gs.content.email.base Documentation

Release 2.3.4

GroupServer.org

January 14, 2016

Contents

Author Michael JasonSmith

Contact Michael JasonSmith <mpj17@onlinegroups.net>

Date 2015-07-24

Organization GroupServer.org

Contents:

CHAPTER 1

Skinning

For GroupServer installations with only one site then skinning is simple: the email messages will be skinned using the same skin-name as is used on the web (see the documentation on configuring a web proxy and GroupServer).

See also:

GroupServer ships with two alternate skins by default gs.skin.blue and gs.skin.green.

Skinning is more difficult for complex installs where multiple sites with different skins are handled by the same server. Below we discuss *the problem* and *the solution*.

1.1 The problem

The problem is that email comes into one site, so the `request` object for all email uses the skin for that site. This is fine if all sites in the GroupServer install use the same skin. However, this is an issue if different sites use different skins, as the email messages will look like they are from the wrong site.

1.2 The solution

The solution is to label the site with the skin-name. This skin is then retrieved and applied to the `request` object. (The `zope.traversing` subsystem does this for web-requests.)

The skin-name is recorded in the `emailSkin` property of the `DivisionConfig` object, or the `GlobalConfiguration` object (the former over-rides the latter). Set this property to the skin name, such as `gs_green` for the GroupServer green skin, or `gs_blue` for the blue skin. (The default is grey.) While you could use a different skin from that used on the web, this is discourage because it is likely to confuse the members of your site.

CHAPTER 2

gs.content.email.base API

There are two main parts to the API for `gs.content.email.base`. The *notifiers* coordinate the rendering and sending of the message, while the *message bodies* (plural) define it.

2.1 Notifiers

All the notifiers inherit from the `NotifierABC` abstract base-class. However, if a notification is going to someone who does not have a profile (they are *anonymous*) then an *anonymous notifier* is used.

2.1.1 Anonymous notifier

The anonymous notifier provides some methods that are useful for creating an email message that is sent to someone that has no profile.

2.1.2 Examples

The *Digest on* notifier (`gs.group.member.email.settings.notifier.DigestOnNotifier`) first inherits from the abstract base-class and defines the names of the two pages that will render the HTML and plain-text versions of the pages.

The `notify` method defines the subject (which is translated using `zope.i18n.translate()`) and then renders the plain-text and HTML versions of the email body.

Finally, the `gs.profile.notify.MessageSender` class is used to send the notification to the relevant person. It is this class that actually constructs the email message out of the two bodies, and the subject, before sending it off. Finally the `Content-type` is reset.

```
class DigestOnNotifier(GroupNotifierABC):
    htmlTemplateName = 'gs-group-member-email-settings-digest-on.html'
    textTemplateName = 'gs-group-member-email-settings-digest-on.txt'

    def notify(self, userInfo):
        subject = _('digest-on-subject',
                    'Topic digests from ${groupName}',
                    mapping={'groupName': self.groupInfo.name})
        translatedSubject = translate(subject)
        text = self.textTemplate()
        html = self.htmlTemplate()
```

```
        sender = MessageSender(self.context, userInfo)
        sender.send_message(translatedSubject, text, html)
        self.reset_content_type()
```

The *Not a member* notification (`gs.group.member.leave.notifier.NotMemberNotifier`) is sent to someone when he or she tries to leave a group but they are not a member. Initially it is very similar to a standard notification, except the `From` address is generated. Also more values are passed to the templates because they are less able to use the context to determine the values. Then `AnonymousNotifierABC.createMessage()` is used to generate the method, before the message is sent using `gs.email.send_email()`.

```
class NotMemberNotifier(AnonymousNotifierABC):
    htmlTemplateName = 'gs-group-member-leave-not-a-member.html'
    textTemplateName = 'gs-group-member-leave-not-a-member.txt'

    def notify(self, groupInfo, toEmailAddress):
        fromAddr = self.fromAddr(groupInfo.siteInfo)
        subject = _('leave-request-problem-subject',
                    'Request to leave ${groupName}',
                    mapping={'groupName': groupInfo.name})
        translatedSubject = translate(subject)
        html = self.htmlTemplate(emailAddress=toEmailAddress,
                                 groupName=groupInfo.name,
                                 groupURL=groupInfo.url)
        text = self.textTemplate(emailAddress=toEmailAddress,
                                 groupName=groupInfo.name,
                                 groupURL=groupInfo.url)

        message = self.create_message(toEmailAddress, fromAddr,
                                      translatedSubject, text, html)
        send_email(groupInfo.siteInfo.get_support_email(),
                   toEmailAddress, message)
        self.reset_content_type()
```

2.2 Message bodies

There are two views. *Site email* is analogous to the `SitePage` [1], while *group email* is the email-equivalent of `GroupPage` [2]. However, both views over-write the `__call__` method.

The `__call__` method of a page is called to produce the rendered form of the view. Both views defined here take the output from the super-class and run it through `premailer.transform` [3] before returning it. This results in HTML that can be sent as a MIME-attachment.

2.2.1 Site email

The `SiteEmail` class is used by messages made from outside the Group context. For example:

```
<browser:page
  name="notification.html"
  for="Products.GSContent.interfaces.IGSSiteFolder"
  class="gs.content.email.base.SiteEmail"
```

[1] See <https://github.com/groupserver/gs.content.base/>
[2] See <https://github.com/groupserver/gs.group.base/>
[3] See <https://pypi.python.org/pypi/premailer/>

```
  template="browser/templates/notification.pt"
  permission="zope2.View"/>
```

2.2.2 Group email

The `GroupEmail` class is a subclass of `SiteEmail` that is used by messages made from within the Group context. For example:

```
<browser:page
  name="notification.html"
  for="gs.group.base.interfaces.IGSGroupMarker"
  class="gs.content.email.base.GroupEmail"
  template="browser/templates/notification.pt"
  permission="zope2.View"/>
```

In addition to the `siteInfo` attribute, this class defines the `groupInfo` attribute, which contains the name, URL, and ID of the current group.

