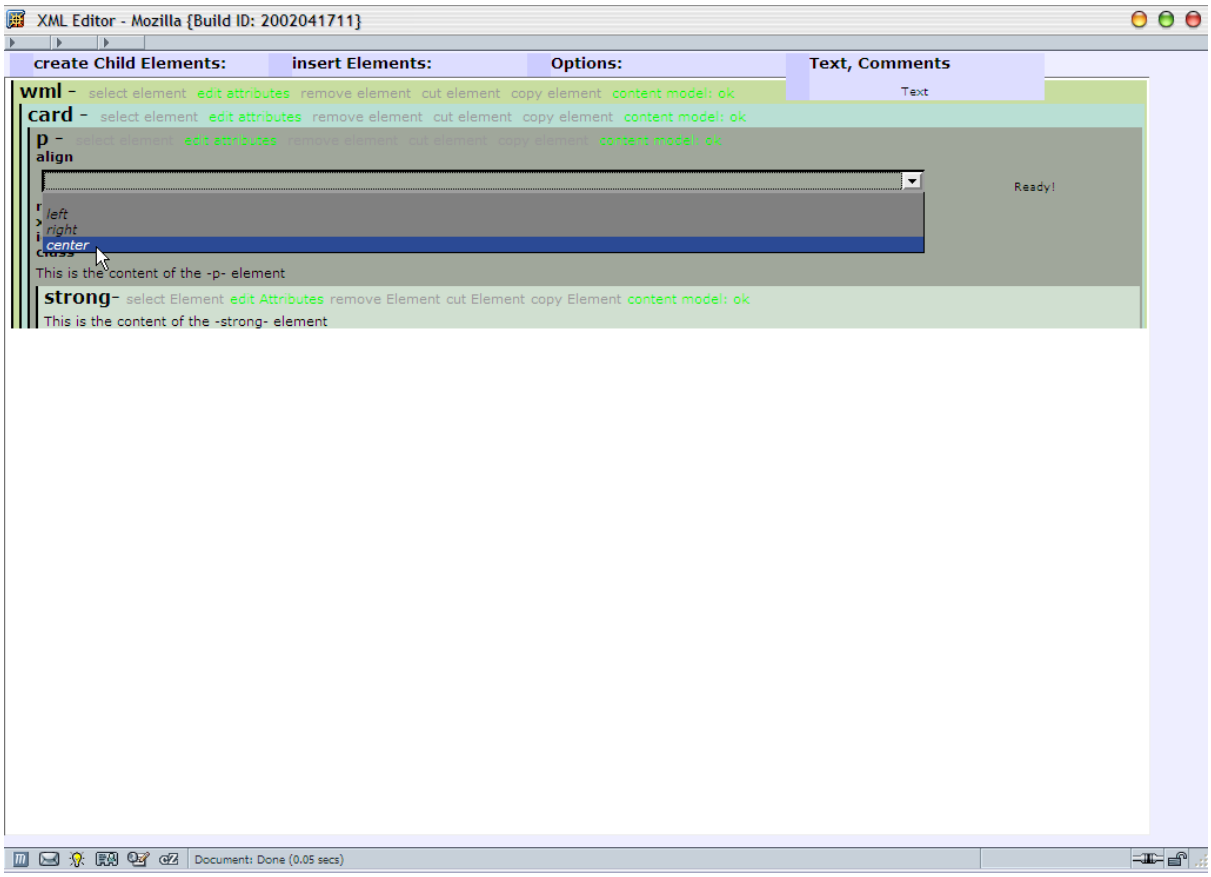
modifying attribute values. In order to modify the attribute values of an element, click on the link edit attributes , this will make the list of attributes visible.

