plone.app.discussion Documentation

Release 2.0

2010, Timo Stollenwerk - Plone Foundation

September 30, 2011

CONTENTS

Contents:

CHAPTER

ONE

ARCHITECTURAL PRINCIPLES

This document outlines architectural principles used in the design of plone.app.discussion.

Discussion items have a portal_type This makes it easier to search for them and manage them using existing CMF and Plone UI constructs.

Discussion items are cataloged It is possible to search for discussion items like any other type of content.

Discussion items are subject to workflow and permission Moderation, anonymous commenting, and auto approve/reject should be handled using workflow states, automatic and manual transitions, and permissions.

Discussion items are light weight objects Discussion item objects are as light weight as possible. Ideally, a discussion item should be as lightweight as a catalog brain. This may mean that we forego convenience base classes and re-implement certain interfaces. Comments should not provide the full set of dublin core metadata, though custom indexers can be used to provide values for standard catalog indexes.

Optimise for retrival speed HTML filtering and other processing should happen on save, not on render, to make rendering quick.

Settings are stored using plone.registry Any global setting should be stored in plone.registry records.

Forms are constructed using extensible z3c.form forms This allows plugins (such as spam protection algorithms) to provide additional validation. It also allows integrators to write add-ons that add new fields to the comment form.

Discussion items are stored in a BTree container This allows faster lookup and manipulation.

Discussion items are accessed using a dict-like interface This makes iteration and manipulation more natural. Even if comments are not stored threaded, the dict interface should act as if they are, i.e. calling items() on a comment should return the replies to that comment (in order).

Discussion items are retrieved in reverse creation date order Discussion items do not need to support explicit ordering. They should always be retrieved in reverse creation date order (most recent for). They can be stored with keys so that this is always true.

Discussion items do not need readable ids Ids can be based on the creation date.

Discussion items send events The usual zope.lifecycleevent and zope.container events are fired when discussion items are added, removed, or modified.

CHAPTER

TWO

DESIGN NOTES

This document contains design notes for plone.app.discussion.

2.1 Storage and traversal

For each content item, there is a Conversation object stored in annotations. This can be traversed to via the ++conversation++ namespace, but also fetched via an adapter lookup to IConversation.

The conversation stores all comments related to a content object. Each comment has an integer id (also representable as a string, to act as an OFS id and allow traversal). Hence, traversing to obj/++conversation++/123 retrieves the comment with id 123.

Comments ids are assigned in order, so a comment with id N was posted before a comment with id N + 1. However, it is not guaranteed that ids will be incremental. Ids must be positive integers - 0 or negative numbers are not allowed.

Threading information is stored in the conversation: we keep track of the set of children and the parent if any comment. Top-level comments have a parent id of 0. This information is managed by the conversation class when comments are manipulated using a dict-like API.

Note that the __parent__/acquisition parent of an IComment is the IConversation, and the __parent__/acquisition parent of an IConversation is the content object.

2.2 Events

Manipulating the IConversation object should fire the usual IObjectAddedEvent and IObjectRemovedEvent events. The UI may further fire IObjectCreatedEvent and IObjectModifiedEvent for comments.

2.3 Factories

Comments should always be created via the ‘Discussion Item’ IFactory utility. Conversations should always be obtained via the IConversation adapter (even the ++conversation++ namespace should use this). This makes it possible to replace conversations and comments transparently.

2.4 The Comment class

The inheritance tree for DiscussionItem is shown below. Classes we want to mix in and interface we want to implement in the Comment class are marked with [x].

```
[ ] DiscussionItem
    [ ] Document
        [ ] PortalContent                       = [ ] IContentish
            [ ] DynamicType                     = [ ] IDynamicType
            [ ] CMFCatalogAware                 = [ ] <no interface>
            [ ] SimpleItem                      = [ ] ISimpleItem
                [ ] Item                          [ ]
                    [?] Base                    = [ ] <no interface>
                    [ ] Resource                = [ ] <no interface>
                    [ ] CopySource              = [ ] ICopySource
                    [ ] Tabs                    = [ ] <no interface>
                    [x] Traversable             = [ ] ITraversable
                    [ ] Element                 = [ ] <no interface>
                    [x] Owned                   = [ ] IOwned
                    [ ] UndoSupport             = [ ] IUndoSupport
                [ ] Persistent                    [ ]
                [ ] Implicit                      [ ]
                [x] RoleManager                 = [ ] IRoleManager
                    [ ] RoleManager             = [ ] IPermissionMappingSupport
        [ ] DefaultDublinCoreImpl               = [ ] IDublinCore
                                                  [ ] ICatalogableDublinCore
                                                  [ ] IMutableDublinCore
            [ ] PropertyManager                 = [ ] IPropertyManager
```

Thus, we want:

- **Traversable, to get absolute_url() and friends**
    - this requires a good acquisition chain at all times
- **Acquisition.Explicit, to support acquisition**
    - we do not want implicit acquisition
- Owned, to be able to track ownership
- RoleManager, to support permissions and local roles

We also want to use a number of custom indexers for most of the standard metadata such as creator, effective date etc.

Finally, we'll need event handlers to perform the actual indexing.

2.5 Discussion settings

Discussion can be enabled per-type and per-instance, via values in the FTI (allow_discussion) and on the object. These will remain unchanged. The IConversation object's 'enabled' property should consult these.

Global settings should be managed using plone.registry. A control panel can be generated from this as well, using the helper class in plone.app.registry.

Note that some settings, notably those to do with permissions and workflow, will need to be wired up as custom form fields with custom data mangers or similar.

2.6 Workflow and permissions

Where possible, we should use existing permissions:

- View

- Reply to Item
- Modify Portal Content
- Request Review

In addition, we'll need a 'Moderator' role and a moderation permission,

- Moderate comment
- Bypass moderation

To control whether Anonymous can post comments, we manage the 'Reply to Item' permission. To control whether moderation is required for various roles, we could manage the 'Bypass moderation' permission.

