DevMtg 2012-02-27 - Virtual Summit

DSpace Developers Virtual Summit (Feb 27 to Mar 2)

Why a "Virtual Summit"?

We've often heard feedback that DSpace Developers just don't get enough time to sit down (face-to-face or otherwise) to discuss larger topics or brainstorm out future ideas for the DSpace platform. Our weekly DSpace Developers Meetings just don't allow for enough time to dig deeply into big topics, and there is sometimes a limit to how much we can discuss via text chat (IRC). Often the only time we are able to discuss larger topics in detail are at the Open Repositories (OR) conference.

The idea is that this "Virtual Summit" will be a form of DSpace "unconference", bringing developers around the world together in a virtual fashion via either audio or video technology. Those in attendance will decide the topics, but we have provided some suggested "hot topics" (which have come up frequently in the past) below.

All discussions will also be fed back to the general public. We'll take notes and send those notes back to everyone on dspace-devel listserv. This allows for non-attendees to also provide feedback throughout the Summit (and afterwards).

If all goes well, we may hold these Virtual Summits a few times a year. We also could brainstorm whether it'd be possible to hold a "Virtual Hack-a-thon" to rapidly develop/prototype out ideas that were brainstormed during the Summit.

Who is invited?

- All Committers,
- Any other interested DSpace developers or technology-savvy individuals

At least initially, this summit is geared towards **developers**, as discussion will primarily revolve around the existing codebase & internal architecture. However, discussions from this summit will be forwarded out to the community for broader feedback at all levels (repository manager, user, developer, etc.).

If you don't fall into one of the above categories, you are still welcome to attend. However, be warned that discussion will likely get very technical at times (which is why we recommend you be a developer or have a technology background).

Logistics

- **When:** Meetings will happen daily starting at 20:00UTC and last for 1-2 hours apiece. _We will attempt to keep discussions to about 1 hour in length. But, if topics run over and people have time to continue, we may extend beyond that.
- **Where:**
  - Call in via Skype or Phone line - See #Connection Instructions below
  - In addition, #duraspace IRC will be utilized as a discussion "backchannel" – to share links to current discussion topics/ideas, and potentially even take notes.
- **Attendance Limit:** None.
- **Daily Notes:** Each day we will have a notetaker. Similar to OR11, we may wish to take notes publicly via IRC or PiratePad. Meeting notes will be sent to 'dspace-devel' listserv after each meeting, in order to allow for broader community feedback.

Connection Instructions

You can call in via either Skype or via your Phone. This call will be audio-only. We will also use the #duraspace IRC as a discussion "backchannel" (to share links, etc.)

Please figure out your mute/unmute settings in advance of the call! As there may be quite a few people on the call, we will ask that you try to mute your line when you are not speaking (as this will ensure we minimize any background noise on the line).

Via Phone:

- Dial-in Number: (805) 399-1200
  - Participant Code: 929807#

Via Skype:

1. Search and Add Contact: "freeconferencecallhd.8053991200"
    - Copy and paste freeconferencecallhd.8053991200 (the entire name) into your contact search area
    - add as a contact
2. After freeconferencecallhd.8053991200 is listed in your contact list, simply call that contact.
3. Enter the participant code through the Skype keypad (not by just typing in the numbers on your keyboard): 929807# (you will prompted to add the # sign).
    - Mac Users: The Skype Keypad can be found at the top of your call window (look for the little phone keypad icon).
    - Windows Users: The Skype Keypad is unfortunately hidden under the `Call -> Show Dial Pad` menu
    - Linux Users: The Skype Keypad can be found at the bottom of your call window (look for the little phone keypad icon).

More info on Skype connections also available at: http://www.freeconferencecallhd.com/skypeinstructions.html

Potential Discussion Topics

Format of Meeting

Each meeting is an open discussion format (think "unconference"). Those in attendance that day will decide which topic(s) they would like to cover. These topics may or may not be on the below list of ideas. All meeting attendees are expected to participate in the discussion.

