Archetypes: Customizing Plone in 60 Seconds
Overview

- What is Plone?
- What is Archetypes?
- Making an Archetype
  - Customising it
  - Customising Plone
- Conclusion
- ...I just thought I'd show you some stuff I think is cool
Plone

• You all attended Joel Burton's tutorial?
• Open Source Content Management System
  - Based up Zope
  - Written in Python
  - Focuses on:
    • Usability
    • Accessibility
    • Internationalization
Content types

• Plone separates “stuff” out into the following:
  - Content types
    • Data added and edited by the user
    • Workflowed and catalogued
  - Templates
    • Page Templates, scripts, images, minor logic, mostly presentation
  - Tools
    • Services that perform certain things
Example Content types

- Almost anything a user wants to edit:
  - Web pages
  - Multimedia files
  - Word documents
  - People
  - Car parts
  - Laboratory samples... you name it
So...

- Users go to a Plone site and add in content types but...
  - How do I edit the content types?
    - The fields that they have
    - The forms users enter
    - The way content types are rendered
    - The answer ... Archetypes
What is Archetypes?

• A way of easily creating and customizing a content type. It does this by...
  – Maintaining a Schema for the content type
  – Maintaining a Widget set of all the widgets
  – Creating validators
  – Creating a standard way of handling content types
Danger, Will Robinson

- Archetype is a *developer* tool
  - It's written by developers for developers
  - So once you get to know it, it's fast and easy to use however
    - Error checking of code is questionable
    - Upgrades may or may not work
    - You'll probably have to be hacking at some point
  - Not to put anyone off, but... this isn't at the level you can give to a secretary to modify
Archetypes can be found

• In your Plone 2 distribution
• At SourceForge
  – http://sf.net/projects/archetypes
• Loads of sample products
  – In the Archetypes cvs and in the Collective
  – http://sf.net/projects/collective
• In the future *all default* content types will be Archetypes
Schema

• At the core of Archetypes is...
  – The Schema
    • Each Schema contains a list of fields
      – There can be more than one field for a schema
    • Each field contains a widget
      – There is only one widget for each field
  – There’s nothing complicated about a schema, its just the definition for that content type
Field

- A field is an attribute or piece of information on a content
  - Example fields could be:
    - Title, Description, URL, Phone number
  - The information such as validation and so on is set on the field
  - Each field must have at least one property id which must be:
    - A valid Python variable name
    - Start with a lower case first letter
Widget

• A widget is a representation of a field
  - Example widgets could be:
    • String, Text, Calendar
  - The widget has a HTML representation that shows the field in a certain manner
  - Information that is shown in HTML is added to the widget
An example widget

• Here's an example widget:

    TextAreaWidget(
        label='Proposal',
        description='Give an overview'),
    )

- Our proposal widget has
  • A label
  • And a description
  • Because it's a TextAreaWidget it's going to be shown as a HTML text area
An example field

• Here's an example field:

```python
TextField('proposal',
    widget = TextAreaWidget(
        label='Proposal',
        description='Give an overview'),
)
```

– Our proposal field has
  • Our proposal widget
  • An id “proposal”
  • And its going to be shown using the TextAreaWidget
An example Schema

• Here's an example schema:

```
Schema((TextField('proposal',
    widget = TextAreaWidget(
        label='Proposal',
        description='Give an overview'),
    ),)),
```

− Our schema now has

  • Our proposal field with its
    − Proposal widget
  • Note: The brackets get confusing, its - a tuple of fields which contain widgets, passed as an argument to the schema
  • Tip: Use an editor that does bracket matching
So let's give this a spin

- Demo
- Please forgive my Plone... it's likely to be dev. Checkout
- Unlike other CMS's:
  - Plone serves content to the users
  - Once you've built this, you are done...
Plone Product

- This whole schema
  - is placed inside a class which
  - is placed inside a product

- So the product contains the whole registration and setup
  - It's kind of boring and boilerplate, so let's not go there
  - See Collective and Archetypes CVS repositories for that
So what's with this 60 second thing

• Lets make:
  – Proposal required
  – Change the description

• To do this I'll
  – Go and change the code
  – Restart Zope
  – Add in a new object and...
Changing all the old object

• Old objects maintain a copy of their old schema
  – So you need to go and update them
  – In the ZMI the archetype tool > Update Schema page will go through and change old objects
  – Note the warning on the page
  – This does not:
    • Enforce things like required
Other Widgets and Fields

• There's a whole bunch:
  - There's doc's on this at:
    - http://plone.org/documentation/archetypes
    - And in the book
  - Each widget has almost arbitrary arguments that it will accept eg:
    - TextWidget will accept rows and cols since these directly relate to HTML attributes
  - Let's add an Image Widget in...
Base schema

• You'll probably note that our OSCON object has more fields than we specified:
  – All objects in Plone must have a title and id
  – This is defined in a Schema called BaseSchema and then we add them together:

    \[
    \text{schema} = \text{BaseSchema} + \text{Schema}(\ldots)
    \]
Custom Widgets and Fields

• You can add in custom widget and field
  – And this is pretty straightforward
  – Let's make an widget for an email address that shows a clickable email link
    • So we'll need to write a Page Template for the view
    • We'll need to register it in Plone (we'll skip this bit)
Email

• Here's the Page Template snippet:

  • Note: there are three macros...
  • Let's take a look

  <div metal:define-macro="edit">
    <div metal:use-macro="here/widgets/string/macros/edit" />
  </div>

  <div metal:define-macro="search">
    <div metal:use-macro="here/widgets/string/macros/search" />
  </div>

  <div class="field" metal:define-macro="view">
    <a href="#" tal:attributes="href string:mailto:${accessor}"
       tal:content="accessor">email</a>
  </div>
View and Edit

• We've seen the edit page which is set up for us, but the view page is well ugly...
  – It's designed to be overridden
  – You override this in a Page Template
Editing TTW sucks

- So we want something fancier
  - EpozWidget will make an Epoz TTW editor for you
    - You must have Epoz installed
    - The user must have Epoz set as the editor of choice
    - Gotcha: for the field set...
      - default_output_type='text/html',
External Editing sucks less

- So a user wants to write and edit in Word (yuk)
  - Archetypes integrates with External Editor, this spawns a local process to edit the content
  - When the file is saved, the copy is uploaded
  - Needs a client tool
  - Then a transformation is done
- Let's see the demo first
Portal transforms allow you to...

• Transform content, its part of Archetypes
  - For example, want to transform:
    • PDF to HTML
    • Word to HTML blah...
  - `portal_transforms` automates this through *external* tools
  - Eg: pdf2html, wvWare, win32
  - You need to set up the external tools
Example: Word to HTML

• Has to be RichWidget and..
  - Set the output type to the destination of the transport (HTML)
  - Allowable content type set to the source (Word)
  - This matches the portal_transforms tool
    
    ```python
    default_output_type='text/html',
    allowable_content_types=('application/msword',),
    ```
  - One more change
Primary Field

- External Editor is going to send a blob of data...
  - How do we know what to change
  - We define a primary field
  - Then we set a marshaller to point to the primary field

```python
from Products.Archetypes.Marshall import PrimaryFieldMarshaller

marshall=PrimaryFieldMarshaller()
```
Conclusion

● Archetypes is cool and fun
  – Definitely productive
  – ArchGenXML provides UML to Archetypes conversion
  – ... but can be a bit of a “black” box

● Any questions?
● andy@enfoldsystems.com