These could work in a workflow like this:

```
* --> [posted] -- {publish} --> [published]--> *
         |                           ^
         |                           |
         +----- {auto-publish} -----+
         |                           |
         +----- {auto-moderate} ----+
```

The 'posted' state is the initial state. 'published' is the state where the comment is visible to non-reviewers.

The 'publish' transition would be protected by the 'Moderate comment' permission. We could have states and transition for 'rejected', etc, but it is probably just as good to delete comments that are rejected.

The 'auto-publish' transition would be an automatic transition protected by the 'Bypass moderation' permission.

The 'auto-moderate' transition would be another automatic transition protected by an expression (e.g. calling a view) that returns True if the user is on an auto-moderation 'white-list', e.g. by email address or username.

2.7 Forms and UI

The basic commenting display/reply form is placed in a viewlet.

The reply form is dynamically created right under the comment when the user hits the reply button. To do so, we copy the standard comment form with a jQuery function. This function sets the form's hidden in_reply_to field to the id of the comment the user wants to reply to. This also makes it possible to use z3c.form validation for the reply forms, because we can uniquely identify the reply form request and return the reply form with validation errors.

Since we rely on JavaScript for the reply form creation, the reply button is removed for non JavaScript enabled browsers.

The comment form uses z3c.form and plone.z3cform's ExtensibleForm support. This makes it possible to plug in additional fields declaratively, e.g. to include SPAM protection.

CHAPTER

THREE

PERMISSIONS AND WORKFLOWS

This document describes how plone.app.discussion handles permissions and workflows.

3.1 Introduction

plone.app.discussion uses permissions and workflows to control what a user is allowed to do. It tries to use the default Plone permissions and workflow engine as much as possible.

3.2 Permissions

plone.app.discussion knows two permissions to control what a user is allowed to do. The ‘Reply to item’ permission to control who is allowed to post a comment on a content object and the ‘Review comments’ permission to control who is allowed to review comments.

1. **Permission to post a comment**:

   The permission to post a comment is controlled by the ‘Reply to item’ permission. By default, this permission is granted to the ‘Member’, ‘Reviewer’, and ‘Manager’ role.

2. **Permission to review comments**:

   The permission to review comments is controlled by the ‘Review comments’ permission. By default, this permission is granted to the ‘Reviewer’ and ‘Manager’ role.

3.2.1 Changing permissions

If you want to change the way plone.app.discussion allows users to post or review comments you can do that by changing which permissions are granted to which rules. In Plone permissions are always granted to roles, not to users directly.

For instance, if you want to allow users without the ‘Member’ role to post comments, you have to grant the ‘Reply to item’ permission to the ‘Authenticated’ role.

Or, if you don’t want to allow ‘Reviewers’ to review comments anymore, you can just remove the ‘Review comments’ permission from the ‘Reviewer’ role.

Note: For a general introduction to permissions and roles in Plone see:

http://plone.org/documentation/kb/understanding-permissions/permissions-and-roles http://plone.org/products/dexterity/documentation/manual/developer-manual/advanced/permissions

3.3 Workflows

plone.app.discussion ships with a simple one-state workflow and a review workflow for comments:

1. **Comment Single State Workflow**:

   Essentially a workflow with no transitions, but it has a published state, so portlets and applications that expect that state will continue to work.

2. **Comment Review Workflow**: A simple review workflow for comments

   A simple review workflow that comes with two states (pending and published) and a single transition (publish).

   The 'pending' state is the initial state. 'published' is the state where the comment is visible to everyone and non-editable.

   The 'publish' transition is protected by the 'Review comments' permission.

   ```
   * --> [pending] -- {publish} --> [published]--> *
   ```

Note: For a general introduction to workflows in Plone see: http://plone.org/documentation/kb/creating-workflows-in-plone/

3.3.1 Custom comment workflow

You can create and enable any custom workflow on the "comment" content type. Though, there are some special hooks in plone.app.discussion that check if the workflow that is enabled for the "comment" content type has a 'pending' state in order to do the following things:

1. A portal message will be shown to the user after posting a comment, if the comment just entered the 'pending' state.
2. A message is shown to the user if he/she accesses the bulk moderation view and workflow is enabled for comments that does not implement a 'pending' state.
3. A moderator will only be emailed when comment moderation is enabled in the discussion control panel and the comment workflow contains a 'pending' state.

CHAPTER

FOUR

CAPTCHA PLUGIN ARCHITECTURE

This document contains design notes for the plone.app.discussion Captcha plugin architecture.

4.1 Introduction

When a Captcha plugin (e.g. plone.formwidget.captcha or plone.formwidget.recaptcha) is installed, plone.app.discussion extends the comment form with a Captcha field/widget and a Captcha validator.

The form extender and validator are only registered if there is a plugin installed that claims to provide the "plone.app.discussion-captcha" feature in its configure.zcml file:

```
<configure
    xmlns:meta="http://namespaces.zope.org/meta"
    xmlns:zcml="http://namespaces.zope.org/zcml">

    <!-- Declare that plone.formwidget.captcha provides a Captcha field that
         can be used by plone.app.discussion to add a Captcha field to comment
         forms. -->
    <meta:provides feature="plone.app.discussion-captcha" />

</configure>
```

Note: Currently plone.formwidget.captcha and plone.formwidget.recaptcha claim to provide such a feature. If you want to write your own Captcha plugin, it has to provide this feature as well.