The following topics are just possible topics we may wish to discussion. Topics need not be limited by this listing, and others can feel free to add topics they feel may be important.

In no particular order:

- What may a "Common Business Logic API" look like? Can we build an API that makes it easier for new DSpace UIs to be built?
- Are there ways we could simplify DSpace ("Do one thing and do it well")? This could be at any level (Object Model, API, User Interface, etc.)
- How could we move towards "Metadata on All Objects" (i.e. Not just metadata for Items, but also for Bitstreams, Communities & Collections)?
- What other ways can we start to "Modernize" DSpace at all levels (Object Model, API, User Interface)?
    - As a potential starting point in modernizing DSpace's API see Richard Rodgers' work on GitHub: https://github.com/richardrodgers/mds
- "App Store" or "Sharing Code": How do we work to better share custom code like custom curation tasks, themes, etc. amongst those in our community?
- "DSpace Virtual Hack-a-thon": Could there be a way to hold a virtual hackathon, as a way of rapidly prototyping/developing out ideas in the span of a week or two?
- How to manage ongoing translations of DSpace? This has become harder & harder as we have increased the number of messages.xml files (Discovery, SWORD Client, etc).
    - Should we rethink this direction or look towards Translation Management software, like Pootle (doesn't work for XML messages though) or similar.
- Authentication/Authorization review:
    - More dynamic permission resolution.
    - Centralize decision making for more consistent results across different interfaces.

Summit Meeting Notes

Day #1 Notes (Feb 27)

Attendees: Kevin Van De Velde, Graham Triggs, Peter Dietz, Andrea Schweer, Kim Shepherd, Hardy Pottinger, Mark Wood, Tim Donohue

Primary Discussion topics:

- Interaction/Code Share via GitHub.
    - Can we come up with some best practices for forking GitHub, so that we can all start to share code more easily? I.e. allow institutions to pull in code from one another, etc.
    - We may need to spend some time developing some GitHub best practices.
    - We should also leverage Maven to obviously create separate GitHub projects for features that can just extend DSpace API, etc.
- Business Logic API discussion led into talking about REST API as a potential area for this "common business logic"
- REST API - Who is using it? What should it do, and how should it interact with SWORD / Solr / etc.
    - No one in attendance is using REST API in Production (though a few have used it for development)
    - We should try and "show it off" more by writing some simple Javascript "embed" code to embed DSpace Content in a website. We can post that code up to http://demo.dspace.org and enable the REST API there. This could also allow for more testing.
    - REST API should **never** create its own 'conventions' for common tasks. Rather it should use existing standards (e.g. SWORD for deposit, etc)
    - Question: Should REST API "wrap" things like SWORD & Solr (which can be accessed RESTFully themselves)? Could "RESTFul" DSpace just be a combination of SWORD + Solr + Admin REST API + other RESTFul interfaces?
        - Several in disagreement here. Pros/Cons to various approaches
        - Fedora actually has a separate REST API from Browse/Search. You Browse/Search via RSearch, and then lookup an item by ID via REST API. This approach allows you to simplify the REST API to very specific tasks, and use other existing services /interfaces where they are more 'standard' or widely accepted.
        - However, AuthZ may be simplified if you 'wrap' everything (even with a thin wrapper).
        - Also if you 'wrap' the Search/Browse in REST API, then at a later time you could swap out Solr (or whatever) for something like Elastic Search, and your clients would not need to change.