Figure 2.7. showing the attributes



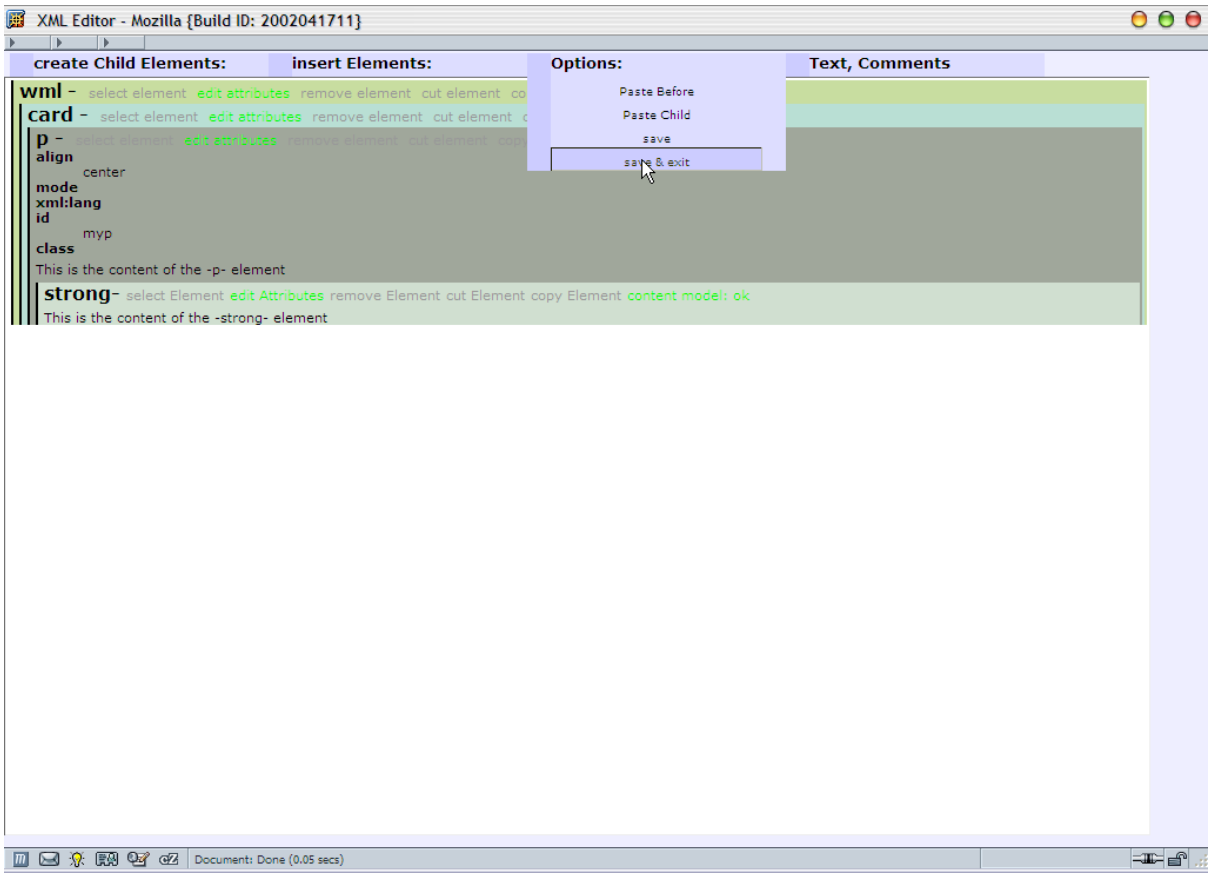
If you click this link again, it will hide the list of attributes. Editing attributes is quite the same like editing text nodes, just click on the value of the attribute or it's name, and an input field or an select box will appear. After having entered the right value click the Ready-Button again.

Figure 2.8. choosing the right attribute value



Saving changes. After having edited the document, you can save the changes by clicking on save to save and stay in the document or save & exit to return to the welcome screen.

Figure 2.9. saving changes



The modified document

If you are interested, what you have clicked and typed in the last steps of this example open the file `results/wml.wml` in your texteditor. What you will see looks like this:

```
<?xml                          version="1.0"                          encoding="UTF-8"?>
<!DOCTYPE   wml   PUBLIC   "-//WAPFORUM//DTD   WML   1.1//EN"   "../DTD/wml/wml1.dtd">
<wml>
                                                                              <card>
                      <p                    id="myp"                    align="center">
           This      is      the      content      of      the      -p-      Element
                                                                              <strong>
             This      is      the      content      of      the      -strong-element
                                                                            </strong>
                                                                                </p>
                                                                              </card>
</wml>
```

This is the code of your document, all indents, line breaks and white spaces have been removed, but reinserted for convienence in this example.