See Also:

- https://svn.plone.org/svn/plone/plone.formwidget.captcha/trunk/plone/formwidget/captcha/meta.zcml
- https://svn.plone.org/svn/plone/plone.formwidget.recaptcha/trunk/plone/formwidget/recaptcha/meta.zcml

4.2 CaptchaExtender

The CaptchaExtender class extends the comment form with a Captcha field and widget. The CaptchaExtender currently uses either the CaptchaFieldWidget from plone.formwidget.captcha or the ReCaptchaFieldWidget from plone.formwidget.recaptcha. If you want to write your own Captcha solution, you have to override the update() method of the CaptchaExtender or write your own CaptchaExtender class.

```
class CaptchaExtender(extensible.FormExtender):
    """Extends the comment form with a Captcha. This Captcha extender is only
    registered when a plugin is installed that provides the
    "plone.app.discussion-captcha" feature.
    """
    adapts(Interface, IDefaultBrowserLayer, CommentForm) # context, request, form

    fields = Fields(ICaptcha)

    def __init__(self, context, request, form):
        self.context = context
        self.request = request
        self.form = form

        registry = queryUtility(IRegistry)
        settings = registry.forInterface(IDiscussionSettings, check=False)
        self.captcha = settings.captcha
        portal_membership = getToolByName(self.context, 'portal_membership')
        self.isAnon = portal_membership.isAnonymousUser()

    def update(self):
        if self.captcha != 'disabled' and self.isAnon:
            # Add a captcha field if captcha is enabled in the registry
            self.add(ICaptcha, prefix="")
            if self.captcha == 'captcha':
                from plone.formwidget.captcha import CaptchaFieldWidget
                self.form.fields['captcha'].widgetFactory = CaptchaFieldWidget
            elif self.captcha == 'recaptcha':
                from plone.formwidget.recaptcha import ReCaptchaFieldWidget
                self.form.fields['captcha'].widgetFactory = ReCaptchaFieldWidget
            elif self.captcha == 'norobots':
                from collective.z3cform.norobots import NorobotsFieldWidget
                self.form.fields['captcha'].widgetFactory = NorobotsFieldWidget
            else:
                self.form.fields['captcha'].mode = interfaces.HIDDEN_MODE
```

4.3 CaptchaValidator

The CaptchaValidator class provides custom versions of the plone.formwidget.captcha and the plone.formwidget.recaptcha validators. It does this, because we want to be able to have more than one Captcha solution installed in one Plone instance. We also want to be able to easily switch between different Captcha implementations inside a single Plone instance.

Therefore we have to check which Captcha solution is enabled in the discussion control panel and use only the selected Captcha validator. It is not enough to check if a Captcha plugin is just installed, because there could be more than one.

We do two checks. First we check for a suitable Captcha solution (check for the plone.app.discussion-captcha feature, see notes above). Second, we check which Captcha solution is enabled in the discussion control panel and apply the corresponding field validator.

The plone.app.discussion captcha validator always checks for a view with the name of the captcha plugin. For instance, if plone.formwidget.captcha is enabled it checks for a “captcha” view.

```
class CaptchaValidator(validator.SimpleFieldValidator):
    implements(IValidator)
    adapts(Interface, IDiscussionLayer, Interface, IField, Interface)
    #       Object, Request, Form, Field, Widget,
    # We adapt the CaptchaValidator class to all form fields (IField)

    def validate(self, value):
        super(CaptchaValidator, self).validate(value)

        registry = queryUtility(IRegistry)
        settings = registry.forInterface(IDiscussionSettings, check=False)

        if settings.captcha in ('captcha', 'recaptcha', 'norobots'):
            captcha = getMultiAdapter((aq_inner(self.context), self.request),
                                      name=settings.captcha)
            if not captcha.verify(input=value):
                if settings.captcha == 'norobots':
                    raise WrongNorobotsAnswer
                else:
                    raise WrongCaptchaCode
            else:
                return True


# Register Captcha validator for the Captcha field in the ICaptcha Form
```

CHAPTER

FIVE

EMAIL NOTIFICATION

This document describes the plone.app.discussion email notification feature.

5.1 Introduction

plone.app.discussion allows users and administrators to be notified about new comments by email. There are two kinds of email notification:

1. **User notification**: Tell users when a comment has been added.

   This method composes and sends emails to all users that have added a comment to this conversation and enabled user notification.

   This requires the user_notification setting to be enabled in the discussion control panel.

2. **Moderator notification**: Tell the moderator when a comment needs attention.

   This method sends an email to the site admin (mail control panel setting) if comment moderation is enabled and a new comment has been added that needs to be approved.

   This requires the moderator_notification to be enabled in the discussion control panel and the comment_review_workflow enabled for the comment content type.

Note: The user notification feature requires z3c.form >= 2.3.3.

5.2 User Notification

plone/app/discussion/comment.py

```
def notify_user(obj, event):
    """Tell users when a comment has been added.

       This method composes and sends emails to all users that have added a
       comment to this conversation and enabled user notification.

       This requires the user_notification setting to be enabled in the
       discussion control panel.
    """

    # Check if user notification is enabled
    registry = queryUtility(IRegistry)
    settings = registry.forInterface(IDiscussionSettings, check=False)
    if not settings.user_notification_enabled:
        return

    # Get informations that are necessary to send an email
    mail_host = getToolByName(obj, 'MailHost')
    portal_url = getToolByName(obj, 'portal_url')
    portal = portal_url.getPortalObject()
    sender = portal.getProperty('email_from_address')

    # Check if a sender address is available
    if not sender:
        return

    # Compose and send emails to all users that have add a comment to this
    # conversation and enabled user_notification.
    conversation = aq_parent(obj)
    content_object = aq_parent(conversation)

    # Avoid sending multiple notification emails to the same person
    # when he has commented multiple times.
    emails = set()
    for comment in conversation.getComments():
        if (obj != comment and
            comment.user_notification and comment.author_email):
            emails.add(comment.author_email)

    if not emails:
        return

    subject = translate(_(u"A comment has been posted."),
                        context=obj.REQUEST)
    message = translate(Message(
            MAIL_NOTIFICATION_MESSAGE,
            mapping={'title': safe_unicode(content_object.title),
                     'link': content_object.absolute_url() +
                             '/view#' + obj.id,
                     'text': obj.text}),
            context=obj.REQUEST)
    for email in emails:
        # Send email
        try:
            mail_host.send(message,
                           email,
                           sender,
                           subject,
                           charset='utf-8')
        except SMTPException:
            logger.error('SMTP exception while trying to send an ' +
                         'email from %s to %s',
                         sender,
                         email)
```

