Plone Testing Documentation Documentation

Release 1.09

Timo Stollenwerk

June 03, 2016

Contents

Contents:

CHAPTER 1

Policy

1.1 Template

test_setup.py

```
from Products.CMFCore.utils import getToolByName
import unittest2 as unittest
from collective.mypackage.testing import \
    COLLECTIVE_MYPACKAGE_INTEGRATION_TESTING


class TestExample(unittest.TestCase):

    layer = COLLECTIVE_MYPACKAGE_INTEGRATION_TESTING

    def setUp(self):
        self.app = self.layer['app']
        self.portal = self.layer['portal']
        self.qi_tool = getToolByName(self.portal, 'portal_quickinstaller')

    def test_product_is_installed(self):
        """ Validate that our products GS profile has been run and the product
            installed
        """
        pid = 'puxam.policy'
        installed = [p['id'] for p in self.qi_tool.listInstalledProducts()]
        self.assertTrue(pid in installed,
                        'package appears not to have been installed')
```

1.2 Dependencies

Test if dependencies have been installed:

```
def test_product_is_installed(self):
    """ Validate that our products GS profile has been run and the product
        has been installed.
    """
    pid = 'collective.mailchimp'
    installed = [p['id'] for p in self.qi_tool.listInstalledProducts()]
    self.assertTrue(
```

```
        pid in installed,
        "The package '%s' appears not to have been installed." % pid')
```

setup.py

```
install_requires=[
  'setuptools',
  'collective.mailchimp',
],
```

profiles/default/metadata.xml

```
<?xml version="1.0"?>
<metadata>
  <version>1</version>
  <dependencies>
    <dependency>profile-collective.mailchimp:default</dependency>
  </dependencies>
</metadata>
```

1.3 Javascript Registration

Test if a Javascript file has been registered

```
def test_js_available(self):
    jsreg = getattr(self.portal, 'portal_javascripts')
    script_ids = jsreg.getResourceIds()
    self.assertTrue('BarackSlideshow.js' in script_ids)
```

1.4 CSS Registration

```
def test_mailchimp_css_available(self):
    cssreg = getToolByName(self.portal, "portal_css")
    stylesheets_ids = cssreg.getResourceIds()
    self.assertTrue(
        '++resource++collective.mailchimp.stylesheets/mailchimp.css'
        in stylesheets_ids
    )

def test_mailchimp_css_enabled(self):
    cssreg = getToolByName(self.portal, "portal_css")
    self.assertTrue(
        cssreg.getResource(
            '++resource++collective.mailchimp.stylesheets/mailchimp.css'
        ).getEnabled()
    )
```

Layer registered

interfaces.py

```
from zope.interface import Interface

class IJungzeelandiaTheme(Interface):
    """"""
```

browserlayer.xml

```
<layers>
  <layer
    name="jungzeelandia.theme"
    interface="jungzeelandia.theme.interfaces.IJungzeelandiaTheme"
    />
</layers>
```

test_setup.py

```
def test_barackslideshow_layer_available(self):
    from plone.browserlayer import utils
    from collective.barackslideshow.tests.layer import IBarackSlideshowLayer
    self.failUnless(IBarackSlideshowLayer in utils.registered_layers())
```

Exclude From Search

```
def makeTypeSearchable(portal, type_id, searchable):
    ptool = getToolByName(portal, 'portal_properties')
    blacklisted = list(ptool.site_properties.getProperty('types_not_searched'))
    if searchable and type_id in blacklisted:
        blacklisted.remove(type_id)
    elif not searchable and type_id not in blacklisted:
        blacklisted.append(type_id)
    ptool.site_properties.manage_changeProperties(
        types_not_searched=blacklisted)

makeTypeSearchable(portal, 'Image', searchable=False)
```

Test

```
def test_exclude_images_from_search(self):
    self.assertTrue(
        'Image' in \
        self.ptool.site_properties.getProperty("types_not_searched"))
```