We took notes via IRC. So, the full notes are up in IRC chatroom: http://irclogs.duraspace.org/index.php?date=2012-02-27

Day #2 Notes (Feb 28)

Attendees: Kevin Van De Velde, Mark Wood, Andrea Schweer, Richard Rodgers, Scott Phillips, Mark Diggory, Hardy Pottinger, Peter Dietz, Tim Donohue

Primary Discussion topics:

- Revisiting REST API Discussion
  - Seems to be an assumption that we should only have **one** REST API
  - No reason why we cannot have multiple APIs, or even different implementations (and let the best one win out in the end)
  - Revisiting whether one REST API should 'wrap' all calls (several question this). Tim noticed Twitter (and many other sites) have many many REST APIs
    - https://dev.twitter.com/docs - Twitter specifically has a basic REST API, Search API, Streaming API, etc. They even do OAuth / AuthZ via REST API.
    - Comments from Bojan Suzic (via dspace-devel):

> *I think this is a good idea. In some cases Twitter has multiple APIs for the reason of different underlying architecture and techical implementations. The other reason could be an iterative evolution of their services and infrastructure. In the case of DSpace - maybe this point could be considered from the functional point of view. Generally, all the REST API versions would depend on the same DSpace-API, so the rationale for separation could be based on some other asumption. For instance, user-centric (browse), admin-centric (update) or similar, and/or based on the development effort or resources necessary to carry out the change. The example for the latter may be an outstanding decision about future development which may hold development/release of the component or part which is already clear and non-disputable.*

- Business Logic API
  - Can we "mature" design/modeling of the Business Logic API by thinking of it more as a REST API?
  - **What should make up a Business Logic API:**
    - Item Submission processes (special ingest workflow)
    - Reviewer Workflow processes
    - Creation processes for Collections / Communities (also includes initializing roles, template items, etc. – almost like a "workflow" process to create these objects)
    - Creation processes for Groups / EPeople?
    - Running Curation Tasks
      - Richard Rodgers notes he's already building a demo REST API for Curation Framework using Jersey (http://jersey.java.net/). This will be posted to GitHub in near future
    - Smaller Activities: User feedback, Statistics, metadata registry, bitstream format registry
    - Authority Control? (managing internal or external controlled vocabularies & taxonomies)
  - Questions / Concerns on Business Logic API
    - Need to avoid it being too "large" / all-encompassing.
    - Where do we draw the line between underlying "DSpace Business Logic" and UI? E.g. Is Search/Browse part of Business Logic API? Or is it a separate API?
      - Mentioned that DSpace Discovery Module has been made more generic (no longer Solr specific) – may be the basis for a separate "Search API"?
    - Can we simplify? Is the "Business Logic API" just the REST API (GET/POST/PUT/DELETE objects)? Or is it still more than that?
- REST API usage of Sakai bus
  - Are we satisfied with the Sakai-based REST implementation? Sounds like several have concerns about complexity & ongoing maintanence
  - May need to work towards a new implementation – based on Spring WebMVC or something else (Jersey? Apache CXF?)
  - Should be able to still **reuse** much of existing REST API work – especially the modeling of DSpace Objects as Resources bound to URLs
- Mobile Device Support - DSpace is lagging behind. How do we plan to move in this direction?
  - Several seem interested in this, but no one known to be working directly on it.
  - Two levels of mobile support: (1) Making current UIs more 'mobile friendly' , (2) making DSpace more RESTful to support building native apps/clients
    - http://en.wikipedia.org/wiki/Responsive_Web_Design
  - Suggestion from Bojan Suzic (via dspace-devel):

> *One approach in this direction could be based on ClientUI idea (from GSoC 2011). Atomization and usage of lightweight components/architecture could lead to easier and less resource intensive development, maintenance and customization of UI. Also usage of cross-platform tools such as jQueryMobile, phoneGap or similar (+ REST API) could provide better coverage and require a less effort.*

We took notes & shared additional links via IRC. http://irclogs.duraspace.org/index.php?date=2012-02-28

Day #3 Notes (Feb 29)

Attendees: Mark Wood, Peter Dietz, Tim Donohue, Andrea Schweer, Graham Triggs, Hardy Pottinger, Richard Rodgers, Brad McLean, Robin Taylor, Mark Diggory, Stuart Lewis

Primary Discussion topics:

- Questions around DSpace w/ Fedora Inside updates.
  - Essentially DuraSpace still highly interested in this initiative. However, no active development going on. There have been ongoing discussions/brainstorms.
- DuraCloud & ReplicationTaskSuite questions

    - Talking through how Replication Suite works with DuraCloud --> DSpace generates AIPs (in either METS or BagIt packages), those get replicated to DuraCloud. In your DuraCloud acct you could choose which underlying storage providers (e.g. Amazon) to use. Can use one, or even replicate across multiple providers (or move between providers at a later time).
    - Main use case is backup & restore (and helping DSpace admins to automate it as much as we can)
        - Currently an automated backup can happen via a ReplicateConsumer which can queue AIPs for upload whenever something changes in the system. Still a work in progress though. Working on stabilizing it.
    - Brainstorming some future use cases – one that came up is storing high-res archival quality images/videos in Cloud (DuraCloud), and using DSpace for access copy.
        - Replication Task Suite can support this as it lets you decide which Item Bundles you want included in AIPs. So, you could only include the "high-res" Bundle in AIPs, and exclude any Bundle which had access copies.
- Metadata For All Objects (briefly touched on)
    - Is the "biggest bang for buck" just Bitstream metadata (esp. preservation/technical metadata)?
    - No, also need for enhanced Collection & Community metadata. Some use cases include:
        - Multi-lingual Communities & Collections
        - Subject Headings on Communities and Collections (as a way to more easily search them or organize them)
    - Bitstream metadata also very important. Especially looking towards JHOVE or DROID. Essentially, storing preservation/technical metadata about files is important.
    - Richard's Modernized DSpace work allows for Bitstream metadata (at least skeleton code is there).
    - Question: How should Bitstream metadata be exposed via Search/Browse?
        - Likely Configurable? But configuration can be difficult, if Bitstream metadata is very heterogenous (different schemas, etc.)
- Discovery / Solr
    - Discovery is working towards making facets pluggable (develop new facets), also working towards making indexing pluggable.
        - The latter (pluggable indexing) may be useful in allowing for different **types** of objects/content to be indexed in different ways (This goes back to question about how to index/expose Bitstream metadata)
    - **Big Question: Should we think about making Discovery the only Search/Browse option?** It would allow us to potentially simplify indexing issues as we get into Metadata on All Objects – since we can work towards **one** solution (Discovery/Solr) rather than multiple at once.
        - What would this mean? Essentially we'd replace the "/search/" directory (old Lucene Indexes) and the DB Browse tables with Solr. All searching/browsing would use Discovery, which currently only works with Solr (but should be pluggable to use other underlying technologies – more below)
        - Some Concerns / Questions posed:
            - Does it need to be Solr? Can't we also achieve this abstraction using just Lucene (which now has browse-based libraries), or even a Solr alternative?
            - Also concerns that Discovery in DSpace 1.7 was a bit buggy (possibly even a small "step back"). However, DSpace 1.8 looks to be better. Still, Discovery may need more testing to ensure it's stable & ready.
            - What about JSPUI? Discovery doesn't work yet on JSPUI. Two options: either port it to work with JSPUI (there was some interest in past), or else we'd have to think about deprecating JSPUI altogether.
        - Discovery in DSpace 1.8 offers more abstraction. It's now more "pluggable" and you could hypothetically use something besides Solr. Though Discovery obviously assumes that whatever it is using has similar features to Solr (faceting, etc)
            - Might be interesting if someone were to try running Elastic Search (http://engineering.socialcast.com/2011/05/realtime-search-solr-vs-elasticsearch/) under Discovery.
        - Also worth noting that Solr Statistics can be separated from Discovery. They can uses entirely separate Solr instances as needed (for better scalability of each).
        - Seems to be some interest in consolidating browse/search around Discovery. But, also several outstanding concerns (see above) that need to be answered.
        - One reason to like Discovery (from Stuart Lewis): http://blog.stuartlewis.com/2011/08/26/the-collection-is-dead-long-live-the-collection/
        - @Mire has some extra documentation about DSpace 1.8 Discovery changes it will be sharing. It's imperative that we work to ensure Discovery directions are brought towards central Committer control. That way we can get the proper 'buy-in' to make sure we can all support it, etc.
- SkylightUI - https://github.com/skylightui/skylight
    - PHP-based UI for read-only searching/browsing DSpace (built out of Auckland). Uses DSpace's Solr indexes (built by Discovery)
    - Queries Solr directly, rather than DSpace REST API, as Solr provides native replication support.
    - Essentially just goes against Solr's own REST interfaces. Uses Solr as a "read" API.

We took notes & shared additional links via IRC. http://irclogs.duraspace.org/index.php?date=2012-02-29

Day #4 Notes (Mar 1)

Attendees: Kevin Van De Velde, Tim Donohue, Andrea Schweer, Mark Wood, Graham Triggs, Mark Diggory, Hardy Pottinger, Peter Dietz

Primary Discussion topics:

- Revisiting Discovery Discussion
    - Kevin Van De Velde volunteers to prototype Discovery + Elastic Search (as alternative backend to Solr). Will start up a JIRA issue & code on GitHub
        - Would be a great proof-of-concept that Discovery can use a different backend (something besides Solr)
        - Would also perhaps allay questions/concerns that some seem to have about Solr (performance / memory). http://engineering.socialcast.com/2011/05/realtime-search-solr-vs-elasticsearch/
        - Could also be a step towards making Discovery the default Search/Browse for DSpace.
- Brief discussion on migration to GitHub
    - Vote is a landslide in favor so far.
    - Hints from GitHub on migrating from SVN to GitHub: http://help.github.com/import-from-subversion/
- Detailed Discussion on Maven Project Consolidation
    - SWORD2 maven project uses a different format. Itdoesn't have sub-projects for '-api' and '-webapp'. Rather, uses the capabilities of Maven to build both JAR & WAR from a single Maven module:
        - http://scm.dspace.org/svn/repo/dspace/trunk/dspace-swordv2/
        - http://search.maven.org/#artifactdetails%7Corg.dspace%7Cdspace-swordv2%7C1.8.2%7Cwar
    - Mark Diggory proposes we use this as a "model" Maven Project.

    - **Question #1: Do we want to reorganize existing Maven projects (XMLUI, JSPUI, LNI, OAI, etc) to consolidate the "-api" and "-webapp" projects?**
        - Some general questions about how this would affect user customizations and/or "vendor drops".
            - NOTE: Should not affect anyone using Overlays.
        - We would need to make this change clear (Documentation).
        - Also would need to ensure it doesn't affect our IDEs / Development environments (we doubt it, but needs further investigation)
        - In general, most seem in favor of idea & feel it could be a simplification of our codebase.
    - **Next Steps** (for Question #1)
        1. This new Maven project organization structure should be documented! Likely under Documentation section on Advanced Customisation
        2. Try out this reorganization for one of the main modules (e.g. XMLUI) and ensure IDEs are not affected
    - **Question #2: What about Maven projects like Discovery?** http://scm.dspace.org/svn/repo/dspace/trunk/dspace-discovery/
        - Technically the "-provider" part likely may belong alongside the 'dspace-api' (in some way). In addition, the "-xmlui" parts likely should belong alongside the "dspace-xmlui".
        - This question seems directly related to whether Discovery is the "default" or the **only** option for Browse/Search.
        - But, in some ways, it would also help to simplify the codebase – less maven projects = less confusion. It also makes sense to consolidate all things related to XMLUI under "dspace-xmlui".
        - Wasn't a real conclusion on this question.
- Stepping Back: Larger reorganization questions posed
    - Should Search/Browse even be part of 'dspace-api'? Technically they are more UI related?
    - Should we be taking a step back and conceptually thinking about what are the "pieces" of DSpace? Some possible "pieces" include:
        - Content Model piece
        - Search/Browse mechanism
        - Event Manager (to pass events between pieces)
        - One or more UIs
    - Where does the "Business Logic API" conceptually fit into these "pieces"? Can better defining these pieces help us to better draw the lines between "Business Logic API", REST API , Content Model, UI?
    - Some reflection back on the DSpace 2.0 thought exercises/prototype from 3 years ago.
    - Should we think about having a similar conceptual thought exercise with the current Committers Team (and reflect back/pull from DSpace 2.0 where it still makes sense)?
        - **Idea to try and start this thought exercise tomorrow, using an online whiteboard like Dabbleboard**
- Feeback on Virtual Summit
    - Tim would like feedback on how this Virtual Summit has gone.
        - Has it been worthwhile to you?
        - Should we do this again? (e.g. every 6 months? every year?)
        - Is the timeframe about right? One week of meetings, with about 1-1.5 hours per day (most days we've ended at about 1 hr 15 mins)?

We took notes & shared additional links via IRC. http://irclogs.duraspace.org/index.php?date=2012-03-01 (Starts at [20:01])

Day #5 Notes (Mar 2)

Attendees: Kevin Van De Velde, Andrea Schweer, Tim Donohue, Peter Dietz, Mark Wood, Graham Triggs, Mark Diggory, Hardy Pottinger

Primary Discussion topics:

- Filling out / Approving the #Actionable Takeaways list below
- Helping new Developers / Users get started (brief discussion)
    - Can we get some better documentation or user training in place for new users & new developers?
        - KnowledgeBase
    - Put some basic install instructions up on GitHub too (once we migrate – which seems definite now)
- Advanced Embargo Feature
    - Being talking about at the next DCAT meeting to make sure it meets community requirements
    - Committers should discuss in more detail in future as well – possible 3.0 feature.
- Higher Level Discussion of what encompasses "DSpace"
    - This began on Dabbleboard whiteboard but ran into some technical issues (multiple editors at same time cause it to get confused).
    - General Goal: **Simplify the 'dspace-api' module!** As much as possible it really should just be the DSpace Data Model / Kernel. There are many "pieces" that should be refactored out of 'dspace-api' in future (hopefully as part of Modernized DSpace work)
    - **Things that could/should be refactored out of 'dspace-api':**
        - Content Manipulation (Curation Tasks) -> includes both curation tasks themselves and MediaFilters (which likely should be rewritten as Curation tasks anyways)
        - Ingesters/Disseminators -> includes both packagers and crosswalks
        - Usage Statistics (it's already separate in Solr Stats)
        - External Identifier Services (ala Dryad – see below)
        - Versioning Services (ala Dryad)
        - Usage Event Services?
        - Search / Browse API (already separate in Discovery)
- Dryad Project codebase
    - Dryad has already built both **versioning** and **identifier services** (e.g. DOI) into DSpace!
    - Versioning
        - Example: Version 1, Version 2 of same document (see versions listed at bottom of page)
        - Codebase: https://dryad.googlecode.com/svn/trunk/dryad/dspace/modules/versioning/
    - Identifier Services:
        - Example: The same document can be accessed via different identifiers: DOI link (http://dev.datadryad.org/resource/doi:10.5061/dryad.1385.2), Handle link (http://dev.datadryad.org/resource/info:hdl/10255/dryad.36265), default DSpace Link (http://dev.datadryad.org/resource/10255/dryad.36265)
        - Codebase: http://code.google.com/p/dryad/source/browse/trunk#trunk%2Fdryad%2Fdspace%2Fmodules%2Fidentifier-services
    - Several are highly interested in seeing these Dryad features make it into DSpace 3.0!
- Feedback on Summit

- Positive overall. Many like the voice chat (easier to cover larger topics)
- A week is too long? 3-4 days is "sweet spot".
- Maybe we should start having occasional meetings via Skype? e.g. once per month? or just "Special Topic Meetings"?