2.2.3 Text bodies

The plain-text version of the email bodies are supported by a *mixin*.

2.2.4 Example

The *Left* notification is sent to a group administrator when a member leaves a group. It is, for the most-part, simple except for the `get_support_email()` method.

Notifications typically end with a link to email the support group. This `mailto:` normally fills the `Subject` (providing a standard topic in the support-group) and body of the message (providing information that the person seeking support may forget to provide). The `Subject` and `body` are translated using `zope.i18n.translate()`, and assembled into a `mailto:` URI using the `SiteEmail.mailto()` method.

```
class LeftHTMLNotification(GroupEmail):
    'The notification to the administrator that a member has left'

    def __init__(self, group, request):
        super(LeftHTMLNotification, self).__init__(group, request)
        self.group = group

    def get_support_email(self, user, admin):
        subject = _('support-notification-member-left-subject',
                    'A member left my group')
        translatedSubject = translate(subject)
        uu = '{}{}'.format(self.siteInfo.url, user.url)
        au = '{}{}'.format(self.siteInfo.url, admin.url)
        body = _('support-notification-member-left-body',
                 'Hello,\n\nA member left my group, ${groupName}, and...'
                 '\n\n--\nThese links may be useful:\n'
                 '  Group   ${groupUrl}\n'
                 '  Me      ${adminUrl}\n'
                 '  Member  ${userUrl}\n',
                 mapping={'groupName': self.groupInfo.name,
                          'groupUrl': self.groupInfo.url,
                          'adminUrl': au, 'userUrl': uu})
```

```
        translatedBody = translate(body)
        retval = self.mailto(self.siteInfo.get_support_email(),
                             translatedSubject, translatedBody)
        return retval
```

The plain-text version of the same body is provided by the `TextMixin` class. It does little other than generating a filename for the page, and setting the correct header.

```
class LeftTXTNotification(LeftHTMLNotification, TextMixin):

    def __init__(self, group, request):
        super(LeftTXTNotification, self).__init__(group, request)
        filename = 'left-{0}-{1}.txt'.format(self.siteInfo.id,
                                             self.groupInfo.id)
        self.set_header(filename)
```

CHAPTER 3

Changelog

3.1 2.3.4 (2015-12-11)

- Fixing the unit-tests

3.2 2.3.3 (2015-10-16)

- Using `gs.core.mailto` to construct the mailto

3.3 2.3.2 (2015-09-21)

- Using `subject` rather than `Subject` in `mailto:` URIs

3.4 2.3.1 (2015-09-17)

- Fixing the colour codes for IBM Notes, based on a hack by Eli Dickinson

3.5 2.3.0 (2015-09-16)

- Extending the look up of the skin name — so it uses the `GlobalConfiguration` object as well as the `DivisionConfiguration` object

3.6 2.2.0 (2015-07-24)

- Improving support for GroupServer installs with multiple sites, by looking up the skin name in the `emailSkin` property of the `DivisionConfig` object

3.7 2.1.3 (2015-06-02)

- Keeping the CSS classes when processing the notification

3.8 2.1.2 (2015-04-23)

- Dropping the `lxml` dependency, as I now understand that the style element sometimes appears for the styles that cannot be in-lined

3.9 2.1.1 (2014-11-10)

- Changing the logging-level for `cssutils` to `CRITICAL`, thanks to the answer by Felix Carmona on Stack Overflow

3.10 2.1.0 (2014-09-15)

- Added some Sphinx documentation
- Added the abstract base-classes for the notifiers:
  - `NotifierABC`
  - `GroupNotifierABC`
  - `AnonymousNotifierABC`
- Added `SiteEmail.mailto()`

3.11 2.0.0 (2014-05-29)

- Added unit tests
- Switched to Unicode literals
- Fixed the code for generating `<base>` element in `SiteEmail`

3.12 1.1.2 (2014-02-20)

- Fixing the HTTP headers when viewing the plain text notification
- Following `to_unicode_or_bust` to `gs.core`

3.13 1.1.1 (2013-10-23)

- Fixing the arguments and keyword arguments to the page template

3.14 1.1.0 (2013-10-10)

- Added the `TextMixin` class
- Pass the arguments and keyword arguments to the super-class

- Better handling of plain-text and Unicode

3.15 1.0.0 (2013-08-26)

- Initial version

This product — in combination with the `gs.content.email.layout` and `gs.content.email.css` — provides support for HTML-formatted email messages going out from GroupServer. The HTML in the *page templates* fills in areas defined by the `layout` module. The page is then styled by the `css` module. This module coordinates the process, providing the code to processes the resulting HTML and CSS into a message that can be sent by email. (The actual sending is done by either the `gs.email.send_email()` function, or the `gs.profile.notify.MessageSender` class.)

CHAPTER 4

Resources

- Documentation: https://groupserver.readthedocs.org/projects/gscontentemailbase/
- Code repository: https://github.com/groupserver/gs.content.email.base/
- Questions and comments to http://groupserver.org/groups/development
- Report bugs at https://redmine.iopen.net/projects/groupserver

CHAPTER 5

Indices and tables

- genindex
- modindex
- search