Resource Directories

test_setup.py

```
def test_resources_directory(self):
    self.assertTrue(
        self.portal.restrictedTraverse(
            "++theme++dkg.contenttypes/medical-information.png"
        )
    )
```

configure.zcml

```
<plone:static
  type="theme"
  directory="resources"
  />
```

Image

```
def test_method_render_grafik(self):
  self.portal.mi.eb.invokeFactory('grafik', 'text1')
  image_file = os.path.join(os.path.dirname(__file__), u'logo.jpg')
  self.portal.mi.eb.text1.grafik = NamedBlobImage(
```

```
      data=open(image_file, 'r').read(),
      contentType='image/jpg',
      filename=u'logo.jpg'
  )
  self.assertTrue(self.portal.mi.eb.text1.render())
```

Test if code is run as test

```
if self.request['URL'] == 'http://nohost':
    # test run
```

Catalog

Catalog Indexes

```
def test_catalog_indexes(self):
    self.assertTrue('title' in self.portal.portal_catalog.indexes())
    self.assertTrue('total_comments' in self.portal.portal_catalog.indexes())
```

`catalog.xml`

```
<?xml version="1.0"?>
<object name="portal_catalog" meta_type="Plone Catalog Tool">
  <index name="autor_in" meta_type="FieldIndex">
    <indexed_attr value="autor_in" />
  </index>
</object>
```

Catalog Metadata

```
def test_catalog_metadata_installed(self):
    self.portal.invokeFactory('freitag.article.article',
                              'article')
    self.portal.article.catchword = "Foo"
    self.portal.article.reindexObject()
    self.assertTrue('catchword' in self.catalog.schema())
    result = self.catalog.searchResults(
        path='/'.join(self.portal.article.getPhysicalPath()))
    self.assertTrue(len(result), 1)
    self.assertEquals(result[0].catchword, "Foo")
```

catalog.xml

```
<?xml version="1.0"?>
<object name="portal_catalog" meta_type="Plone Catalog Tool">
 <index name="autor_in" meta_type="FieldIndex">
 <ndexed_attr value="autor_in" />
 </index>

 <column value="autor_in" />

</object>
```

Searchable index

```
def test_subjects_searchable(self):
    self.folder.invokeFactory("Document", "doc1")
    doc1 = self.folder.doc1
    doc1.setSubject([u"Python", u"Pyramid"])
    doc1.reindexObject()
```

```
    result = self.catalog.searchResults(dict(
        SearchableText = "Python"
        ))
    self.assertTrue(len(result), 1)
    self.assertTrue(result[0].title, "doc1")
```

Generic Setup Hide content type from navigation

```
def test_hide_types_form_navigation(self):
    navtree_properties = self.portal.portal_properties.navtree_properties
    self.assertTrue(navtree_properties.hasProperty('metaTypesNotToList'))
    self.assertTrue('freitag.membership.emailresetter' in
        navtree_properties.metaTypesNotToList)
    self.assertTrue('freitag.membership.member' in
        navtree_properties.metaTypesNotToList)
    self.assertTrue('freitag.membership.passwordresetter' in
        navtree_properties.metaTypesNotToList)
    self.assertTrue('freitag.membership.registrator' in
        navtree_properties.metaTypesNotToList)
```

profiles/default/propertiestool.xml

```
<?xml version="1.0"?>
<object name="portal_properties" meta_type="Plone Properties Tool">
 <object name="navtree_properties" meta_type="Plone Property Sheet">
  <property name="title">NavigationTree properties</property>
  <property name="metaTypesNotToList" type="lines">
   <element value="freitag.membership.emailresetter"/>
   <element value="freitag.membership.passwordresetter"/>
   <element value="freitag.membership.registrator"/>
  </property>
 </object>
</object>
```

Do not search content type

```
def test_types_not_searched(self):
    types_not_searched = self.portal.portal_properties\
        .site_properties.types_not_searched
    self.assertTrue('freitag.membership.emailresetter'
                    in types_not_searched)
    self.assertTrue('freitag.membership.passwordresetter'
                    in types_not_searched)
    self.assertTrue('freitag.membership.registrator'
                    in types_not_searched)
```

profiles/default/propertiestool.xml

```
<?xml version="1.0"?>
<object name="portal_properties">
  <object name="site_properties">
    <property name="types_not_searched" purge="false">
      <element value="freitag.membership.emailresetter"/>
      <element value="freitag.membership.passwordresetter"/>
      <element value="freitag.membership.registrator"/>
    </property>
  </object>
</object>
```

