

Building a Humane CMS with Plone

Naples Conference 2007

Author: Joel Burton <joel@joelburton.com>
Copyright: Copyright 2007 Joel Burton. All rights reserved.
Covering: Plone 2.5 or newer
Notice: Distribution outside of presentation prohibited.
Company: PloneBootcamps: www.plonebootcamps.com

Contents

1	Building a Humane CMS with Plone	3
1.1	Introduction	3
1.1.1	Welcome	3
1.1.2	Who Am I?	3
1.1.3	Our Inspiration	3
1.1.4	Background	4
1.1.5	Profile	4
1.1.6	Goals	4
1.2	Adding the Right Type	4
1.2.1	Specialized Factory Forms	4
1.2.2	Specialized Factory Forms	5
1.2.3	Obvious Choice for Content Types	5
1.2.4	Same Thing; Different Name	5
1.2.5	What Users Guessed	6
1.3	Content Creation	6
1.3.1	How to Structure Site	6
1.3.2	How to Structure Site (2)	6
1.3.3	How To Structure Site (3)	6
1.3.4	Content in the Right Place	7
1.3.5	Content in the Right Place (2)	7
1.3.6	Content in the Right Place (3)	7
1.3.7	Content in the Right Place (4)	7
1.3.8	Composition	7
1.3.9	Composition Details	8
1.3.10	Only Folderish Things At Top	8
1.3.11	Removing "My Folder"	8
1.4	Editing Content	8
1.4.1	Edit Forms: General	8
1.4.2	Getting Checkboxes	9
1.4.3	Explaining Multiselect	9
1.4.4	Edit Forms: Dates	9
1.4.5	Edit Forms: Keywords	9
1.4.6	Keywords vs Categories	9

1.4.7	Renaming “Categories”	10
1.4.8	Customizing Kupu	10
1.4.9	Remove Buttons from Kupu	10
1.4.10	Kupu Style Menu	10
1.4.11	Kupu Drawers and Your Types	11
1.4.12	Click to Edit	11
1.4.13	Click to Edit Caveats	11
1.4.14	Oh, the Humanity!	11
1.4.15	Dependent Items	12
1.4.16	Dependent Items	12
1.5	Simplifying Interface	12
1.5.1	Power Users	12
1.5.2	Undo	12
1.5.3	“Undo” for Other Users	12
1.5.4	Properties Tab	13
1.5.5	Effective/Expiration Date	13
1.5.6	Display Menu	13
1.5.7	Default Page	13
1.5.8	Add Menu	14
1.5.9	Actions Menu	14
1.5.10	Cut/Copy/Paste	14
1.5.11	Duplicate	14
1.5.12	Sharing Tab	14
1.6	Features	15
1.6.1	Smart Folders	15
1.6.2	Smart Folder Criteria	15
1.6.3	Smart Folder Criteria	15
1.6.4	Smart Folder Dates	15
1.6.5	Show All or Published?	16
1.6.6	RSS	16
1.6.7	Keywords	16
1.6.8	Workflow	16
1.6.9	Simple Workflow	16
1.6.10	Easy to Use Workflows	17
1.6.11	Obvious Workflow Security	17
1.6.12	Use Workflow At All?	17
1.7	CMFEditions	17
1.7.1	CMFEditions	17
1.8	Iterate	17
1.8.1	Iterate	17
1.9	Consistency of Solutions	18
1.9.1	Smart Folder Where Possible	18
1.9.2	Example: EZPortlet	18
1.9.3	Example: Filtered Smart Folders	18
1.10	Consistency of Appearance	18
1.10.1	Document Actions	18
1.10.2	base_view-based Skin	18
1.10.3	base_view Macros	19
1.10.4	base_view Macros (2)	19
1.10.5	Another base_view-based Skin	19
1.10.6	Consistency of KSS/Field Appearance	20
1.10.7	base_view Example	20
1.11	Usability Testing	20

1.11.1 Usability Testing	20
1.11.2 Testing on the Cheap	20
1.11.3 Sample Lessons	21
1.11.4 Thanks!	21

1 Building a Humane CMS with Plone

1.1 Introduction

1.1.1 Welcome

- Handouts at:
<http://plonebootcamps.com/resources/humane.pdf>
- Sample product at:
<http://plonebootcamps.com/resources/humane.zip>

1.1.2 Who Am I?

- Plone developer & consultant
- Developer & Trainer of “Plone Bootcamp”
 - 5-day, hands-on integrator-focused class
- Work mostly with large public sites
 - Many unsophisticated content editors

1.1.3 Our Inspiration

I could count twenty such
Who strive
To paint a little thing like that you smeared
...
Yet do much less--so much less!
Well, **less is more**

-- Andrea del Sarto (1486-1531)

1.1.4 Background

- Consulting engagement to test Plone usability
 - Classic videocamera & tasks setup
 - Data compiled by external UI consultant
- Formal study done for a 2.5 site
 - 3.0 information is less formal (so far!)