Chapter 3. Customizing xmlwebgui

URL parameters

URL parameters are important for telling xmlwebgui which templates you want to be opened, which CSS should be used for displaying the elements and under which filename the result should be saved. Following parameters are used:

template	specifies the template which is going to be used. The filename is relative to the rooDir as defined in `WEB-INF/web.xml`
styleurl	specifies the stylesheet which will style the elements. The filename is relative to the rooDir as defined in `WEB-INF/web.xml`
saveas	Under which filename should the result be stored. The filename is relative to the rooDir as defined in `WEB-INF/web.xml`

Every other parameter is passed through the xmlwebgui. This allows to preserve sessionids when switching from your application to xmlwebgui and back to your application.

Setting the servlet's parameters

By setting the parameters of the xmlwebgui servlet you can control it's behaviour to fit into your application framework.

setting the rootDir parameter

The `rootDir` parameter should point to the directory where XML Web GUI gets it's templates from and saves it's restlts to.

security issues

Please remember that XML Web GUI adresses templates and results via relative paths and that moving out of the `rootDir` is for security reasons not allowed.

Example 3.1. Default configuration

<context-param> <param-name>rootDir</param-name> <!-- Enter the base path for templates and results here --> <param-value>C:/java/tomcat401/webapps/xmlwebgui/</param-value> </context-param>

User-Management and security

the session validator. The session validator is another security feature. This parameter should point to the URL of a servlet or script of your application which accepts a POST-request with your application's common session id included. This servlet should resolve the username or groupname of the user this session belongs to and return it.

For the case that you want to run XML Web GUI standalone or your application does not support multiple users you will not need to change anything because a “Dummy-Session-Validator” is inclued in the XML Web GUI distribution

Example 3.2. the default session validator

<context-param> <param-name>sessionValidatorURL</param-name> <!-- Enter the url for validating the session here --> <param-value>http://localhost:8080/xmlwebgui/username</param-value> </context-param>

defining the name of your session-id parameter. Edit this parameter that it matches your application's session-id parameter name.

In most cases this is something like `session` or `sid`.

Example 3.3. the default session id parameter name

This is the default configuration of the session-id parameter name <context-param> sessionIdPa- <param-name>rameterName</param-name> <!-- Enter the variable name of your webapp's session id here --> <param-value>sessionid</param-value> </context-param>

the users.xml configuration file

In the `users.xml` configuration file the rights of users or usergroups are stored. After getting the username from the session-validator XML Web GUI will look up this file to find out, if the user is allowed to read and write this file or not.

The rules for specifing user rights are defined in `users.dtd` which is in the same directory.

The file contains the root element `users` which contains a sequence of at least one `user` elements. Each `user` element needs to have the attribute `name` specified. This is an *ID*-attribute which means that every `name` can only occur once in a document.

The content of `user` is a sequence of one `read` and one `write` element. Each `read` or `write` element contains a sequence of `allow` and `disallow` elements. Theese elements are empty and have the `pattern` attribute which contains a regular expression which the allowed or disallowed path has to match.

Example 3.4. unrestricted access

This example gains unlimited access to the filesystem for the user `root`. <user name="root"> <read> <allow pattern=".*" /> </read> <write> <allow pattern=".*" /> </write> </user>

exit and error URL

The last two parameters `exitUrl` and `errorUrl` point to an URL that is accessed after saving and exiting XML Web GUI or if an error occurs.

Using custom DTDs

XML Web GUI comes with a variety of DTDs, but of course you can use any other DTD for your documents or even write an own.

For a wide choice of XML Applications [http://dmoz.org/Computers/Data_Formats/Markup_Languages/XML/Applications/] and DTDs the open direc-

tory project has an own category.

If you would like to write your own DTD, the ZVON DTD Tutorial [http://www.zvon.org/xxl/DTDTutorial/General/book.html] is a recommended ressource.

Loading external DTD

You have three way to adress a DTD

- **External HTTP reference.** You can adress a DTD by referring to the URL of a DTD on another server.

Example 3.5. adressing the Docbook DTD

http://www.oasis-open.org/committees/docbook/xml/4.1.2/docbookx.dtd

The advantage is that you can be sure that the DTD is always up-to-date. The disadvantage is that every access to XML Web GUI will cause network traffic.

- **File reference.** You can also adress a DTD by referring to a file in your filesystem.

