Sendmail Logicsheet (2.1 legacy document)

Table of contents

Table of contents

Warning:
This document was copied as is from the Cocoon 2.1 documentation, but has not yet been fully reviewed or moved to its new home.

1. Description

The Sendmail logicsheet (taglib) is a XSP logicsheet that wraps XML tags around the operation of sending an email message. Specifically, the Sendmail logicsheet provides an XML interface to the primary methods of the Java Mail API for sending an internet mail including the ability to attach any binary data files to the message (see the Java Mail API) for more information.

The Sendmail logicsheet is an alternative to the Sendmail action.

2. Usage

As an XSP logicsheet, the Sendmail logicsheet can only be used in an XSP page. It may be helpful to be familiar with XSP before working with this (or any) logicsheet.

Since the Sendmail logicsheet is not activated in the default Cocoon setup, a couple of steps must be taken before an email can be send.

First of all Cocoon must have been compiled with the required Java Mail API libraries. The libraries mail.jar from the Java Mail distribution and the library activation.jar from the Java Activation Framework have to be copied to the location lib/local of Cocoon's source distribution. Cocoon must then be recompiled.

Before the Sendmail logicsheet can be used, some setup in cocoon.xconf is required. See, if the following fragment is already existing.

```
<builtin-logicsheet> <parameter name="prefix" value="sendmail"/> <parameter name="uri" value="http://apache.org/cocoon/sendmail/1.0"/> <parameter name="href" value="resource://org/apache/cocoon/components/language/markup/xsp/java/sendmail.xsl"/> </builtin-logicsheet>
```

If it is not present, it is easiest to simply locate the entry xsp-response for the Response logicsheet and put the above fragment before the <builtin-logicsheet> of the Response logicsheet entry. This can either be done before recompilation or later, when Cocoon is already deployed. If done later, Cocoon must be reloaded.

To use the Sendmail logicsheet, you must first declare the *sendmail* namespace, mapping it to the uri *http://apache.org/cocoon/sendmail/1.0*. These steps will result in code like this:

```
<xsp:page language="java" xmlns:xsp="http://apache.org/xsp" xmlns:sendmail="http://apache.org/cocoon/sendmail/1.0"> ... </xsp:page>
```

You may then use any of the elements in the *sendmail* namespace described in the Elements Reference section below.

3. Example Code

The following code shows an example of using the Sendmail logicsheet. A HTML form is used, to post information about updated documentation to some imaginary mailing list. The XSP page is invoked from a HTML form like this.

```
<form action="/cocoon/xsp/mail/send-a-mail" method="post" enctype="multipart/form-data"> <input type="text" name="subject" size="56" /> ... <input type="text" name="url1" size="56" />
```

... <input type="text" name="url1" size="56" /> ... <input type="file" name="uploaded_file1" size="56" /> ... <input type="file" name="uploaded_file2" size="56" /> ... <textarea name="changes" rows="5" cols="72"> </textarea> ... </form>

Since the form allows to attach upto two arbitrary files, it is important, that enctype="multipart/form-data" is used. This is the XSP code, that is invoked:

<?xml version="1.0" encoding="ISO-8859-1"?> <xsp:page language="java" xmlns:xsp="http://apache.org/xsp" xmlns:xsp-request="http://apache.org/xsp/request/2.0" xmlns:sendmail="http://apache.org/cocoon/sendmail/1.0"> <email> <xsp:logic> StringBuffer body = new StringBuffer(); body.append(" Documentation:\n----------\n"); body.append("URL: "); body.append(<xsp-request:get-parameter name="url1"/>); body.append("\n"); body.append("URL: "); body.append(<xsp-request:get-parameter name="url2"/>); body.append("\n\n"); body.append(" Changes:\n----------\n"); body.append(<xsp-request:get-parameter name="changes"/>); body.append("\n\n"); </xsp:logic> <sendmail:send-mail> <sendmail:from>from address</sendmail:from> <sendmail:to>some maillinglist address</sendmail:to> <sendmail:subject><xsp-request:get-parameter name="subject"/></sendmail:subject> <!-- Uncomment to override defaults from cocoon.xconf <sendmail:smtphost>localhost</sendmail:smtphost> <sendmail:smtpuser>localhost</sendmail:smtpuser> <sendmail:smtppassword>localhost</sendmail:smtppassword> --> <sendmail:body><xsp:expr>body.toString()</xsp:expr></sendmail:body> <sendmail:attachment> <sendmail:param name="object"><xsp:expr>request.get("attachment")</xsp:expr></sendmail:param> </sendmail:attachment> <sendmail:attachment url="context://welcome.xml" mime-type="text/plain" name="foo.txt"/> <sendmail:attachment url="cocoon:///" mime-type="text/html" name="welcome.html"/> <sendmail:on-success> Email successfully sent. </sendmail:on-success> <sendmail:on-error> An error occurred: <sendmail:error-message/> </sendmail:on-error> </sendmail:send-mail> </email> </xsp:page>

Cocoons feature to automatically disassemble the incoming request puts the uploaded files automatically into the upload directory and the files are accessible through the uploaded_file[12] request parameters (make sure, that autosave-uploads is set to true in the WEB-INF/web.xml file of the Cocoon context). By using <xsp:expr>request.get("req-param")</xsp:expr> you actually get an object of class org.apache.cocoon.servlet.multipart.Part. The <sendmail:send-mail> fragment is replaced with an <error> element, if an error occurs during the sending of the message.

Another noteworthy item is the formatting of the body text through a Java StringBuffer. Any formatting, that would be placed inside the <sendmail:body> element would be lost due to the internal workings of the Sendmail logicsheet.

4. Elements Reference

Element Name	Attributes/Child Elements	Description
sendmail:attachment		Sets the attachment for this email. Attributes for setting name, mime-type, or an URL (e.g. using cocoon:-protocol!). Parameters can be set dynamically using <sendmail:param/> syntax. If an object is to be attached, it

		must be set this way. Use the following expression to obtain the correct object from the request: <xsp:expr>request.get("req-param")</xsp:expr>.
sendmail:bcc		Sets the recipients of a blind carbon copy of the email. This can be a list of comma separated email addresses.
sendmail:body		Sets the body text of the message.
sendmail:cc		Sets the recipients of a carbon copy of this email. This can be a list of comma separated email addresses.
sendmail:charset		Sets the character set for encoding the message. This tag has only an effect, if no attachments are send.
sendmail:from		Sets the from address of the message.
sendmail:smtphost		The IP-address or the name of the host, which should deliver the email message. Better known as the mail transfer agent or short MTA.
sendmail:subject		Sets the subject line of the message.
sendmail:to		Sets the destination/to address of the email message. This can be a list of comma separated email addresses.

5. Hint

Please read this hint, since it applies here as well.

1. Comments

add your comments