Portal Actions

```
def test_actions(self):
    user_actions = self.portal.portal_actions.user
    self.assertTrue("preferences" in user_actions.objectIds())
    self.assertTrue('@@my-profile' in user_actions.preferences.url_expr)
    self.assertEquals(user_actions.preferences.visible, True)
```

profiles/default/actions.xml

```
<?xml version="1.0"?>
<object name="portal_actions"
   xmlns:i18n="http://xml.zope.org/namespaces/i18n">
 <object name="user">
  <object name="preferences" meta_type="CMF Action" i18n:domain="freitag.membership">
   <property name="title" i18n:translate="">Preferences</property>
   <property name="description" i18n:translate=""></property>
   <property
      name="url_expr">string:${globals_view/navigationRootUrl}/@@my-profile</property>
   <property name="icon_expr"></property>
   <property name="available_expr">python:member is not None</property>
   <property name="permissions">
    <element value="View"/>
   </property>
   <property name="visible">True</property>
  </object>
 </object>
</object>
```

enable user folder

```
self.mtool = self.portal.portal_membership
self.assertEquals(self.mtool.memberareaCreationFlag, 1)
self.assertEquals(self.mtool.memberarea_type, 'freitag.membership.member')
self.assertEquals(self.mtool.getMembersFolder().absolute_url(),
                  'http://nohost/plone/autoren')
```

setuphandlers.py

```
membership_tool.membersfolder_id = MEMBERS_FOLDER_ID
logger.info("Members folder set up: %s\n" % MEMBERS_FOLDER_ID)

# Configure member areas
membership_tool.setMemberAreaType(MEMBER_AREA_TYPE)
logger.info("Member area type: %s\n" % MEMBER_AREA_TYPE)

membership_tool.setMemberareaCreationFlag()
logger.info("Member area creation active\n")
```

1.5 Workflow

```
def test_workflows_installed(self):
    """Make sure both comment workflows have been installed properly.
    """
    self.assertTrue('one_state_workflow' in
                    self.portal.portal_workflow.objectIds())
    self.assertTrue('comment_review_workflow' in
                    self.portal.portal_workflow.objectIds())
```

```
def test_default_workflow(self):
    """Make sure one_state_workflow is the default workflow.
    """
    self.assertEqual(('one_state_workflow',),
                     self.portal.portal_workflow.getChainForPortalType(
                         'Discussion Item'))
```

Users and Groups

```
def test_users_installed(self):
    pas = getToolByName(self.portal, 'acl_users')
    user_ids = [x['login'] for x in pas.searchUsers()]
    self.assertTrue('zell' in user_ids)
```

setuphandlers.py

```
def setupGroups(portal):
    acl_users = getToolByName(portal, 'acl_users')
    if not acl_users.searchGroups(name='Editorial'):
        gtool = getToolByName(portal, 'portal_groups')
        gtool.addGroup('Editorial', roles=[])
```

Test

```
def test_editorial_group_installed(self):
    self.assertTrue(
        'Editorial' in self.utool.source_groups.getGroupNames())
```

Roles

```
<?xml version="1.0"?>
<rolemap>
  <roles>
    <role name="Freitag Site Administrator" />
  </roles>
</rolemap>
```

test_setup.py

```
def test_freitag_site_administrator_role_installed(self):
    self.assertTrue(
        "Freitag Site Administrator" in self.portal.valid_roles())
```

