Documentation Thoughts

Documentation for (Potential Adopters / Architects / Coders)

Not all of this will be solely for coders.

documentation style guide

- date the docs and indicate versions numbers of the code
- jump to index for each page
- searchable in site specific google

Overviews

- where to find documentation
  - projecthydra.org ...
  - duraspace wiki ...
  - github wiki
  - rdocs
  - docs included with gem (e.g. README)
- diagrams (can have hovers, be clickable)
  - simplest “read” “cud” (Create-Update-Delete / Read ; Hydra / Blacklight ; Fedora / Solr )
  - entire stack
  - where are API access points
  - relationships among gems
  - If you use xxx then you must also use ... (OM -> Solrizer, AF -> Fedora, ...)
  - access rights
  - get Tom’s pres for docs

- high level overview
  - all the pieces
  - relationships among the pieces
  - links out to more information for each piece / api
  - detailed enough for full info
  - Ray (UVa) will start a doc like this.

- Should I Adopt?
  - what is Hydra
    - what does it do?
    - here are the solutions
    - examples of applications
    - here are the people
    - start out with what it does
  - what is in hydra?
  - how can I use it
  - do I need to use ALL the pieces
  - getting started (three sections: Overview / Architecture / Implementers)

projecthydra.org – new adopters (John Dunn, ...)
duraspace wiki – architects (Lynn, Dean, Simeon, Giarlo)
github – coders (Ben, Naomi, Giarlo)

adopting the full stack

- you don’t have to use full stack to be part of the community
- for architects, for managers (Dean, Lynn, Tom, Richard)
- for implementers - how to do it

diagram/illustrations

detailed enough for full info

example implementations

- duraspace wiki tech details to share (e.g. hypatia)
- projecthydra links to top page for each example

utility of each gem

each gem

- how to use it
- examples of how it is being used
  - in full stack
  - not in stack (multiple examples welcome)
- diagrams / illustrations
Additional Docs

dealing with access / rights

- full stack
- blacklight

Coders

- post to list
- what if no rvm

Reviewing matt's hierarchy of promises document

https://docs.google.com/document/d/1q7jnOOpCHQNY1m7udCbltSh72gTZzNeTUEgzz8fLVilco/edit

look at Islandora user documentation

what is the hydra stack?

how is hydra different from a rails app

how do I know whether or not I'm making a hydra-head

"each piece is pretty clear, it's just when you put them into the stack that makes it fuzzy"

how do we make documentation simple?

high level

each component

when we say "it might be this, it might be that" it becomes much harder to explain

"hierarchy of promises" - what does each piece do for us?

Solr

Fedora

We never interact directly with filesystem, we use Fedora.

Fedora provides these pieces that we actually use (remove extraneous information)

document: comparing Fedora to Microservices

microservices is the next generation design of Fedora, sort of. Michael Giarlo is interested as of 2012

"How Hydra uses Fedora" <-- not a "promise"

document - how you might leverage other parts of Fedora

SQL database and triplestore - not really relevant

Solr .. it's our search index, and lets us do facets

Hydra-Jetty

Hydra-Jetty - gives us a copy of jetty with solr/fedora set up

addl doc: how you can install these pieces into the web service of your choice (links out)

addl doc: how to set up a production web server(s) with these pieces

Rubydora

Rubydora - ruby way to talk to fedora rest api

"is it a gem?"

ActiveFedora

ActiveFedora - a way to work with fedora models as easily as sql models in rails

object-relational mapping to fedora

you can do the equivalent of modeling/creating schema for database tables

also affects how you index things into solr.

how else can this be done (to clarify what is being done with the example)?

if content has xml datastreams ...

then you're just dealing with xml that could have come from anywhere
"active-fedora uses om pretty centrally"

**OM**

OM a way to have a DSL to map xml into solr field stuff simplifying way to manipulate xml perhaps I should try skipping it and using nokogiri is a way to parse xml into ..

nokogiri, libxml, libxslt --> you can always go down to this level and skip the DSL part

OM implies talking to Solr – "currently a convenience thing we could still split off" "is it a gem"?