1.1.5 Profile

- Incidental to semi-regular content contributor
 - Won't learn much about Plone
 - Won't do often enough to remember everything
 - Ease of use over "bells and whistles"

1.1.6 Goals

- Fewer options for content placement
- Correct choices about content types
- Power user options just for power users
- Less geekish complexity
- Sane workflow/security
- Consistency in UI & methods

1.2 Adding the Right Type

1.2.1 Specialized Factory Forms

- "Make more like me"
 - On a piece of content have a link to "Make New ___ Item"
- You can create content-creation links:
`/folder/createObject?type_name=My+Type`

1.2.2 Specialized Factory Forms

- newsitem_additional portlet

```
<dl class="portlet" metal:define-macro="portlet"
  tal:define="newsfolder nocall: portal/news"
  tal:condition="python:
    checkPermission('Add portal content', newfolder)
    and checkPermission('ATContentTypes: Add News Item', newfolder)">
<dt class="portletHeader">News Item</dt>
<dd class="portletItem">
  <form tal:attributes="action string:$portal_url/news">
    <input class="standalone" type="submit" value="Add News Item" />
    <input type="hidden" name="type_name" value="News Item" />
  </form>
</dd>
</dl>
```

1.2.3 Obvious Choice for Content Types

- Give things the “right” title
 - Can rename title of type without problem
Once you re-title a content type, you should reindex the catalog.
It is possible for this not work well; add-on products can make hard-wired assumptions about the title of the content type, but this is bad behavior.
As developers, you should only write queries/code that deal with *portal_type*, not with *Type*.
 - Can re-use title of type; don’t have to be unique across site
- Don’t forget to update description
 - Used in mouseovers and factory forms

1.2.4 Same Thing; Different Name

- Users find “File” confusing
 - Make separate types:
 - * Word Document
 - * Powerpoint Document
 - * PDF Document
- This “fits peoples brains”
 - Plus, allows separate searching

1.2.5 What Users Guessed

- For Powerpoint file:
 - Event (it's a presentation)
 - News Item (it's newsworthy)
 - Image (there are graphics in it)
 - Folder (it has lots of slides)
 - Page (are slides pages?)
 - File (phew!)

1.3 Content Creation

1.3.1 How to Structure Site

- Member's folder:
 - `http://site.org/Members/joel/project-a`
 - “My Documents” feeling
 - Bad URLs
 - * What happens when ownership moves?
 - Wrong location or broken URL
 - Useful for community sites

1.3.2 How to Structure Site (2)

- Folder of like content:
 - `http://site.org/events/party`
 - Requires users to have permissions here
 - Useful when one person/department manages items of that type

1.3.3 How To Structure Site (3)

- By project / authority:
 - `http://college.edu/chemistry`
 - `http://college.edu/biology`
 - Location should follow security
 - Generally, the most useful
 - Type-specific folders within:
 - `http://college.edu/chemistry/events/party`

1.3.4 Content in the Right Place

- Restrict what's addable in folders using "restrict..."
 - Use the preferred/not-preferred
 - * Also gives help via factory form

1.3.5 Content in the Right Place (2)

- Restricting what's addable techniques
 - In the ZMI
 - In Install.py
 - In GS (using *setupVarious* technique)
 - * Cannot be done now in .xml files

1.3.6 Content in the Right Place (3)

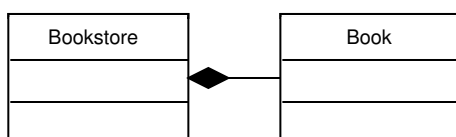
- Create specialized folder types
 - Cloning type in *portal_types*
 - * Fast and easy
 - * Can be done in GS with *types.xml*
 - Creating new Archetype based on other
 - * More flexible for future
- "Staff Directory" is clone of "Folder"

1.3.7 Content in the Right Place (4)

- Specialized folder types techniques
 - Cloning
 - * In ZMI *portal_types*
 - * GenericSetup