Mock Mailhost

```
from zope.component import getSiteManager

from Products.MailHost.interfaces import IMailHost
from Products.CMFPlone.tests.utils import MockMailHost


class EasyNewsletterTests(unittest.TestCase):

    layer = EASYNEWSLETTER_INTEGRATION_TESTING

    def setUp(self):
        # Set up a mock mailhost
        self.portal._original_MailHost = self.portal.MailHost
        self.portal.MailHost = mailhost = MockMailHost('MailHost')
        sm = getSiteManager(context=self.portal)
```

```
        sm.unregisterUtility(provided=IMailHost)
        sm.registerUtility(mailhost, provided=IMailHost)
        # We need to fake a valid mail setup
        self.portal.email_from_address = "portal@plone.test"
        self.mailhost = self.portal.MailHost

    def test_send_email(self):
        self.assertEqual(len(self.mailhost.messages), 1)
        self.assertTrue(self.mailhost.messages[0])
        msg = str(self.mailhost.messages[0])
        self.assertTrue('To: john@plone.test' in msg)
        self.assertTrue('From: portal@plone.test' in msg)
```

CHAPTER 2

Views

2.1 Test view registration

Test if view has been properly registered:

```
def test_delete_view_registered(self):
    try:
        getMultiAdapter(
            (self.portal.mi.se.tc, self.request),
            name="delete"
        )
    except:
        self.fail("Delete view is not registered properly.")
```

2.2 Test with getMultiAdapter

Test:

```
def test_view_is_registered(self):
    # Get the view
    view = getMultiAdapter((self.portal, self.portal.REQUEST), name="create-user")
    # Put the view into the acquisition chain
    view = view.__of__(self.portal)
    # Call the view
    self.failUnless(view())
```

2.3 Test with restrictedTraverse

Test:

```
def test_view_is_registered(self):
    view = self.portal.restrictedTraverse('@@list-products')
    self.failUnless(view)
    self.assertEquals(view(), 'ListProductsView')
```

2.4 Test view with parameter

Test:

```
def test_autocomplete_tags_view_registered(self):
    self.request.set('term', 'foo')
    view = getMultiAdapter((self.portal, self.request),
                           name="autocomplete-tags")
    view = view.__of__(self.portal)
    self.failUnless(view())
```

2.5 Test with restrictedTraverse and parameter

Test:

```
def test_view_with_restrictedTraverse_and_params(self):
    view = self.context.restrictedTraverse("comment-statistics-batch")
    view = view.__of__(self.context)
    view(query, base_number * i, base_number * (i + 1) - 1)
```

2.6 Test if view is protected

Test:

```
def test_view_is_protected(self):
    from AccessControl import Unauthorized
    self.logout()
    self.assertRaises(Unauthorized,
                      self.portal.restrictedTraverse,
                      '@@deploymentmanager')
```

2.7 Test if object exists in folder

Test:

```
def test_object_in_folder(self):
    self.failIf('yoda' in self.portal.objectIds())
```

2.8 Test Redirect

Test:

```
def test_component_view(self):
    self.portal.mi.sec.invokeFactory(
        "TextComponent",
        id="tx",
        title="Text Component 1",
    )
    view = getMultiAdapter(
```

```
        (self.portal.mi.sec.tx, self.request),
        name="view"
    )
    view = view.__of__(self.portal.mi.sec)

    view()

    self.assertEqual(
        self.request.response.headers['location'],
        'http://nohost/plone/mi/sec'
    )
```

CHAPTER 3

Test View HTML Output

Test:

```
from lxml import html
output = lxml.html.fromstring(view())
self.assertEqual(len(output.xpath("/html/body/div")), 1)
```

CHAPTER 4

Troubleshooting

KeyError: 'ACTUAL_URL':

```
def setUp(self):
    self.portal = self.layer['portal']
    self.request = self.layer['request']
    setRoles(self.portal, TEST_USER_ID, ['Manager'])
    self.portal.invokeFactory('Folder', 'test-folder')
    self.folder = self.portal['test-folder']
    self.request.set('URL', self.folder.absolute_url())
    self.request.set('ACTUAL_URL', self.folder.absolute_url())

def test_view(self):
    view = self.collection.restrictedTraverse('@@RSS')
    self.assertTrue(view())
    self.assertEquals(view.request.response.status, 200)
```