5.3 Moderator Notification

plone/app/discussion/comment.py

```
def notify_moderator(obj, event):
    """Tell the moderator when a comment needs attention.

       This method sends an email to the moderator if comment moderation a new
       comment has been added that needs to be approved.

       The moderator_notification setting has to be enabled in the discussion
       control panel.

       Configure the moderator e-mail address in the discussion control panel.
       If no moderator is configured but moderator notifications are turned on,
       the site admin email (from the mail control panel) will be used.
    """
    # Check if moderator notification is enabled
    registry = queryUtility(IRegistry)
    settings = registry.forInterface(IDiscussionSettings, check=False)
    if not settings.moderator_notification_enabled:
        return

    # Get informations that are necessary to send an email
    mail_host = getToolByName(obj, 'MailHost')
    portal_url = getToolByName(obj, 'portal_url')
    portal = portal_url.getPortalObject()
    sender = portal.getProperty('email_from_address')

    if settings.moderator_email:
        mto = settings.moderator_email
    else:
        mto = sender

    # Check if a sender address is available
    if not sender:
        return

    conversation = aq_parent(obj)
    content_object = aq_parent(conversation)

    # Compose email
    subject = translate(_(u"A comment has been posted."), context=obj.REQUEST)
    message = translate(Message(MAIL_NOTIFICATION_MESSAGE_MODERATOR,
        mapping={
            'title': safe_unicode(content_object.title),
            'link': content_object.absolute_url() + '/view#' + obj.id,
            'text': obj.text,
            'link_approve': obj.absolute_url() + '/@@moderate-publish-comment',
            'link_delete': obj.absolute_url() + '/@@moderate-delete-comment',
            }),
        context=obj.REQUEST)

    # Send email
    try:
        mail_host.send(message, mto, sender, subject, charset='utf-8')
    except SMTPException, e:
        logger.error('SMTP exception (%s) while trying to send an ' +
                     'email notification to the comment moderator ' +
                     '(from %s to %s, message: %s)',
                     e,
                     sender,
                    mto,
                    message)
```

5.4 Event Subscribers

Email notifications are triggered by event subscribers that are called when a comment has been added to a page.

Note: In Plone 3, the event subscribers were located in the zope.lifecycleevent package. They moved to zope.app.container in Plone 4. Therefore plone.app.discussion registers one of the two subscribers, dependent on the Plone version.

To create custom email notifications, register a new event subscriber or override an existing one.

plone/app/discussion/notifications.zcml

```
<configure
    xmlns="http://namespaces.zope.org/zope"
    xmlns:zcml="http://namespaces.zope.org/zcml"
    i18n_domain="plone.app.discussion">

  <subscriber
      for="plone.app.discussion.interfaces.IComment
           zope.app.container.interfaces.IObjectAddedEvent"
      handler=".comment.notify_user"
      />

  <subscriber
      for="plone.app.discussion.interfaces.IComment
           zope.app.container.interfaces.IObjectAddedEvent"
      handler=".comment.notify_moderator"
      />

</configure>
```

CHAPTER

SIX

HOWTOS

6.1 Howto extend the comment form with additional fields

This document explains how to extend the plone.app.discussion comment form with additional fields in an add-on product.

plone.app.discussion uses the plone.z3cform.fieldsets package which provides support for modifications via "extender" adapters. The idea is that a third party component can modify the fields in a form and the way that they are grouped and ordered.

Note: This howto applies only to plone.app.discussion >= 2.0.4 and >= 1.1.2. Prior versions will not store the extended fields on the comment.

See Also:

The source code of this howto can be found here: https://github.com/collective/example.commentextender/

6.1.1 Howto extend the comment form with an additional "website" field

First, create a new plone package:

```
$ paster create -t plone example.commentextender
```

Go to the main directory of the package (example.commentextender/example/commentextender) and create a new file *commentextender.py*.

This file contains the ICommentExtenderFields interface definition with a "website" field, a persistent CommentExtenderFields class to store the value of the "website" field, a CommentExtenderFactory to create the CommentExtenderFields, and a CommentExtender class to extend the default comment form with the "website" field:

```python
from persistent import Persistent

from z3c.form.field import Fields

from zope import interface
from zope import schema

from zope.annotation import factory
from zope.component import adapts
from zope.interface import Interface
from zope.publisher.interfaces.browser import IDefaultBrowserLayer
from plone.z3cform.fieldsets import extensible

from plone.app.discussion.browser.comments import CommentForm
from plone.app.discussion.comment import Comment

# Interface to define the fields we want to add to the comment form.
class ICommentExtenderFields(Interface):
    website = schema.TextLine(title=u"Website", required=False)

# Persistent class that implements the ICommentExtenderFields interface
class CommentExtenderFields(Persistent):
    interface.implements(ICommentExtenderFields)
    adapts(Comment)
    website = u""

# CommentExtenderFields factory
CommentExtenderFactory = factory(CommentExtenderFields)

# Extending the comment form with the fields defined in the
# ICommentExtenderFields interface.
class CommentExtender(extensible.FormExtender):
    adapts(Interface, IDefaultBrowserLayer, CommentForm)

    fields = Fields(ICommentExtenderFields)

    def __init__(self, context, request, form):
        self.context = context
        self.request = request
        self.form = form

    def update(self):
        # Add the fields defined in ICommentExtenderFields to the form.
        self.add(ICommentExtenderFields, prefix="")
        # Move the website field to the top of the comment form.
        self.move('website', before='text', prefix="")
```

See Also:

- See the plone.z3cform pypi page for more documentation about how to add, hide, and reorder fields: http://pypi.python.org/pypi/plone.z3cform#fieldsets-and-form-extenders

Now register the CommentExtenderFactory and CommentExtender Classes that has been created by adding the following lines to your configure.zcml:

```
<adapter
  factory=".commentextender.CommentExtenderFactory"
  provides=".commentextender.ICommentExtenderFields" />

<adapter
  factory=".commentextender.CommentExtender"
  provides="plone.z3cform.fieldsets.interfaces.IFormExtender" />
```

Create a new Plone instance, globally allow commenting, allow commenting on a content object and you will see a comment form with an additional “website” field.