1.3.8 Composition

- Bookstores hold only books
- Books are held only in libraries



1.3.9 Composition Details

- Creates folder/contained setup
 - Ensures folder is folder
 - Folder holds just child
 - Child only addable in folder
- “Composition”
 - Black diamond points to parent

1.3.10 Only Folderish Things At Top

- Nothing but folders (& smart folders) at root
 - Well-organized
 - Models good behavior
- Can change in *portal_types/Portal*
 - Can be done in GS with *types.xml*

1.3.11 Removing “My Folder”

- Few sites really want user folders
 - Confusing to leave them around
 - Become a magnet for junk
- Just delete “Members” folder
 - In Plone 3, controlled by “Enable User Folders” in *Security* control panel

1.4 Editing Content

1.4.1 Edit Forms: General

- Avoid multiselect boxes
 - *Many* people don’t know how to select multiple
 - List of checkbox is much better
 - If multiselects are required, explain how to select multiple

1.4.2 Getting Checkboxes

- In your Archetype UML, for multiselect field:

```
widget:format = checkbox    # default is "select"
```

1.4.3 Explaining Multiselect

- In your Archetype UML:

```
widget:description =  
    Hold control CONTROL (Mac users: COMMAND) to select  
    multiple items
```

1.4.4 Edit Forms: Dates

- People *hate* the drop-down menus
 - Slower, limited range of dates
 - Replace with add-on product, *DateBox*
 - * Allows textual entry in any form
 - * Plus ideas like “yesterday”
 - * Immediately turns into canonical format

1.4.5 Edit Forms: Keywords

- Most useful organizational tool
 - But underused
- Move keywords onto edit form
 - Call “tags”
I’ve found that the term “keyword” seems scary to people, but calling this “tags” seems more friendly.
 - Use `AddRemoveWidget`

1.4.6 Keywords vs Categories

- Early to tell for sure, but ...
 - People find “categories” confusing
 - * Suggests a structured taxonomy
 - * With a single place
 - “Tags” is probably still best

1.4.7 Renaming “Categories”

- Patch AT’s “subject” field
 - See *AAASTuff* product
- Or handle using i18n tools
 - msgid is “label_categories”

1.4.8 Customizing Kupu

- Can customize with XML
- But easier to do in CSS, and easier for upgrading:


```
#kupu-subscript-button { display: none; }
```
- In Plone 3, tool allows customization

1.4.9 Remove Buttons from Kupu

- You probably don’t want
 - superscript, subscript
 - definition lists
 - full-screen button
- You might not want
 - tables
 - left, center, right justify

1.4.10 Kupu Style Menu

- Add classes to kupu menu
 - Single best thing you can do for users
- Can add classes per type-of-content:

```
TextField('text',
  widget=RichWidget(
    parastyles=(
      ('div|contactInformation', 'Contact Information'),
      ('div|notesToEditors', 'Notes to editors'), ),
  ),
),
```

1.4.11 Kupu Drawers and Your Types

- Add new content types to kupu “resource types”
 - Otherwise, can’t link to/browse in Kupu
 - Folderish things: add to “collection”
 - Images/movies with `tag()`: add to “mediaobjects”
 - Everything: add to “linkable”
- In Plone 3, can happen automatically

1.4.12 Click to Edit

- View screens for content allow click-to-edit
 - Excellent for power-users doing fast edits
 - Sometimes confusing for novice users
 - * Easy to trigger unexpected errors
 - Powered by KSS and async JavaScript (AJAX)

1.4.13 Click to Edit Caveats

- Only works with
 - Uncustomized Archetype views
 - “Plone 3-ified” view templates
 - * Since practically no one did things to anticipate this
- Not (yet) supported for all AT field types
 - Missing for *float*, *lines*, and others
 - Not supported for add-on field types

1.4.14 Oh, the Humanity!

- Early research shows confusing for end-users
 - May get better as it gets refined
 - Can turn off by turning off *at.kss* in *portal_kss*
 - * This loses live validation on edit form, though
 - * Refactoring may allow this to be easier in future

1.4.15 Dependent Items

- Confusing to users that image isn't "in" document
 - Different workflow, separate title, etc.
- *RichDocument* keeps images/files inside document

1.4.16 Dependent Items

- Can also use kupu's *ReffieldText* sets references for images, links
 - Can caption images automatically
 - With references, can move workflow together, etc.