4.1 ComponentLookupError

If a view can not be looked up on a particular context, Plone will raise a ComponentLookupError (because views are multi-adapters), e.g.:

```
ComponentLookupError: ((<PloneSite at /plone>, <HTTPRequest, URL=http://nohost/plone>), <InterfaceCla
```

This can be solved for instance by providing a browser layer that has been missing:

```
def setUp(self):
    self.request = self.layer['request']
    from zope.interface import directlyProvides
    directlyProvides(self.request, IJungzeelandiaContenttypes)
    ...
```

4.2 AttributeError: @@plone_portal_state

CHAPTER 5

Test View Methods

Test:

```
def test_method_sections(self):
    self.portal.mi.invokeFactory("Section", id="s1", title="Section 1")
    self.portal.mi.invokeFactory("Section", id="s2", title="Section 2")
    view = getMultiAdapter(
        (self.portal.mi, self.request),
        name="view"
    )
    view = view.__of__(self.portal.mi)

    self.assertEqual(len(view.sections()), 2)
    self.assertEqual(
        [x.title for x in view.sections()]
        [u'Section 1', u'Section 2']
    )
```

5.1 View Status Messages

Test:

```
def test_delete_comments_sets_status_message(self):
    view = getMultiAdapter(
        (self.portal.mi.se.tc, self.request),
        name="delete"
    )
    view.__of__(self.portal.mi.se)

    view()

    self.assertEqual(
        IStatusMessage(self.request).show()[0].message,
        u'Item deleted'
    )
```

View Class:

```
class DeleteComponent(BrowserView):

    def __call__(self):
        section = aq_parent(self.context)
```

```
        section.manage_delObjects([self.context.id])
        IStatusMessage(self.context.REQUEST).addStatusMessage(
            _("Item deleted"),
            type="info"
        )
        self.request.response.redirect(section.absolute_url())
```

CHAPTER 6

Viewlets

Viewlet integration test:

```
import unittest2 as unittest

from zope.component import queryMultiAdapter
from zope.viewlet.interfaces import IViewletManager

from Products.Five.browser import BrowserView as View

from jungzeelandia.theme.testing import JUNGZEELANDIA_THEME_INTEGRATION_TESTING


class TestViewletsIntegration(unittest.TestCase):

    layer = JUNGZEELANDIA_THEME_INTEGRATION_TESTING

    def setUp(self):
        self.portal = self.layer['portal']
        self.request = self.layer['request']
        from jungzeelandia.theme.interfaces import IJungzeelandiaTheme
        from zope.interface import alsoProvides
        alsoProvides(self.request, IJungzeelandiaTheme)

    def test_topnav_viewlet_is_present(self):
        view = View(self.portal, self.request)
        manager = queryMultiAdapter(
            (self.portal, self.request, view),
            IViewletManager,
            'plone.portalheader',
            default=None)
        self.failUnless(manager)
        manager.update()
        topnav_viewlet = [
            v for v in manager.viewlets \
            if v.__name__ == 'jungzeelandia.topnav']
        self.failUnlessEqual(len(topnav_viewlet), 1)
```

CHAPTER 7

Dexterity

A example for a Dexterity Content Type called IDepartment:

```
class DepartmentIntegrationTest(unittest.TestCase):

    layer = FREITAG_OVERVIEW_INTEGRATION_TESTING

    def setUp(self):
        self.portal = self.layer['portal']
        self.request = self.layer['request']
        self.request['ACTUAL_URL'] = self.portal.absolute_url()
        setRoles(self.portal, TEST_USER_ID, ['Manager'])
        self.portal.invokeFactory('Folder', 'test-folder')
        self.folder = self.portal['test-folder']
        self.portal.invokeFactory('freitag.overview.overview', id="overview1")
        self.overview = self.portal.overview1

    def test_schema(self):
        fti = queryUtility(IDexterityFTI,
                           name='freitag.overview.department')
        schema = fti.lookupSchema()
        self.assertEquals(IDepartment, schema)

    def test_fti(self):
        fti = queryUtility(IDexterityFTI,
                           name='freitag.overview.department')
        self.assertNotEquals(None, fti)

    def test_factory(self):
        fti = queryUtility(IDexterityFTI,
                           name='freitag.overview.department')
        factory = fti.factory
        new_object = createObject(factory)
        self.failUnless(IDepartment.providedBy(new_object))

    def test_adding(self):
        self.overview.invokeFactory('freitag.overview.department',
                                    'department1')
        department1 = self.overview['department1']
        self.failUnless(IDepartment.providedBy(department1))

    def test_global_adding_disallowed(self):
        self.assertRaises(ValueError,
                          self.folder.invokeFactory,
```

```
                        'freitag.overview.department',
                        'department1')
```