Since we do not only want to store the “website” value on the comments, but also to show these values for existing comments, we have to override the comments viewlet. The easiest way to do this is to use z3c.jbot.

First, add z3c.jbot. to the setup.py of the example.commentextender package:

```
install_requires=[
    ...
    'z3c.jbot',
],
```

Next, create a new directory called “overrides” inside the example.commentextender package and register it together with z3c.jbot in your configure.zcml:

```
<configure
    ...
    xmlns:browser="http://namespaces.zope.org/browser">

  ...

  <include package="z3c.jbot" file="meta.zcml" />

  <browser:jbot
    directory="overrides" />

</configure>
```

Copy plone.app.discussion/plone/app/discussion/browser/comments.pt to the overrides directory we just created and rename comments.pt to plone.app.discussion.browser.comments.pt.

You can now add code to show the website attribute to the documentByLine:

```
<div class="documentByLine" i18n:domain="plone.app.discussion">
    ...
    <div class="commentWebsite"
         tal:condition="reply/website|nothing">
       <a href=""
          tal:attributes="href reply/website"
          tal:content="reply/website"></a>
    </div>
</div>
```

Restart your Plone instance and you will see the “website” field in the documentByLine next to the comments.

CHAPTER

SEVEN

API/INTERFACES

The conversation and replies adapters.

The conversation is responsible for storing all comments. It provides a dict-like API for accessing comments, where keys are integers and values are IComment objects. It also provides features for finding comments quickly.

The IReplies adapter provides an API for finding and manipulating the comments directly in reply to a particular comment (implemented by the CommentReplies adpater) or at the top level of the conversation (implemented by the ConversationReplies adapter).

CHAPTER

EIGHT

CHANGELOG

8.1 2.1.1 (unreleased)

- Fix control panel help text typos [jonstahl]
- Documentation about overriding the comments viewlet js added. [timo]
- Corrected location of Japanese po file. [tyam]

8.2 2.1.0 (2011-08-22)

- Avoid error when moving objects that are contentish but not annotatable. [davisagli]
- New feature: Markdown syntax added to possible comment text transforms. [timo]
- Make sure the comment brains are updated properly when the content object is renamed. [hannosch, timo]
- Make sure only comments to the content object are removed from the catalog when the content object is moved. [hannosch, timo, davisagli]
- Make sure the conversation.getComments method returns acquisition wrapped comments. [timo]
- Ukrainian translation added. [chervol]
- Remove one_state_workflow customizations. [timo]

8.3 2.0.9 (2011-07-25)

- Make sure the creator index always stores utf-8 encoded stings and not unicode. [timo]

8.4 2.0.8 (2011-07-25)

- Automatically reload batch moderation page if no comments are left. This fixes http://dev.plone.org/plone/ticket/11298. [timo]
- Use Plone's safe_encode method instead of encode() for the creator index to make sure unicode encoded strings can be indexed too. [timo]

8.5 2.0.7 (2011-07-15)

- Fix discussion control panel submit for Google Chrome. This fixes http://dev.plone.org/plone/ticket/11486.

8.6 2.0.6 (2011-07-04)

- Update comment brains in zcatalog when moving a content object with comments. This fixes http://dev.plone.org/plone/ticket/11331. [timo]
- Plone 3 specific exclusion of plone.app.uuid removed. [timo]

8.7 2.0.5 (2011-06-16)

- Simplify CSS and JS registrations. CSS will now be imported using the standard link and so can be merged, inserted after forms.css. JS will now be imported after collapsibleformfields.js. [elro]
- Enable the left-menu on the configlet, to be more consistent with all other configlets. Related to https://dev.plone.org/plone/ticket/11737 [WouterVH]
- Do not render/update the comment form in CommentViewlets if commenting is disabled, since this slows down the page rendering. This fixes http://dev.plone.org/plone/ticket/11930 [fafhrd]

8.8 2.0.4 (2011-05-28)

- Refactor/clean up the handleComment method. [timo]
- Make handleComment method store comment attributes from form extenders. This allows us to extend the comment form with external add-ons. See http://packages.python.org/plone.app.discussion/howtos/howto_extend_the_comment_form.html for details. [timo]

8.9 2.0.3 (2011-06-19)

- Updated Simplified Chinese translation [jianaijun]
- Italian translation review. [gborelli]

8.10 2.0.2 (2011-05-12)

- Moderation should be enabled only if there is a workflow set for Discussion Item. [erico_andrei]

8.11 2.0.1 (2011-04-22)

- Translations updated. German translations for notifications added. [timo]
- Add links to delete/approve a comment in the moderator notification email. [timo]

- Remove the unnecessary workflow_action parameter from the PublishComments request. [timo]
- Make sure the email settings in the control panel are disabled when commenting is disabled globally. [timo]
- Enable/disable moderator_email setting dynamically as mail settings or discussion settings change. [timo]
- Remove ImportError exceptions for Plone < 4.1 code and plone.z3cform < 0.6.0. [timo]
- Provide the comment body text in the email notification. [timo]
- Fix comment link in email notification. This fixes http://dev.plone.org/plone/ticket/11413. [timo]
- Redirect to the comment itself when notifying a user about a new comment. [timo]

8.12 2.0 (2011-04-21)

- No changes.