1.5 Simplifying Interface

1.5.1 Power Users

- The people who everyone turns to
- The people who think they are
- Make a role for it!
 - In GS, using *rolemap.xml*

1.5.2 Undo

- Can be complex to understand
 - Seeing all objects listed
 - Dependencies of undoing
- Too complex for most users
 - And most turn to power users anyway
- Control with "List undoable changes" permission

1.5.3 "Undo" for Other Users

- May find it helpful to keep "undo" in user bar
 - People are used to it
 - Could point to History tab of item
 - * Or a template pointing toward Power Users

1.5.4 Properties Tab

- “Metadata” versus “data” is weak distinction
- Often entirely ignored
- Move useful things onto edit tab
 - Leave esoteric stuff for power users
- Handled in Plone 3

1.5.5 Effective/Expiration Date

- Called “Publishing Date” in Plone 3
 - Confuses people that it’s related to “publish” transition
 - Suggest changing back to “effective date”
- In general, these confuse most people
 - And are not commonly needed
- Recommend: protect with `write_permission` tied to `PowerUser`

1.5.6 Display Menu

- Remove skins that won’t be used
 - In many cases, only one makes sense
 - Change in `portal_types` for your type
- Control with “Modify view template” permission

1.5.7 Default Page

- Tricky concept for many users
 - Often, having a body field for folderish item is easier
- Can turn off with:

```
def canSetDefaultPage(self):  
    """Disallow setting default page."""  
    return False
```

1.5.8 Add Menu

- “Restrict” may be not useful
 - Especially for a specialized folder type
- Control with “Modify constrain types” permission
- Can turn off with:

```
def canSetConstrainTypes(self):  
    """Disallow 'restrict' choice"""  
    return False
```

1.5.9 Actions Menu

- Rename allows people to break links
 - Which they often don’t understand
 - Make for power users only
 - * Don’t change permission--some products assume rename capabilities
 - * Change in *portal_actions*.

1.5.10 Cut/Copy/Paste

- People can find confusing
 - Allows them to move content
 - Non-obvious why you can’t paste into certain places
- Make for power users only

1.5.11 Duplicate

- People often expect “copy” to *make a copy*
 - Not copy to clipboard
- “Duplicate” action is very helpful
 - Implemented in *humane* product

1.5.12 Sharing Tab

- Useful feature, but complex form
 - Simplified in Plone 3
- Consider making for power users
- Or, offering simplified sharing tab
 - Can only share Owner role
 - * Or, Editor/Contributor/Reader, etc.

1.6 Features

1.6.1 Smart Folders

- Too tricky for ordinary users to make
 - Good use for “Power User” role
 - Create sample smart folders, users can duplicate & modify
- Smart folders to find forgotten content (ie, never submitted/published)

1.6.2 Smart Folder Criteria

- *and / or* criteria always confuses people
 - Assume this is *and/or across* predicates
 - * It’s any/all of choices for predicate
- Suggest: change to *any / all*
 - Can change *CompareOperators* in *ATContentType/criteria/selection*
 - * Or handle via i18n

1.6.3 Smart Folder Criteria

- Some changes make *no sense* with *all* choice
 - A single piece of content cannot have multiple
 - * *portal_type*, *workflow*, etc.
- Fixing would require adding logic to Smart Folders for this
 - Best handled now with training

1.6.4 Smart Folder Dates

- Relative dates have 3-set choices
 - “More than” “1 week” “in the past”
 - * Is that older than one week? Or newer?
 - Many people get wrong!
- Change to “ago” in *ATContentTypes/criteria/dates/RangeOperator*
 - Or handle in i18n

1.6.5 Show All or Published?

- Show all? Or just published?
- Either confuses people
 - “Ack! My private stuff is showing!”
 - “Ack! Where’s my personal stuff?”
- Pick your poison
 - Showing non-published is usually easier
 - * Especially with color-coded workflow hints

1.6.6 RSS

- Underused by normal users
 - Few have dedicated RSS readers
- In-browser RSS readers can simplify
 - “Sage” for Firefox
- Dashboards in Plone 3 help, too

1.6.7 Keywords

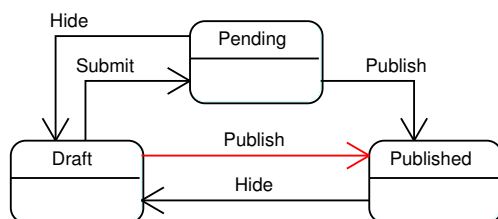
- Stock sample keywords
 - Make private document with starter keywords
- Use PloneKeywordManager to clean up keyword space

1.6.8 Workflow

- Simpler is usually better
 - The more steps, the less attention

1.6.9 Simple Workflow

- Simple and trusting



- Review portal content given out freely

1.6.10 Easy to Use Workflows

- Transitions for *all* stages
 - Can publish from hidden, hide from published, etc.
 - Fixed in recent Plone

1.6.11 Obvious Workflow Security

- Visible things in private folders are visible
 - Our worst security/UI choice
 - 90% of my students don't expect
- Fix or document thoroughly!