7.1 Test Reference to file

```
def test_method_render_pdf_file_component(self):
    # Create file to reference to
    self.portal.invokeFactory('File', 'pdf_file')
    pdf_file = os.path.join(
        os.path.dirname(__file__), 'content', u'loremipsum.pdf'
    )
    self.portal.pdf_file.file = NamedBlobFile(
        data=open(pdf_file, 'r').read(),
        contentType='application/pdf',
        filename=u'loremipsum.pdf'
    )
    from zope import component
    from zope.app.intid.interfaces import IIntIds
    intids = component.getUtility(IIntIds)
    pdf_file_id = intids.getId(self.portal.pdf_file)
    # Create file component
    self.portal.mi.eb.invokeFactory('FileComponent', 'fc')
    from z3c.relationfield import RelationValue
    self.portal.mi.eb.fc.file = RelationValue(pdf_file_id)
    self.assertTrue(self.portal.mi.eb.fc.render())
    self.assertEqual(
        self.portal.mi.eb.fc.render(),
        u'<a href="http://nohost/plone/pdf_file/@@download">'
        u'loremipsum.pdf</a>'
    )
```

7.2 Permissions

```
def test_add_medical_information_permission(self):
    self.portal.invokeFactory('MedicalInformation', 'mi1')
    sm = getSecurityManager()
    self.assertTrue(
        sm.checkPermission("dkg.addMedicalInformation", self.portal.mi1)
    )

def test_editor_has_add_permission(self):
    logout()
    setRoles(self.portal, TEST_USER_ID, ['Editor'])
    login(self.portal, TEST_USER_NAME)
    self.assertTrue(
        self.portal.invokeFactory('MedicalInformation', 'mi1'),
        "The 'Editor' role does not possess the " +
        "'dkg.addMedicalInformation' permission."
    )

def test_external_typist_has_add_permission(self):
    logout()
    setRoles(self.portal, TEST_USER_ID, ['External Typist'])
```

```
    login(self.portal, TEST_USER_NAME)
    self.assertTrue(
        self.portal.invokeFactory('MedicalInformation', 'mi1'),
        "The 'External Typist' role does not possess the " +
        "'dkg.addMedicalInformation' permission."
    )
```

CHAPTER 8

Robot Framework

8.1 Text Field

HTML

```
<input type="text" name="form.widgets.title">
```

Robot Selector:

```
Input Text  name=form.widgets.title  My Title
```

See also:

http://rtomac.github.io/robotframework-selenium2library/doc/Selenium2Library.html#Input%20Text

8.2 Text Area

HTML

```
<textarea name="form.widgets.description"></textarea>
```

Robot Selector:

```
Input Text  name=form.widgets.title  My Text
```

See also:

http://rtomac.github.io/robotframework-selenium2library/doc/Selenium2Library.html#Input%20Text

8.3 Rich Text (TinyMCE)

HTML

```
<textarea
  class="mce_editable"
  id="form.widgets.text"
  name="form.widgets.text"></textarea>
<span id="form.widgets.text_parent" class="mceEditor">
  <table id="form.widgets.text_tbl" class="mceLayout">
    <tbody>
      <tr class="mceFirst">...</td>
```

```
      <tr>
        <td class="mceIframeContainer mceFirst mceLast">
          <iframe id="form.widgets.test_ifr">
            <html>
              <head>...</head>
              <body id="content">...</body>
            </html>
          </iframe>
      </tr>
      <tr class="mceLast">...</tr>
    </tbody>
  </table>
</span>
```