8.13 2.0b2 (2011-04-21)

- Added Japanese translation. [tyam]
- Move all tests from testing layer to plone.app.testing. [timo]
- Move some policy out of the conversation storage adapter into a view, specifically "enabled()". Prevents having to replace/migrate persistent objects to change policy which really only concerns the context and possibly the request, not the conversation storage. Fixes #11372. [rossp]
- Fix unindexing of comments when deleting content resulting from iterating over a BTree while modifying it. Fixes #11402. [rossp]
- Fix Missing.Value for Creator in the catalog. Fixes #11634. [rossp]
- Don't add the annotation unless a comment is actually being added. Fixes #11370. [rossp]
- Fixed i18n of the "Commenting has been disabled." message. [vincentfretin]
- Add a moderator_email setting to control where moderator notifications are sent. [davisagli]

8.14 2.0b1 (2011-04-06)

- Make discussion.css cacheable when registering it. [davisagli]
- Fix issue where GMT datetimes were converted into local timezone DateTimes during indexing. [davisagli]
- Handle timezones correctly while converting dates during the migration of legacy comments. [davisagli]
- When returning a comment's title, give preference to its title attribute if set. [davisagli]
- Use the cooked text of legacy comments when migrating. [davisagli]
- Make sure that comment text is transformed to plain text when indexing. [davisagli]
- Move logic for transforming comment text to the Comment class's getText method. Use a comment instance's mime_type attribute in preference to the global setting for the source mimetype. Use text/x-html-safe as the target mimetype to make sure the safe HTML filter is applied, in case the source is untrusted HTML. [davisagli]
- Provide a filter_callback option to the migration view, so that a custom policy for which comments get migrated can be implemented. [davisagli]

- Fixed RoleManager import to avoid deprecation warning on Zope 2.13. [davisagli]
- French translations. [thomasdesvenain]
- Fixed internationalization issues. [thomasdesvenain]
- Added Afrikaans translations [jcbrand]

8.15 2.0a3 (2011-03-02)

- Fixed test failure for the default user portrait, which changed from defaultUser.gif to defaultUser.png in Products.PlonePAS 4.0.5 [maurits]

8.16 2.0a2 (2011-02-08)

- Fixed test failure for the default user portrait, which changed from defaultUser.gif to defaultUser.png in Products.PlonePAS 4.0.5. [maurits]
- Remove "Plone 3 only" code. [timo]
- Do not monkey patch the BAD_TYPES vocabulary or plone.app.vocabularies anymore. [timo]

8.17 2.0a1 (2011-02-07)

- Split up development into two branches. The 1.x branch will be for Plone 3.x and Plone 4.0.x and the 2.x branch will be for Plone 4.1 and beyond. [timo]
- Import Owned from OFS.owner to avoid deprecation warnings. [timo]
- Disable discussion by default. [timo]
- Enable ajaxify comment deletion again ([thomasdesvenain]). This has been disabled in 1.0b12 because of problems with Plone 3. [timo]
- Remove collective.autopermission dependency that has become unnecessary in Plone 4.1. [timo]

8.18 1.0 (2011-02-07)

- Do not check for a comment review workflow when sending out a moderator email notification. This fixes http://dev.plone.org/plone/ticket/11444. [timo]
- Check if the current user has configured an e-mail address for the email notification option. This fixes http://dev.plone.org/plone/ticket/11428. [timo]

8.19 1.0RC2 (2011-01-24)

- Remove moderation_enabled setting from registry to avoid migration problems to 1.0RC1. This fixes http://dev.plone.org/plone/ticket/11419. [timo]

8.20 1.0RC1 (2011-01-22)

- Always show existing comments, even if commenting is disabled. [timo]
- Fix CSS for commenter images with a width of more than 2.5em. This fixes http://dev.plone.org/plone/ticket/11391. [timo]
- Show a 'Comments are moderated.' message next to the comment form if comments are moderated. [timo]
- Make sure plone.app.registry's ZCML is loaded, so that its import step will run when plone.app.discussion is installed. [davisagli]
- Avoid sending multiple notification emails to the same person when he has commented multiple times. [maurits]
- Move discussion action item from actionicons.xml to actions.xml to avoid deprecation warning. [timo]
- Fix cancel button on edit view when using Dexterity types. This fixes http://dev.plone.org/plone/ticket/11338. [EpeliJYU]
- Assigning the 'Reply to item' permission to the 'Authenticated' role. The old commenting system allowed 'Authenticated' users to post comments. Also, OpenID users do not possess the 'Authenticated' role. [timo]
- Make sure the handleComment method checks for the 'Reply to item' permission when adding a comment. [timo]
- Make the mail-setting warning message show up in the discussion control panel. [timo]
- Link directly to the "Discussion Item" types control panel in the moderation view. [timo]
- Show "moderate comments" link in the admin panel only if a moderation workflow is enabled for comments. [timo]
- Do not allow to change the mail settings in the discussion control panel, if there is no valid mail setup. [timo]
- Disable all commenting options in the discussion control panel if comments are disabled globally.
- Check for the 'review comments' permission instead of 'manage' to decide if the user should see a 'this comment is pending' message. [timo]
- Move "moderate comments" site action above the logout action. [timo]
- Moderator notification description updated. [timo]
- Redirect back to the discussion control panel when the discussion control panel form is submitted. [timo]
- Fix document_byline bottom margin if commenter images are disabled. [timo]
- Dynamically show the comment formatting message dependent on the text transform setting. [timo]
- Description for text transform added to the discussion control panel. [timo]
- Move the discussion control panel to the core Plone configuration. [timo]
- Always set the effective date of a comment to the same value as the creation date. [timo]
- Fix SMTP exception when an email is send to the moderator. [timo]
- Make sure comment UIDs in the catalog are always unique. This fixes http://dev.plone.org/plone/ticket/10652. [timo]
- Fix 'check all' on batch moderation page. [davisagli]
- Use safe_unicode to decode the title of the content. encode("utf-9") caused Dexterity based content types to raise a unicode decode error. This fixes http://dev.plone.org/plone/ticket/11292 [dukebody]
- Spanish translation updated. [dukebody]

