Using the Allura Platform to Create Your Own Forge

Rick Copeland, Software Architect
SourceForge.net

Overview

In late 2009 SourceForge embarked on a plan to "reboot" our developer tools on an open platform including Python, MongoDB, RabbitMQ, and SOLR. The result was the Allura platform, and was released under the Apache License in February 2011.

Our main goal for the Allura project was to build a platform that would provide commonly needed services for various developer tools (Wiki, Tracker, Repository, Forums, etc.) that are installed in various projects. Project admins should be able to customize the set of tools installed for their project and have fine-grained control over access rights in their project.

The resulting project, self-hosted at http://sf.net/p/allura, provides users wishing to build their own forge with project administration, full-text search, ticket tracking, wiki, forums, repository management (Git, Subversion, and Mercurial included), along with a solid platform for building your own custom tools that can be installed alongside the native tools provided by Allura.

URL Structure

Standard (user-space) URLs

`/p/allura/git/ci/36f02a9f3e7b31282bb74732481af7b0aa425441/tree/`

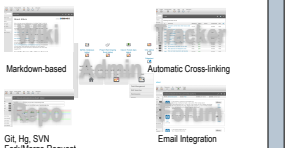
- /p – Neighborhood short name
- /allura – Project short name (can include subprojects)
- /git – Tool mount point
- /ci/…tree/ - Tool-controlled custom URL space

Admin URLs

`/p/allura/admin/git/extensions`

- /p – Neighborhood short name
- /allura – Project short name (can include subprojects)
- /admin – Admin tool mount point
- /git – Tool being administered mount point
- /extensions - Tool-controlled custom URL space

Built-in Tools

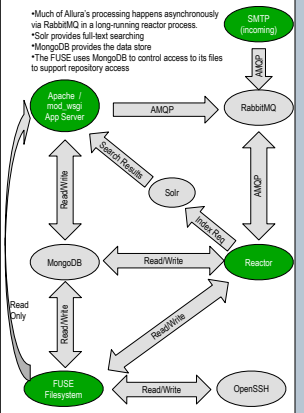


Markdown-based — Automatic Cross-linking

Git, Hg, SVN Fork/Merge Request — Email Integration

Architecture

- Much of Allura's processing happens asynchronously via RabbitMQ in a long-running reactor process.
- Solr provides full-text searching
- MongoDB provides the data store
- The FUSE uses MongoDB to control access to its files to support repository access



Custom Tools

```
...
entry_points=''
[allura]
Chat=chat.main:ChatApp
''
...
```

setup.py

```
Class ChatApp(Application):
    def __init__(self, project, config):
        self.root = RootController()
        self.admin = AdminController()
        self.api_root = APIController()
```

chat/main.py

```
class RootController(object):
    @expose('chat.templates.index')
    def index(self):
        return dict(app=c.app)
...
class AdminController(object):
    def _check_security(self):
        require(has_project_access('tool'))

    @expose('chat.templates.admin_index')
    def index(self, project, config):
        return dict(app=c.app)

class APIController(object):

    @expose('json')
    def index(self):
        return dict(messages=message_list())
```

chat/controllers.py

Custom Artifacts

```
class ChatMessage(Artifact):
    class __mongometa__:
        name='chat_message'
        indexes = [ 'timestamp' ]
    timestamp = FieldProperty(datetime, if_missing=utcnow)
    sender = FieldProperty(str)
    text = FieldProperty(str)

    def index(self):
        return dict(super(ChatMessage, self).index(),
            text=self.text)

    def url(self):
        return self.app_config.url() + self.timestamp.strftime('%Y%m%d')

    def shorthand_id(self):
        return '[' + self._id + ']'
```

Platform Services

All artifacts automatically indexed in Solr and searchable

Markdown rendering with automatic cross-referencing between artifacts

Threaded discussions with email integration

Flexible email notifications and subscriptions

RSS and Atom feeds

Role-based permissions admin

Fine-grained per-artifact ACLs

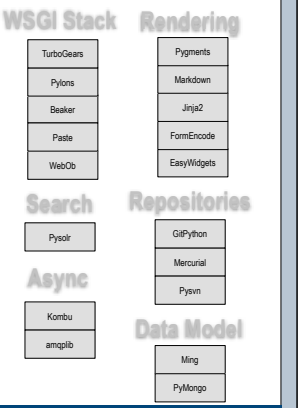
Server-side OAuth 1.0

Asynchronous Processing

```
class ChatApp(Application):
    ...
    @audit('Chat.#')
    def my_custom_auditor(self, routing_key, data):
        # do stuff based on c.project, c.app, c.user, etc.
    ...
    @react('Tickets.change')
    def my_custom_auditor(self, routing_key, data):
        # perhaps publish a change notification to channel
    ...
```

- AMQP topic subscriptions used to route messages
- Reactor setup command provided to configure RabbitMQ
- Use Auditors for async processes, Reactors for passive processes

With Special Thanks



Contributing

Feedback (suggestions, bugs, tool ideas) always welcome

Fork our repo, hack away, and request merge
git://git.code.sf.net/p/allura/git

Join us for our sprint at PyCon!

Contact and URL

Twitter: @rick446
Email: rick@geek.net
http://sf.net/p/allura