Robot Selector:

```
Select frame  id=form.widgets.text_ifr
Input text  id=content  My Rich Text
Unselect Frame
```

Robot Keyword:

```
Input RichText
  [Arguments]  ${input}
  Select frame  id=form.widgets.text_ifr
  Input text  id=content  ${input}
  Unselect Frame
```

8.4 Checkbox

HTML

```
<input
  type="checkbox"
  value="Cologne"
  name="form.widgets.city:list">
```

Robot Selector:

```
Select Checkbox  xpath=//input[@name='form.widgets.city:list' and @value='Cologne']
```

8.5 Radiobox

todo

8.6 Select

todo

8.7 Tags

RF supports tags. Add a line [Tags] tag1 tag2:

*** Test cases ***

Scenario: Clicking the submit button hides it

Given i am logged in

and i am on an article

When i simulate clicking the comment submit button Then the submit button has class disabled

Scenario: Submitting a comment displays it in the page [Tags] working_on_it

Given i am logged in

and i am on an article

When i type something in the comment box and i click the comment submit button

Then the page shows the comment

You can now run only the latter test: ./bin/test -m der.freitag -t working_on_it (This is Plone-specific. See Asko's comment below.)

Note: http://keeshink.blogspot.de/2013/03/robot-framework-testing-hints.html

---

CHAPTER 9

Workflows, Roles and Permissions

9.1 Role installed

Test if a certain role (‘External Editor’) has been installed:

```
def test_external_editor_role_installed(self):
    self.assertTrue('External Editor' in self.wftool.validRoles())
```

profiles/default/rolemap.xml

```
<?xml version="1.0"?>
<rolemap>
  <roles>
    <role name="External Editor" />
  </roles>
</rolemap>
```

9.2 Roles has permission

Test if a certain role (‘External Editor’) has a certain permission (‘request review’):

```
def test_external_editor_has_request_review_permission(self):
    self.assertEqual(
        [
            x for x in self.portal.rolesOfPermission('Request review')
            if x['name'] == 'External Editor'
        ][0]['selected'],
        'SELECTED',
        'External Editor role does not possess the "Request review" '
        'permission')
```

9.3 Workflow installed

```
def test_workflows_installed(self):

    """Make sure both comment workflows have been installed properly."""
    self.assertTrue(
        'one_state_workflow' in self.portal.portal_workflow.objectIds())
```

```
    self.assertTrue(
        'comment_review_workflow' in self.portal.portal_workflow.objectIds())
```

9.4 Default workflow

```
def test_default_workflow(self):

    """Make sure one_state_workflow is the default workflow."""

    self.assertEqual(
        ('one_state_workflow',),
        self.portal.portal_workflow.getChainForPortalType(
            'Discussion Item'
        )
    )
```

z3c.form

```
def test_flag_article_as_inappropriate_with_comment(self):
    provideAdapter(adapts=(Interface, IBrowserRequest),
                   provides=Interface,
                   factory=FlagArticleAsInapproriateForm,
                   name=u"flag_as_inappropriate")
    flagAsInappropriateForm = getMultiAdapter(
        (self.article, self.request),
        name=u"flag_as_inappropriate")
    flagAsInappropriateForm.request.form = {
        'form.widgets.reason': 'This is a very naughty comment!'}
    flagAsInappropriateForm.update()
    data, errors = flagAsInappropriateForm\
        .extractData()

    self.assertEqual(len(errors), 0)
    flagAsInappropriateForm.handleApply(flagAsInappropriateForm, "foo")
    self.assertEquals('flagged',
                      self.wftool.getInfoFor(self.article, 'review_state'))
    self.assertEquals('This is a very naughty comment!',
                      self.wftool.getInfoFor(self.article, 'comments'))
    self.assertEquals(len(self.mailhost.messages), 1)
```

