AtoM Feature Development

An intro on how to create and contribute

Steve Breker, May 2017
AtoM Camp SJC


isad-ead-crosswalk-2.3.xml
scripts_transform_csv.py
layout_3col.php
<!DOCTYPE html>
<html lang="<?php echo $sf_user->getCulture() ?>" dir="<?php echo sfCultureInfo::getInstance($sf_user->getCulture())->direction ?>">
<head>
<?php include_http_metas() ?>
<?php include_metas() ?>
<?php include_title() ?>
<link rel="shortcut icon" href="<?php echo public_path('favicon.ico') ?>"/>
<?php include_stylesheets() ?>
<?php include_component_slot('css') ?>
<?php include_javascripts() ?>
</head>
<body class="yui-skin-sam
<?php echo get_partial('header') ?>
<?php include_slot('pre') ?>

<?php echo get_partial('alerts') ?>

<?php include_slot('sidebar') ?>

Feature Development Process

Feature development process overview

1. Feature idea
2. Technical classification
3. Development preparation
4. Creating a feature
5. Contributing a feature

1. Feature idea

- New archival standard?
- New theme?
- New way to bring data in or out?
- ???

2. Technical classification

- Plugins
- CLI tasks
- Background jobs
- Core features

2. Technical classification

- **Plugins** <- probably the most common
- CLI tasks
- Background jobs
- Core features

3. Preparation: design

- Think of broad use-cases
- Follow known open standards whenever relevant
- Implement as simply as possible
- Change as little as possible to get the functionality you want
- Avoid breaking backwards compatibility
- Run your ideas by the community

3. Preparation: technical

- Read up on Symfony 1.4
- Explore AtoM's codebase
- Ask questions in the AtoM user forum

4. Developing your feature

- Fork AtoM on GitHub to have your own repository to work with
- Follow AtoM coding standards:
  `https://wiki.accesstomemory.org/Development/Coding_standard`
- Review our wiki page on contributing code:
  `https://wiki.accesstomemory.org/Development/Contribute_code`
- Make sure any third-party code libraries you add are AGPL v3 compatible

5. Contributing your feature

1. Submit a pull request on GitHub with your work
2. Respond to feedback in the pull request until the pull request's approved:

   `https://wiki.accesstomemory.org/Development/Code_review`
3. Fill in a Contributor's agreement:

   `https://wiki.accesstomemory.org/Development/Contribute_code`


IT'S PARTY TIME

Symfony 1.4

Symfony 1.4 overview

- Symfony 1.4 is the web application framework AtoM is built with
- Symfony follows the MVC (model/view/controller) pattern
- Models represent types of data
- Views represent how data is rendered
- Controllers represent logic that determines what data ends up being rendered by the view

Symfony models

- AtoM's Symfony is using an ORM called Propel
- To define models, data schemas are defined using YAML files
- A CLI tool is run that uses these YAML files as a guideline to generate PHP code to that defines model classes
- These classes are referred to as "base" models and aren't supposed to be manually altered
- Model characteristics can be added or overriden by creating child classes that extend the base models
- Migrations handle changes between schema versions

Symfony controllers

- There are two main types of controllers: actions and components
- Actions define the behaviour of pages
- Components define logic shared between actions
- Both types of controllers are represented by classes
- Actions extend `sfAction` (or a child class)
- Components extend `sfComponent` (or a child class)
- If actions or components are part of a plugin they are given a class name that includes the plugin name
- Example: `sfRadPluginEditAction`

Symfony views

- There are two main types of views: page templates and partials
- Page templates define how an action's data is rendered
- Partials have multiple uses:
  - They can be used to define how a component's data is rendered
  - They can be used by page templates to render repeating data
  - They can be used by page templates to encapsulate a complex part of a page
- The `use_helper` function can be used in page templates to include functions intended to be used within templates (for rendering dates and URLs, etc.)

Symfony routing

- AtoM's routing is largely defined in `apps/qubit/config/routing.yml`
- Plugins can dynamically add routes as well
- Example: the arRestApiPlugin's configuration class adds routes

Symfony debugging/development tools

- The clear cache CLI task is useful during development:
  `php symfony cc`

- Also useful is the CLI task to purge all user-created data:
  php symfony tools:purge

- See Steve Breker's presentation, slide 4, for how to enable debug mode

Developing Plugins

Plugin Development Overview

- Plugins can be used to implement optional features as mentioned earlier (support for individual archival standards, new themes, etc.)
- Plugins are also used in AtoM to escapsulate functionality (the `arElasticSearchPlugin` plugin for example)
- Plugins can also be used to add new classes that other plugins can share

Plugin directory structure

- Plugins have four optional subdirectories: `config`, `lib`, `modules`, and `web`
- `config` is where plugin-related configuration files can be put
- `lib` is where plugin code libraries, such as plugin-related classes, can be put
- `modules` is where module-related code can be put
- `web` is where plugin-related web assets are put

Plugin configuration files

- Plugin configuration files are named using the convention: `<plugin name>Configuration.class.php`
- An example: `sfRadPluginConfiguration.class.php`
- These files define a class named `<plugin name>Configuration` that inherits from Symfony's `sfPluginConfiguration` class
- These files specify the plugin's name, version, etc.
- These files also can contain plugin initialization logic, etc.