Example 3.6. adressing the WML DTD

file:///c:/java/tomcat401/webapps/xmlwebgui/DTD/wml/wml1.dtd

The advantage of this is that it is very fast and requires no HTTP transfer. The disadvantage is that the created XML files are not distributable unless on the client's computer there is the DTD at the same path.

- **Internal HTTP reference.** The recommended adressing of DTD is via HTTP, but to your own server. (Or the server XML Web GUI is running on).

Example 3.7. adressing the Simplesite DTD

http://127.0.0.1:8080/xmlwebgui/DTD/simplesite/page.dtd

If you repace 127.0.0.1 with the ip-adress or domain-name of your server, the the DTD will be available for validating as long as your server is running and XML Web GUI will not need a network conncection to validate the files.

Using custom templates

What is a template. Templates are valid XML files with a reference to a DTD.

You can use any valid XML file with a reference to a DTD as template. The template that will be opened can be specified by the URL-parameter template.

Defining own Cascading Stylesheets

For each DTD you can assign a CSS styleseet. This stylesheet can control how an element will be displayed in XML Web GUI 's editing window.

The rules are quite simple. For every element defined in the DTD you can add a style rule by typing

```
div.divnameOfTheElement {
    your style rules
}
```

For the element `html` of the XHTML DTD the code will look as following:

Example 3.8. DTDstyle for XHTML

```
.divhtml {
    background:#996633;
}
```

changing XML Web GUI's global appearance. In the installation directory of XML Web GUI you will find a file called `editor.css`. You can also modify this file to make XML Web GUI fitting your needs.

Modifying the entry page

This is quite obvious. You can replace the entry page by every other html page you like.

Defining own XSL Stylesheets

With changing the XSL stylesheets you can funadmentally change XML Web GUI's appearance.

Caution

A lot of XML Web GUI's application logic lies in a defined arrangement of html tags. If you change the XSL stylesheets, XML Web GUI may not work any more.

FIXME [http://info.astrian.net/jargon/terms/f/FIXME.html]

Chapter 4. Building xmlwebgui from source code

Requirements for building from source

- The sources. The sources can be obtained at the xmlwebgui website [http://].
- A Java Software Development Kit (SDK). Most common the the Sun Java Software Development Kit, look at java.sun.com [http://java.sun.com] for further information.
- `servlet.jar` this java archive contains all classes and interfaces needed for compiling the servlet-specific part of xmlwebgui. It is included in your servlet engine.
- `xalan.jar` and `xercesImpl.jar` and `xml-apis.jar` and `xmlParserAPIs.jar` This files are needed for the xml-specific part of xmlwebgui They are included in the source release of xmlwebgui, but it is useful to download Apache Xalan from the Apache Software Foundtaion [http://xml.apache.org] because you will also need a xslt engine for transforming the documentation.
  For transforming the documentation you can use any other XSLT-engine. A very good engine is Saxon [http://saxon.sourceforge.net] by Michael Kay.
- Apache Ant. This build tool from the Apache Software Foundtaion can be found at the Jakarta Project Site [http://jakarta.apache.org].
- For transforming the documentation you will also need to download the docbook xslt stylesheets from The Docbook Open Repository [http://docbook.sourceforge.net].

Building XML Web GUI.

1. Download the lastest source distribution from the XML Web GUI homepage [http://xmlwebgui.sourceforge.net]
2. Unpack the distribution to a directory and enter this directory on the command line
3. Then type **ant war** for building the web-archive for JDK1.3 or type **ant war_le** for building the web-archive for Java 1.4
4. continue the installation as described here: Chapter 1

Building the documentation.

1. download and unpack the documentation source files.
2. Use your favorite XSLT-emgine to transform `documentation.xml` using the docbook sytlesheets to HTML or XSL:FO.

FIXME [http://info.astrian.net/jargon/terms/f/FIXME.html]

Appendix A. Apache Software License

Version 1.1

Copyright © 2000 The Apache Software Foundation. All rights reserved.

Copyright 2000 The Apache Software Foundation. All rights reserved. © Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment:

   "This product includes software developed by the Apache Software Foundation (http://www.apache.org/ [http://www.apache.org])."

   Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. 4. The names "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.

5. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <http://www.apache.org/ [http://www.apache.org]>.

Portions of this software are based upon public domain software originally written at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign.