Website Content

See Also

Richard's Sample Website
Hydra Philosophy & Value Proposition
Samvera Community Principles
Hydra Website Information Architecture
Positioning Hydra relative to other projects

OR10 Hydra Proposal (docx)
OR09 Hydra Proposal (doc)

Web, Wiki & Lists

Web Site

Audience: the projecthydra.org web site will be the primary face of the project to the general public. This may include potential adopters, contributors, & funders; other members of the open source community engaged in related projects; vendors; and the press. Visitors may be interested in the Hydra core, a particular Hydra head or the collaboration. Many of these visitors will be non-technical; none of them will have a surfeit of time to do in depth research.

Objectives:

• present the "Hydra elevator pitch" on "What is Hydra?"
• convey the sophistication & maturity of the project
• demonstrate the size and strength of the project's community
• give orientation & pathway to those who want to engage further

Supporting Content:

• extremely polished visual presentation
• brief, accessible expository text on
  • technical framework & components
  • community framework, principles & members
• demo apps / links to specific Hydra heads
• license information
• link to more resources / wiki / lists
  • include data models
  • include service definitions
  • include links to GIT
• late-breaking news / blogroll
• articles, presentations, events (link to wiki...)

How do you handle keeping the content fresh? News tends to get stale unless someone is either a.) compulsive, b.) paid to do it. Besides best effort, what can we do to keep this current?

• rss of latest commits?
• pointer to web sites

Wiki

Audience: The primary audience for the wiki are active members of the Hydra community; the wiki is their working space and repository for project-related documentation. Curious and interested external parties might also consult the wiki for more details on particular projects, designs, or technical components.

Objectives:

• give Hydra partners a space for collaboration
• give Hydra partners a document repository
• give new and prospective partners access to in depth documentation relevant to their particular areas of interest

Supporting Content:

• Technical documentation on overall framework, components, standards, principles, versions
  • links to GitHub for code
  • more in depth than at Hydra web site
• Data model documentation
• Community framework, groups, members, activities
  • more in depth than at Hydra web site
  • working space for cross-group efforts
• Timeline / History of effort
• Meeting agendas & notes
• Articles, presentations, events ->
  • Press Kit / Publicity materials
Lists

hydra-steering: http://groups.google.com/group/hydra-steering
A coordinating list for members of the Hydra Steering Group.

hydra-tech: http://groups.google.com/group/hydra-tech
The primary mailing list for development work on the Hydra Framework and on Hydra heads like Hydrangea.

hydra-ui: http://groups.google.com/group/hydra-ui
The primary mailing list for design work on the Hydra user experience overall, and for individual Hydra heads of shared interest.

active-fedora: http://groups.google.com/group/active-fedora
The primary mailing list for anyone using or improving ActiveFedora.

Blacklight: http://groups.google.com/group/blacklight-development
The primary mailing list for anyone using or improving Blacklight.

Github

Audience: Active and prospective developers engaged in Hydra.

Objectives:

- distributed version control systems for Hydra source code
- code documentation repository

Supporting Content:

- code
- rdocs

Hydra Project Identity & Structure

The headings of the following sections are copied directly from Producing Open Source Software Section 2.2: Getting Started :: Starting from What You Have [Fogel, 19-29].

1. Mission Statement
2. State that the Project is Free
3. Features and Requirements List
4. Development Status
5. Downloads
6. Version Control & Bug Tracker Access
7. Communications Channels
8. Developer Guidelines
9. Documentation
10. Example output & Screenshots
11. License
12. <Community>

We should make sure we have all of these things covered for _each_ of the Hydra component projects --

- ActiveFedora
- Solrizer
- Opinionated Metadata
- Hydra Repository Plugin for Rails

If we treat each of the Hydra heads as its own collaborative open source project, we should do the same for each of them.

- Hydrangea -- Self-Deposit of Articles & Datasets?
- Hydra ETD
- HyPATIA -- Archival Collections Management (AIMS)
- Image Collections
- Moving Images & Audio
- Automated Workflow tools & Administrative UI
- Repository Administration Tool

Name

Fogel suggests considering the following when choosing the name for a project [Fogel, 21-22]:

- Gives some Idea what the project does
- Is easy to remember
- Is not the same as some other project's name and does not infringe on any trademarks
- Is available as a domain name

Is Hydra a safe name to use?
Mission Statement

Brief Mission Statement

[One mission statement that speaks to all audiences]

Fuller Mission Statement

[Two mission statements -- one speaking to Business Audience, one speaking to Technical Audience]

Tom’s slides from OR2010 & DLF Fall Forum 2010 have a lot of great info for these.

Business Audience: Knowledge Curators* What does it do?

• Services
• User Communities
• Sustainability, Traction
• Support
• ie. Fedora, not ePrints

Technical Audience: Developers* How does it work?

• Technical Features
• Technical Principles

Matt’s fast stab at a statement:

Hydra grew from a number of developers, librarians, archivists and project managers scratching a collective itch. This itch grew from the fact that humanity is producing an ever-growing corpus of rich, complex digital information. We care deeply about capturing this information, curating it, disseminating it, and preserving it for the long term. In 2008 at the advent of the Hydra project, the tools for curating, presenting, and searching through this content were sorely lacking. We sought to remedy this by producing software that

• provide rich, flexible, faceted search & discovery interfaces
• allow for, and actively encourage, re-use & re-purposing of content
• embrace a diversity of metadata
• lend themselves to being re-used or re-purposed to suit the needs of varying users with distinctive content
• satisfy a baseline of requirements for long-term preservation of content

In addition to these technical goals, we aimed to create the technologies in ways that facilitate and encourage

• inter-institutional collaboration on the code bases
• exploring, identifying and fostering best practices (for repositories, developers, archivists and project managers alike)
• development of specialized applications or "Heads" base on the technologies

Our vision for achieving this focuses around the idea of allowing a single content repository to store & deliver content through a plethora of interfaces that are specialized around particular users interacting with particular content -- one repository body interacting with a wide range of people through its many heads. Hence the name Hydra.