10.1 OLD WAY

```
from zope.interface import Interface
from zope.component import getMultiAdapter
from zope.component import createObject, queryUtility
from zope.component import provideAdapter
from zope.publisher.interfaces.browser import IBrowserRequest
from zope import interface
from zope.interface import alsoProvides
from zope.publisher.browser import TestRequest
from zope.annotation.interfaces import IAttributeAnnotatable
from z3c.form.interfaces import IFormLayer
from zope.component import getMultiAdapter
from freitag.membership.views.memberprofile import MemberProfileEditForm

...

    def test_edit_form(self):
```

```
        def make_request(form={}):
            request = TestRequest()
            request.form.update(form)
            alsoProvides(request, IFormLayer)
            alsoProvides(request, IAttributeAnnotatable)
            return request

        provideAdapter(adapts=(Interface, IBrowserRequest),
                       provides=Interface,
                       factory=MemberProfileEditForm,
                       name=u"personal-preferences")
        request = make_request(form={})
        editForm = getMultiAdapter((self.portal, request),
                                   name=u"personal-preferences")

        editForm.update()
        data, errors = editForm.extractData()
        self.assertEqual(len(errors), 0)
```

CHAPTER 11

zope.testbrowser

11.1 Debugging

```
open('/tmp/testbrowser.html', 'w').write(self.browser.contents)
import pdb; pdb.set_trace()
```

11.2 Browser Login

```
from plone.app.testing import TEST_USER_NAME
from plone.app.testing import TEST_USER_PASSWORD
self.browser.open(portal_url + '/login_form')
self.browser.getControl(name='__ac_name').value = TEST_USER_NAME
self.browser.getControl(name='__ac_password').value = TEST_USER_PASSWORD
self.browser.getControl(name='submit').click()
```

11.3 Mock Mailhost

```
from Products.CMFPlone.tests.utils import MockMailHost
from Products.MailHost.interfaces import IMailHost
from zope.component import getSiteManager

portal._original_MailHost = portal.MailHost
portal.MailHost = mailhost = MockMailHost('MailHost')
sm = getSiteManager(context=portal)
sm.unregisterUtility(provided=IMailHost)
True
sm.registerUtility(mailhost, provided=IMailHost)
portal.email_from_address = "portal@plone.test"
mailhost = portal.MailHost
```

11.4 Input

todo

11.5 Text Area

HTML

```
<textarea name="form.widgets.mytext"></textarea>
```

Test

```
self.browser.getControl(name='form.widgets.mytext').value = '<p>Lorem Ipsum</p>'
```

11.6 Radio Buttons

```
self.browser.getControl(name='form.widgets.city:list').value = ['Berlin']
```

11.7 Checkboxes

HTML

```
<input type="checkbox"
       value="selected"
       checked="checked"
       name="form.widgets.city:list">
```

Test

```
self.browser.getControl(
    name="form.widgets.city:list"
).value = ['checked']
```

11.8 Select

todo

11.9 Links

```
self.browser.getLink('Publish').click()
```

11.10 Buttons

```
self.browser.getControl('Save').click()
```

11.11 Image Upload

```
self.browser.getLink('Image').click()
self.browser.getControl(name='form.widgets.title')\
  .value = "My image"
self.browser.getControl(name='form.widgets.description')\
  .value = "This is my image."
image_path = os.path.join(os.path.dirname(__file__), "image.png")
image_ctl = self.browser.getControl(name='form.widgets.image')
image_ctl.add_file(open(image_path), 'image/png', 'image.png')
self.browser.getControl('Save').click()
```

11.12 File Upload

todo

CHAPTER 12

Miscellaneous

Login

```
self.loginAsPortalMember()
self.loginAsPortalOwner()
self.logout()
self.login("myusername")
```

Generator

```
def test_generator(self):
  self.assertEquals(sum(1 for w in self.viewlet.get_replies(workflow_actions=True)), 2)
  replies = self.viewlet.get_replies(workflow_actions=True)
  replies.next()
  replies.next()
  self.assertRaises(StopIteration, replies.next)
```

Redirect

```
def test_is_redirected(self):
  # XXX: not working!!!
  self.assertEquals(200, self.app.REQUEST.response.getStatus())
```

CHAPTER 13

Indices and tables

- genindex
- modindex
- search