Plugin library files

- Plugin library files are named with the plugin name included in the filename
- An example: `plugins/arOaiPlugin/lib/arOaiPluginComponent.class.php`
- This is merely a convention, however
- Any class that is put in the plugin's lib directory will be auto-loaded by AtoM once the cache is cleared
- The `sfHistoryPlugin` plugin is an example of a plugin whose sole functionality is encapsulated in a class in the plugin's `lib` directory

Plugin modules

- Modules are an organizational unit used in Symfony to encapsulate a bunch of files related to a group of application web pages

```
sfFacebookPlugin
    modules
        photos
            actions
                indexAction.class.php
                editAction.class.php
            templates
                indexSuccess.php
                editSuccess.php
```

Enabling/disabling Plugins

- Plugins that consist entirely of plugin code library, put in `lib`, don't need to be explicitly enabled
- Plugins must contain a configuration file in their `config` subdirectory in order to be enabled by an admin
- Enable or disable plugins using the `sfPluginAdminPlugin/plugins` page

```
    [InformationObject] Tweedsmuir Village History: For Home and Country Chelmsford Women's Institute Scrapbook - 1 inser
    [InformationObject] Tweedsmuir Village History: For Home and Country Chelmsford Women's Institute Scrapbook - 2 inser
    [InformationObject] Papers inserted (46.43s) (440/476)
    [InformationObject] Plants inserted (46.5s) (441/476)
    [InformationObject] The African Violet inserted (46.57s) (442/476)
    [InformationObject] Azilda inserted (46.62s) (443/476)
    [InformationObject] Azilda, centre rural progressif inserted (46.69s) (444/476)
    [InformationObject] Typed Transcript - Azilda, centre rural progressif inserted (46.73s) (445/476)
    [InformationObject] Chelmsford Pioneers and Churches inserted (46.79s) (446/476)
    [InformationObject] The First Pioneers and The History of Chelmsford inserted (46.86s) (447/476)
    [InformationObject] The First Pioneers & The History of Chelmsford inserted (46.93s) (448/476)
    [InformationObject] Chelmsford Schools inserted (46.98s) (449/476)
    [InformationObject] The Schools in Our Community inserted (47.04s) (450/476)
    [InformationObject] The School History of Our Community inserted (47.1s) (451/476)
    [InformationObject] School History of Our Community inserted (47.17s) (452/476)
    [InformationObject] Chelmsford Telephones inserted (47.3s) (453/476)
    [InformationObject] Historical Research inserted (47.38s) (454/476)
    [InformationObject] Chelmsford Growth and Development inserted (47.46s) (455/476)
    [InformationObject] Chelmsford Growth and Development inserted (47.54s) (456/476)
    [InformationObject] Jack Bush fonds inserted (47.63s) (457/476)
    [InformationObject] Gallery 44 Centre for Contemporary Photography fonds inserted (47.72s) (458/476)
    [InformationObject] Florence Vale fonds inserted (47.8s) (459/476)
    [InformationObject] A.H. Howard fonds inserted (47.87s) (460/476)
    [InformationObject] Shieky Brownstone - Jack Chambers collection inserted (47.94s) (461/476)
    [InformationObject] David Brown Milne fonds inserted (48s) (462/476)
    [InformationObject] J.A. Wainwri                Robert Markle                ction inserted (48.07s) (463/476)
    [InformationObject] Will Munro c                ection inserted (48.14s) (464/476)
    [InformationObject] Allon Peebles personal p                e inserted (48.44s) (470/476)
    [InformationObject] Reports and Publication                rted (48.49s) (471/476)
    [InformationObject] Photographs inserted (48.53s) (471/476)
    [InformationObject] Photographs, Souvenir of New Westminster B.C. [Album] inserted (48.58s) (472/476)
    [InformationObject] 1-23 Original envelopes for photographs 153.1/1 through 153.1/13 with attached notes inserted (48
    [InformationObject] Multimedia inserted (48.66s) (474/476)
    [InformationObject] Condolence Letters and Budget Book inserted (48.69s) (475/476)
    [InformationObject] A. Richard King fonds inserted (48.74s) (476/476)
Index populated with 4075 documents in 48.74 seconds.
[~/atom] (qa/2.4.x)
vagrant$ php symfony help search:populate
Usage:
 symfony search:populate [--application[="..."]] [--env="..."] [--exclude-types[

Options:
 --application    The application name (default: qubit)
 --env            The environment (default: cli)
 --exclude-types  Exclude document type(s) (command-s                indexing
 --show-types     Show available document type(s), t                ded, before

Description:
 The search:populate task empties, populates, and o
 in the current project. It may take quite a while t

 To exclude a document type, use the --exclude-types o                ample:

   php symfony search:populate --exclude-types="term,actor"

 To see a list of available document types that can be excluded use the --show-t

[~/atom] (qa/2.4.x)
vagrant$
```

Developing Tasks

Task Development Overview

- Tasks are primarily used by advanced users and system administrators
- Tasks add command-line accessible features
- Task examples: cache clearing, bulk import/export, adding administrators, etc.
- Task code can be found in `lib/tasks`

Developing Background Jobs

Background Job Development Overview

- Jobs are used to perform "heavy lifting" in the background of an AtoM instance
- For example, if a user requests a CSV export of all descriptions the user will be informed that the export has started, but won't have to wait for the export to complete to get a web page response
- Users can visit a webpage to see the status of their jobs, whether in-progress, completed, or failed
- A job is analogous to a pizza delivery order: the doorbell will ring when the pizza arrives and when in doubt you can call the pizzaria to enquire about whether the pizza's done or not

Questions?

info@artefactual.com