- Catalan translation added. [sneridagh]
- Convert anonymous-supplied name to unicode as done for authenticated members. [ggozad]
- Catch SMTP exceptions when sending email notifications. [timo]
- Updated italian translation. [keul]

8.21 1.0b12 (2010-11-04)

- Remove AJAX comment deletion binding. This function relies on the nextUntil() selector introduced by jQuery 1.4 and therefore breaks in Plone 3 (that currently uses jQuery 1.3.2). [timo]

8.22 1.0b11 (2010-11-03)

- Fix Dutch and Czech language code and name. [timo]
- Re-add the CommentsViewlet can_manage method. This method has been removed in version 1.0b9 and added again in 1.0b11 because we don't want to change the API in beta releases. [timo]
- Declare z3c.form and zope.schema as minimum version dependencies in setup.py in case people use a different KGS. [timo]
- Add and update es and eu l10ns. [dukebody, on behalf of erral]
- Ajaxify comment deletion and approval. [thomasdesvenain]
- New feature: As a logged-in user, I can enable/disable email notification of additional comments on this content object. [timo]
- Disable the plone.app.registry check on schema elements, so no error is raised on upgrades. This fixes https://dev.plone.org/plone/ticket/11195. [timo]
- Remove the too generic id attribute of the comment form. [timo]
- Fixed handling of non-ascii member data, like fullname and email. [hannosch]

8.23 1.0b10 (2010-10-15)

- Fixed "global name 'WrongCaptchaCode' is not defined" if norobots captcha, but no other validation package is installed. [naro]
- Check if there is a 'pending' review state in the current workflow for comments instead of just checking for the 'comment_review_workflow'. This allows integrators to use a custom review workflow. This fixes http://dev.plone.org/plone/ticket/11184. [timo]
- fixed plone-it.po (improper language code ('en' instead of 'it')) [ajung]

8.24 1.0b9 (2010-10-07)

- Replace the can_manage method with a can_review method that checks the 'Review comments' permission. This fixes http://dev.plone.org/plone/ticket/11145. [timo]

- Fix moderation actions (publish, delete) in the moderation view with virtual hosts. This is a replacement for http://dev.plone.org/plone/changeset/35608. [timo]
- Do not show two "login to add comments" buttons when there are no comments yet. This fixes http://plone.org/products/plone.app.discussion/issues/12. [timo]

8.25 1.0b8 (2010-10-04)

- Apply the comment viewlet template and styles to the new title-less comments. This might require integrators to apply their custom templates and styles. [timo]
- Remove title field from the comment form and replace it with an auto-generated title ("John Doe on Welcome to Plone"). [timo]
- Fix http://dev.plone.org/plone/ticket/11098: "Comment byline shows login name, not full name" [kiorky]
- Make sure the __parent__ pointer (the conversation) of a comment is not acquisition wrapped in conversation.addComment. This fixes http://dev.plone.org/plone/ticket/11157. [timo]
- Revert r35608 since this was breaking the comment moderation bulk actions. The BulkActionsView expects the absolute path of the comments without the portal url (e.g. '/plone/doc1/++conversation++default/1285346769126020'). This fixes http://dev.plone.org/plone/ticket/11156. [timo]
- Use "(function($) { /* some code that uses $ */ })(jQuery)" instead of "$(document).ready(function(){ /* some code that uses $ */ });" to invoke jQuery code. [timo]
- Finnish translation added. [saffe]
- Italian translation updated. [keul]

8.26 1.0b7 (2010-09-15)

- Captcha plugin support for collective.z3cform.norobots (version >= 1.1) added. [saffe]
- Store dates in utc and not in local time. Display local time [do3cc]
- Fetch context for the comment view with "context = aq_inner(self.context)". [timo]
- Raise an unauthorized error when authenticated users try to post a comment on a content object that has discussion disabled. Thanks to vincentfrentin for reporting this. [timo]
- Czech translation added. [naro]
- Clean up code with PyLint. [timo]
- Make Javascripts pass JSLint validation. [timo]
- Put email notification subscribers into their own zcml file so it is easier for integrators to override them. [timo]
- Plain text and intelligent text options for comment text added to preserve basic text structure and to make links clickable. [timo]
- Rewrote all tal:condition in comments.pt. The authenticated user has the reply button and the comment form if he has the "Reply to item" permission And the discussion is currently allowed. [vincentfretin]

8.27 1.0b6 (2010-08-24)

- Fixed the case where a folder has allow_discussion=False and conversation.enabled() on a document in this folder returned False instead of True because of allow_discussion acquisition. [vincentfretin]
- Redirect to the comment form action instead of the absolute URL when a comment is posted. This fixes the accidentally triggered file upload when a comment is posted on a file content object. [timo]
- We need five:registerPackage to register the i18n folder. [vincentfretin]
- Added Traditional Chinese (zh_TW) translation. [TsungWei Hu]
- Added French translation. [vincentfretin]
- Renamed legend_add_comment to label_add_comment to have the translation from plone domain. [vincent-fretin]
- label_comment_by and label_commented_at are not in Plone 4 translation anymore, so these two messages moved to plone.app.discussions i18n domain. [vincentfretin]
- Translate "Warning" shown in @@moderate-comments in the plone domain. [vincentfretin]
- Fixed i18n markup of message_moderation_disabled. [vincentfretin]
- Catch Type errors in indexers if object can not be adapted to IDiscussion [do3cc]
- Call the CaptchaValidator even when no captcha data was submitted. This is necessary to ensure that the collective.akismet validator is called when installed. [timo]
- Spanish translation added. Thanks to Judith Sanleandro. [timo]

8.28 1.0b5 (2010-07-16)