Actionable Takeaways

These are final takeaways from the discussion notes above. We've tried to assign these to one or more people when we could, but some are just 'philosophical' takeaways (in how we plan to move the platform forward). For more details on each takeaway, see the discussion notes above.

- **REST API**
  1. Look to replace Sakai bus implemenation with something else (Spring WebMVC?) – *Unassigned* (though Mark Diggory & Bojan Suzic have been discussing)
  2. Install REST API on http://demo.dspace.org (allows for more testing/visability, etc) – *Unassigned* (Tim can take if no one else has time)
  3. Create some "embed this" Javascript code that goes against REST API – (Assigned: Peter Dietz with help from others)
  4. "RESTFul DSpace" philosophy:
     - Individual REST APIs should stay simple & not attempt to replace/rewrite any existing standards (e.g. SWORD).
     - We should encourage multiple RESTful interfaces (similar to Twitter). For example, Search/Browse API may likely be separate from the basic REST API.
     - We should also encourage multiple implementations (as necessary), with an eye towards choosing a "best in class".
- **Discovery**
  1. Prototype whether Elastic Search can be used as an alternative to Solr. (Assigned: Kevin Van De Velde)
  2. Vote on whether Discovery should become default or even "only" Search/Browse mechanism in future. (Tim & all)
- **Maven Project Consolidation**
  1. Document our new standard Maven Project structure (SWORD2 is the model). Eventually this should go under: Advanced Customisation
  2. Test to ensure this new consolidated structure still works well with major IDEs
  3. Actually consolidate existing Maven subprojects (e.g. XMLUI and JSPUI subprojects) to use this new structure (Assigned: Mark Diggory with help from others)
- **Mobile Device Support**
  1. Look to better optimize our existing UIs for mobile devices (a vague task) – *Unassigned*
     - Andrea Schweer knows of folks who are working on making Mirage Theme more Mobile friendly. Hopefully they can give that work back, if it can be made generally useful.
  2. Hope that a stable set of REST APIs can let others build native client apps or mobile-tailored UIs (e.g. ClientUI project (from GSoC 2011))
- **Metadata for all Objects**
  - *Takeaways were not very specific.* Just that this is important, and that there are use cases for having metadata on Bitstreams, Communities & Collections.
- **Modernized DSpace**
  1. This is an important project which many seem to be watching closely. Important to keep it moving forward (All of us).
  2. "DSpace Kernel/API" philosophy:
     - We should strive to simplify the Kernel. It has gotten bloated over time, and we should work to separate out individual "functions" into other APIs. This will be an ongoing process, but it falls to all of us.
- **Business Logic API**
  1. Bring some of the related Dryad work out into more public areas (GitHub? Wiki?) – Mark Diggory with Ryan Scherle
  2. Investigate whether WebMVC code could be the basis for some of the Business Logic API (Mark Diggory and many others)
- **Dryad Codebase -> Possible 3.0 Features**
  1. Dryad has several major features (Versioning & External Identifiers) that would be very flashy for 3.0. We need to start looking more closely at this work, and get a team of us working on bringing it into 3.0. (Assiged to All in coming weeks)
  2. Pester MarkD & @mire (in a nice way) to help get this Dryad code documented so that the Committers can begin helping to make it generalizable.
- **GitHub Migration**
  1. We will turn off the "issue tracker" as we don't want to confuse folks. We want all issues logged in JIRA.
  2. We likely may want to enhance the README to be more helpful / perhaps even provide some quick install hints (along with links to install docs, etc.)
- **Virtual Summit itself**
  - Mostly positive feedback so far (Send your feedback to Tim!)
  - In future, may want to hold Summit from Mon-Thurs (4 days) or Mon-Weds. The Friday meeting is harder as it ends up being Friday evening in Europe/UK & Saturday morning in NZ.
  - Regularly Scheduled Summit?
    - Once a year? e.g. January/February
    - Twice a year? e.g. January & May/June
    - Three times a year? e.g. January, May & Sept