State that the Project is Free

The front page must make it unambiguously clear that the project is open source. … State up front, right below the mission statement, that the project is “free software” or “open source software”, and give the exact license. [Fogel, 23]

Features and Requirements List

…

Does this fit the stack I've got?
Does this fit with my developers’ skill sets?

Development Status

How do we represent the current status of Hydra to outsiders? How do we ensure that this stays up to date?

Near-term Goals & Needs

Development Status Page

Release Cycle
Past releases
How we define "progress"
Continuous Integration?
Real-Time RoadMap
Advice from Fogel:

Don't be afraid of looking unready, and don't give in to the temptation to hype the development status. Everyone knows that software evolves by stages; there's no shame in saying "This is alpha software with known bugs. It runs, and works at least some of the time, but use at your own risk." Such language won't scare away the kinds of developers you need at that stage. As for users, one of the worst things a project can do is attract users before the software is ready for them. A reputation for instability or bugginess is very hard to shake, once acquired. Conservatism pays off in the long run; it's always better for the software to be more stable than the user expected than less, and pleasant surprises produce the best kind of word-of-mouth. [Fogel, 24]

Downloads

What forms of downloads should we provide?* git repository

- Any code libraries distributes as Ruby Gems
- Rails template installer?
- ... war file versions of particular heads?
- Debian packages? RPMs? Mac OS X .dmg installer?

Version Control & Bug Tracker Access (& Continuous Integration)*

Version Control: Git & github

Reference info for using git & github
Description of code structure
Links to each component's github project (active_fedora, solrizer, opinionated_metadata)

Bug Tracker: Jira hosted by DuraSpace

Epics, Stories, Bugs, Documentation, Technical Tasks
Roadmap in Jira ... Greenhopper
Saved searches
Ticket workflow: Open, In Progress, In Review/In Testing, Closed

Continuous Integration: Hudson

What Hudson is, where we have it running.

Communications Channels

Mailing Lists
irc
wiki ... meeting/call notes
Weekly Committers Call
Meetings
Workshops & Training Events

Commit Emails?

Developer Guidelines

Developer guidelines are not so much technical as social: they explain how the developers interact with each other and with the users, and ultimately how things get done.

[...]
- pointers to forums for interaction with other developers
- instructions on how to report bugs and submit patches
- some indication of how development is usually done—is the project a benevolent dictatorship, a democracy, or something else [Fogel, 26]

Documentation

- Tell the reader clearly how much technical expertise they're expected to have.
- Describe clearly and thoroughly how to set up the software, and somewhere near the beginning of the documentation, tell the user how to run some sort of diagnostic test or simple command to confirm that they've set things up correctly.
- Give one tutorial-style example of how to do a common task.
- Label the areas where the documentation is known to be incomplete. By showing the readers that you are aware of its deficiencies, you align yourself with their point of view.

[Fogel, 27]

Availability of Documentation

Documentation should be available from two places: online (directly from the web site), and in the downloadable distribution of the software [Fogel, 28]

Non-Technical Documentation
Developer Documentation

API Documentation on RDModinfo
README files in the code base (& on GitHub)
Read the tests -- RSpec & Cucumber...

Wiki...
Content Modeling: Concepts, Vocabulary, Guidelines, Best Practices, Conventions
FAQ?

Example output & Screenshots

License

Choice of License

Apache2 License?

Applying the Licence

We must be consistent...

Website

Samples:*
Zotero

• Collect, Organize, Cite, Sync, Collaborate (with descriptions & screenshots)*
• clear statement of purpose
• screenshots
• links to support

VuFind

• prominent “view demo”*

Omeka

• Serious Web Publishing, Cost-Effective Design, Flexible and Extensible, Free and Open Source
• Lengthy list of adopters and links to installs
• double-layer menu: one oriented towards casual visitors, the other for serious visitors

Diaspora


Cucumber

Contrast
mysql

Compared against:*
Islandora
ContentDM
Rosetta
CDL Microservices
DSpace
Vital
Omeka
Archivist Toolkit

Sample: Fluid Infusion* Base website: http://www.fluidproject.org/products/infusion/

• Get Involved: http://fluidproject.org/getinvolved/
• Demo/Samples: http://wiki.fluidproject.org/display/fluid/InfusionDocumentation#InfusionDocumentation-Tutorials
• Further Documentation on Wiki: http://wiki.fluidproject.org/display/fluid/Components

Practices
Avoid private discussions [Fogel 32]
Nip Rudeness in the Bud [Fogel 33]
Always record decisions in tickets
Guidelines for writing a good Jira ticket...
Code Reviews [Fogel 34-35]

Reviews should be public. Even on occasions when I have been sitting in the same physical room with developers, and one of us has made a commit, we take care not to do the review verbally in the room, but to send it to the development mailing list instead. Everyone benefits from seeing the review happen. People follow the commentary and sometimes find flaws in it, and even when they don't, it still reminds them that review is an expected, regular activity, like washing the dishes or mowing the lawn.

Social & Political Infrastructure

Forkability
* Collaboration
* Components

Governance

ActiveFedora: Benevolent Dictator (aka community-approved arbitrator)
Solrizer: Benevolent Dictator (aka community-approved arbitrator) .. but should change soon
OM: Benevolent Dictator (aka community-approved arbitrator) .. but should change with maturity
Hydra Rails Plugin: ???