**Solrizer**

way to get stuff into solr gives you a basic api interface for connecting to Solr you can index an object you can index all the known objects create the solr document for this given object it could get the info from fedora, an xml file, a spreadsheet ...

provides a "to _solr" method for OM docs: "if you've got an OM terminology, then Solrizer knows what to do with it" you can override or extend this

Solrizer itself doesn't do anything - b/c the input to Solrizer link isn't there

**Solrizer-Fedora**

the fedora specific implementation, so Fedora is the source of the data going in to Solr

**JettyWrapper**

to use the hydra jetty,
start, stop, use hydra jetty in your test suite

Jetty is a peer of tomcat; it's a java web application server

Hydra-Cmodels can be used in and of themselves.

All of the above is independent of rails.

"I don't know if the application I'm building is a hydra head"
"I'm interested in using Hydra pieces"
"Do I need to use the whole stack to get the benefits?"
"If I don't use the whole stack, am I using Hydra?"
"How can folks using pieces of the Hydra stack be a part of the community" some folks just use hydra cmodels some folks use pieces of the stack, but not the whole stack

"How would somebody take these things and use them differently?" (examples!) "which pieces require other pieces"

"Hierarchy of Promises" document is:
an architectural vision a way to look at the stack to determine how we should develop the architecture

"Now for the Controversal part"

**Ruby on Rails**

Ruby on Rails is a way to develop a web application MVC Can I use the stack without a Rails app? Can I use the stack without a Ruby app? M - rails + active-model V - rails C - rails rails gets better and better: authentication, engines, ...

**Blacklight:**

if you have stuff in Solr, you can use Blacklight as a read-only (search and discovery) application to your solr index it accommodates heterogenous types of objects

"Each hydra-head has a blacklight for search and discovery"

**HydraHead**
HydraHead - all other application logic needed for hydra head
AssetsController - deprecated
CatalogController - deprecated (the hydra catalog controller - make a rails controller)
"how/why do i do a controller?"
Access Controls - gated discovery, CRUD, granularity <-- could be its own gem - useful outside of rails
"what do these do? how does it work?"
this is in fedora object, but the app uses it from Solr.
Official Hydra Object Datastreams and Models

- are the types on the wiki STILL the right types?
- this code is not really set up for the way we're moving towards with models

"does the hydra plugin imply a dependency on fedora? yes! if you use hydrahead without fedora, it will blow up"

assumes AF, OM, Solr, Fedora ...
does NOT assume HydraJetty, or JettyWrapper

Hydra is slanted towards mods

MODS datastream models and so on is provided – has this worn out its welcome as an expectation that is coded in
"Can I use views if I don't use mods?"

FileAssets and datastreams
how do you find the actual content?
how is this exposed in blacklight and in hyhead

Views:
Submission Workflow - multi-step construction of objects? any thing besides object construction?
Mods Submission Workflow – should this just be an exemplar?
UI/Layout
Javascript - allows workflow to be shown as a single page; auto-save; customization of jquery date-picker
here is where "is it a framework or application" becomes troublesome

UI:
Testability
Accessability

Tension: hydra out of the box is something but should we be providing views?
should submission workflow be a separate gem? etc.

Can use Blacklight + AF if you don't want editing
Can't use HydraHead without
Chart: If you use xxx, then you must use yyy

Poll - who uses which pieces? What versions of those pieces?

Your local rails app - blah blah

Which pieces are plugins vs. not? (Blacklight is a plugin)

What would be easy for people to pick up? What should be in "HydraHead"? What should be in "Hydra"?

example models:
Active-Fedora: sample models and objects with tests in spec directory (AF approach)
Hydra-Head:
official object and datastream models - should be in its own gem? in HydraHead?

random notes:

split out AccessControls so you can apply them to Blacklight (without Hydra)
"how can i use submission workflow"