- Use self.form instead of CommentForm for the CommentsViewlet update method so integrators don't have to override the viewlet's update method. [matous]
- Make sure the form fields in the reply form are always placed under the field labels. [timo]
- Fix CSS overflow bug that occurs with the new Plone 4.0b5 comment styles. [timo]
- Unnecessary imports and variables removed. [timo]
- Added norwegian translation. [ggozad]
- Protect against missing canonical in conversationCanonicalAdapterFactory. [hannosch]
- Documentation for Captcha plugin architecture and email notification added. See http://packages.python.org/plone.app.discussion. [timo]
- Use sphinx.plonetheme for plone.app.discussion documentation. [timo]
- Avoid deprecation warning for the Globals package. [hannosch]
- Remove the hard coded check for title and text when the comment form is submitted. This allows integrators to write schema extenders that remove the title from the comment form. [timo]
- Move captcha registration to its own captcha.zcml file. [timo]
- Akismet (http://akismet.com/) spam protection plugin (collective.akismet) support added. [timo]
- Simplify the CaptchaValidator class by dynamically adapting a view with the name of the captcha plugin (e.g. recaptcha, captcha, akismet) for the validator. [timo]
- Dutch translation added. [kcleong]

- Enable caching and merging for comments.js to save some requests. [pelle]
- Design notes for the Captcha plugin architecture added. [timo]
- Make IDiscussionLayer inherit from Interface again. Remove IDefaultPloneLayer, since Plone 4.0b1 and plone.theme 2.0b1 are out now. [timo]
- Clean up Javascript code. [timo]
- Fix encoding error in migration procedure, otherwise migration procedure breaks on joining output list in case we have there any non-ascii characters. [piv]
- plone.z3cform 0.6.0 compatibility (fix maximum recursion depth error which appears with plone.z3cform higher than 0.5.10). [piv]
- Removed moderation.js from js registry and include it only in moderation.pt as that is the only place where it is used. [ggozad]

8.29 1.0b4 (2010-04-04)

- New feature: As a moderator, I am notified when new comments require my attention. [timo]
- Sphinx-based developer documentation added. See http://packages.python.org/plone.app.discussion. [timo]
- Rename "Single State Workflow" to "Comment Single State Workflow". [timo]
- Rename 'publish comment' to 'approve comment'. This fixes #1608470. [timo]
- Show a warning in the moderation view if the moderation workflow is disabled. [timo]
- Move 'Moderate comments' link from site actions to user actions. [timo]
- Fix #662654: As an administrator, I can configure a Collection to show recent comments. Comment.Type() now correctly returns the FTI title ('Comment') [chaoflow]
- German translation updated. [juh]
- Fix #2419342: Fix untranslated published/deleted status messages. [timo]
- Remove fixed width of the actions column of the moderation view. The translated button titles can differ in size from the English titles. [timo]
- Fix #2494228: Remove comments as well when a content object is deleted. [timo]
- Fix unicode error when non-ASCII characters are typed into the name field of a comment by anonymous users. [regebro]
- Make p.a.d. work with the recent version of plone.z3cform (0.5.10) [timo]
- Make p.a.d. styles less generic. This fixes #10253. [timo]
- Added greek translation. [ggozad]
- A bug in the moderator panel meant you couldn't delete items in a virtual host, if your portal was named "plone". [regebro]

8.30 1.0b3 (2010-01-28)

- Added an i18n directory for messages in the plone domain and updated scripts to rebuild and sync it. [hannosch]
- Added an optional conversationCanonicalAdapterFactory showing how to share comments across all translations with LinguaPlone, by storing and retrieving the conversation from the canonical object. [hannosch]

- Play by the Plone 3.3+ rules and use the INavigationRoot as a base for the moderation view. [hannosch]
- Added a commentTitle CSS class to the comment titles. [hannosch]
- Update message ids to match their real text. [hannosch]
- Set CSS classes for the comment form in the updateActions method. [timo]
- Respect the allow_comments field on an object and avoid calculations if no comments should be shown. [hannosch]
- Automatically load the ZCML files of the captcha widgets if they are installed. [hannosch]
- Fixed i18n domain in GenericSetup profiles to be `plone`. Other values aren't supported for GS profiles. [hannosch]
- Provide our own copy of the default one state workflow. Not all Plone sites have this workflow installed. [hannosch]
- Register the event subscribers for the correct interfaces in Plone 3. [hannosch]
- Factored out subscriber declarations into its own ZCML file. [hannosch]
- Bugfix for #2281226: Moderation View: Comments disappear when hitting the 'Apply' button without choosing a bulk action. [timo]
- Allow to show the full text of a comment in the moderation view. [timo]
- German translation added. [timo]
- Italian translation added. [keul]

8.31 1.0b2 (2010-01-22)

- Bugfix for #2010181: The name of a commenter who commented while not logged in should not appear as a link. [timo]
- Bugfix for #2010078: Comments that await moderation are visually distinguished from published comments. [timo]
- Bugfix for #2010085: Use object_provides instead of portal_type to query the catalog for comment. [timo]
- Bugfix for #2010071: p.a.d. works with plone.z3cform 0.5.7 and plone.app.z3cform 0.4.9 now. [timo]
- Bugfix for #1513398: Show "anonymous" when name field is empty in comment form. [timo]
- Migration view: Dry run option added, abort transaction when something goes wrong during migration, be more verbose about errors. [timo]

8.32 1.0b1 (2009-12-08)

- Fix redirect after a adding a comment [timo]
- Replace yes/no widgets with check boxes in the discussion control panel [timo]
- Make comments viewlet show up in Plone 4 [timo]
- Apply Plone 4 styles to comment form [timo]
- Simplify moderation view by removing the filters [timo]

8.33 1.0a2 (2009-10-18)

- Plone 4 / Zope 2.12 support [timo]
- Comment migration script added [timo]
- Pluggable plone.z3cform comment forms [timo]
- Captcha and ReCaptcha support added [timo]

8.34 1.0a1 (2009-06-07)

- Basic commenting functionality and batch moderation. [timo]

CHAPTER

NINE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*