1.6.12 Use Workflow At All?

- For intranet/planning sites, may be superfluous
- Can change workflow to single state
 - But still shows up in UI
 - Can hide menu with:

```
#statusMenu {display: none; }
```
 - Or can modify template

1.7 CMFEditions

1.7.1 CMFEditions

- Initial studying shows good usability
 - “Revision 0” has confused some people
 - * “Revision 1” might be better
- Ensure people understand it's not time travel

1.8 Iterate

1.8.1 Iterate

- Initial studying shows good usability
 - But staging itself is more than most sites need

1.9 Consistency of Solutions

1.9.1 Smart Folder Where Possible

- Much integration can be done with Smart Folder
 - “Collections” in 3.0
- Provides consistent way to express queries
 - People can learn once and re-use

1.9.2 Example: EZPortlet

- Proof-of-concept portlet
 - Show results of any smart folder

1.9.3 Example: Filtered Smart Folders

- Smart folders with trimming or extending

1.10 Consistency of Appearance

1.10.1 Document Actions

- Print this page, email to friend, etc.
 - Moved from top -> bottom in 2.5 -> 3.0
 - Switch from icons to text
- 2.5 products/existing skins will still be at top
 - Consistency is very important

1.10.2 base_view-based Skin

- **movie_view** Page Template:

```
<div metal:define-macro="body">
  <p>Director:
    <i tal:content="context/director">[Dir]</i>
  </p>
  <p>Rating:
    <b tal:content="context/rating">[Rating]</b>
  </p>
</div>
```

1.10.3 base_view Macros

It's a Wonderful Life



by [admin](#) — last modified 2007-03-21 07:04 Copyright 1935 by MGM.

Director:

Frank Capra

Rating:

5

[Capra Biography](#) — by [admin](#) — last modified 2007-03-21 07:03

Details of the director's life.

[Plot Description](#) — by [admin](#) — last modified 2007-03-21 07:03

Synopsis of plot for new viewers.

[Footer goes here]

header**body****folderlisting****footer**

1.10.4 base_view Macros (2)

- *header*
- *body*
- *footer*
 - Normally empty
- *folderlisting*
 - Listing of child objects in folders
 - Override with empty to prevent this

1.10.5 Another base_view-based Skin

- **movie_view** Page Template:

```
<div metal:define-macro="header">
  <h1 tal:content="string:
    Movie Info: ${context/Title}">[Title]</h1>
</div>

<div metal:define-macro="footer">
  <p>Movie data is from IMDB.</p>
</div>
```

This example is too simplified: our header macro replaces the entire header, which should contain the styles to show the print-this-page button and send-to-friend button.

A realistic header replacement should contain:

```
<h1 tal:content="context/title_or_id"
  class="documentFirstHeading">
  Title or id
</h1>
```

```
<div metal:use-macro="
  context/document_actions/macros/document_actions">
  Document actions (print, sendto etc)
</div>
```

1.10.6 Consistency of KSS/Field Appearance

- Can output view widget for field
 - As opposed to using accessor directly
- Will allow inline editing in Plone 3
 - And other advantages of widget
 - * LinkField might show URL as link

1.10.7 base_view Example

- *yourtype_view.pt*:

```
<div metal:define-macro="body">
  <div metal:use-macro="python:
    context.widget('myfield')" />
</div>
```

1.11 Usability Testing

1.11.1 Usability Testing

- Science of observation
 - Not of personal theory :)
- No substitute for observing unprompted users

1.11.2 Testing on the Cheap

- Don't need a camera / mirrors / consultant
- Do need a script of tasks
 - Keep them the same for all testers
- Do need testers unfamiliar with Plone
- Ask users to explain what they're thinking

1.11.3 Sample Lessons

- User being asked to add a Powerpoint file to a 2.5 portal:

Well, I guess I should go into my folder. [Clicks]
I'll look in contents so I can add it
[Missing quicker add menu]. Powerpoint are
presentations, so I'll add it as an Event ...

1.11.4 Thanks!

- I'd love your ideas and feedback:

joel@joelburton.com

- Handouts online at:

<http://plonebootcamps.com/resources>