DSpace 7 Community Sprint 1

Sprint 1: May 7-18, 2018

Sprint 1 Summary

Our first (virtual) community sprint featured eight developers and four coaches, representing nine institutions and five different countries. Two sprint participants even created their first pull request to DSpace!

During the two-week sprint, ten Pull Requests (PRs) were developed, approved & merged into DSpace 7, with another four PRs still in-progress. Work concentrated on bug fixes and building out administrative functionality, primarily in the REST API layer.

- Completed
  - REST API: #2050, #2051, #2052, #2053, #2057, #2060
  - Angular UI: #243, #263, #267, #268
- In-Progress
  - REST API: #2061, #2064, #2066
  - Angular UI: #264

Sprint 1 Participants

Thanks to our Sprint #1 Participants and their institutions! We appreciate your hard work during this sprint, and we hope to see you continue to contribute to DSpace.

- Terrence W Brady (Georgetown University, USA)
- Bill Tantzen (U of Minnesota Libraries, USA)
- Pascal-Nicolas Becker (The Library Code, Germany)
- Pablo Prieto (Almat, Mexico)
- Michael Marttila (Georgetown University, USA)
- James Silas Creel (Texas A&M, USA)
- Michael Spalti (Willamette University, USA)
- Giuseppe Digilio (4Science, Italy)

Sprint 1 Coaches

Thanks also to our Sprint #1 Coaches! Your help and support during the sprint was noted as a great positive in the participant feedback (see below)

- Art Lowel (Atmire, Belgium)
- Andrea Bollini (4Science, Italy) and Luigi Andrea Pascarelli (4Science, Italy)
- Tim Donohue (DuraSpace)

Sprint 1 Feedback

Feedback from participants (provided during Sprint Wrap Up meeting). Feedback has been anonymized / aggregated.

- Positive:
  - Very helpful to receive more hands-on experience. Greate way to get started with DSpace 7
  - Appreciated ability to get immediate feedback (day to day) during sprint on Slack, etc.
  - Enjoyed that Sprint helped prioritize this effort (on their local schedule) / helped them set aside time for concentrated work/attention on DSpace
  - Learned a lot
  - Felt well supported by Sprint coaches.
  - HAL Browser for new REST API is awesome
- Could be improved / Challenges
  - Lots to learn. Getting around DSpace code took a while (for those newer to DSpace)
  - Some tickets could use more information on how to get started / recommended approach (especially useful for newer developers). But, lots of feedback provided from coaches & on Slack as they dug in.
  - Integration Tests took getting used to. May want to improve documentation on creating them / detailing how to run tests, etc.
  - Some were working with Spring for first time (in new REST API). More training / documentation on Spring concepts & how they are used in REST API could be useful in getting up-to-speed more quickly
  - DSpace Developer environment (with an IDE) took some time to get setup. Wiki docs could be improved / updated, but Dev Show and Tell - IDE Showcase video was helpful to some
  - Not as much effort on the Angular UI during this Sprint. We need to work on more balance for future Sprints. We could even consider doing some future sprints specific to either REST API or Angular UI, to ensure consistent effort on both layers.

Sprint 1 Schedule

- Sprint 1 Meetings (Please do your best to attend all three meetings. If you cannot attend a meeting, please touch base with Tim Donohue for any updates, etc.)
  - Sprint 1 Kick Off Meeting (one hour): **Monday, May 7** at 14:00UTC (10:00am EDT) in DSpace Meeting Room
  - Mid-Sprint Meeting (one hour): **Monday, May 14** at 14:00UTC (10:00am EDT) in DSpace Meeting Room
  - Sprint Wrap-Up (Closing) Meeting (one hour): **Friday, May 18** at 14:00UTC (10:00am EDT) in DSpace Meeting Room
- Daily Standups are on #dev-sprint Slack. Just provide your daily updates via text chat, prior to 15:00UTC each day.
- More information on Sprint Schedule / Meetings can be found at Sprint Schedule

Sprint Goals

(NOTE: Currently these goals are very high-level. They need to be broken down into specific tasks / tickets prior to the Sprint, so that those tickets are claimable by individuals.)

1. *Alphabetic jump to (specific letter) in results listing*
   a. E.g. Jump to a Title / Author starting with a specific letter or string, like http://demo.dspace.org/xmlui/browse?type=title
2. Build out Administrative Search/Browse tools (each of these tools is limited to Site Administrators)
   a. *Find Item by Internal Item ID/Item Handle* (e.g. XMLUI: http://demo.dspace.org/xmlui/admin/item , JSPUI: http://demo.dspace.org/jspui/tools/edit-item)
   b. *Administrative Search feature, which allows for faceted searching & browsing of "hidden" items. This feature would **replace** the following two Administrative Browse use cases (by providing a way to filter items by withdrawn status and/or private status)*
      i. *Browse Withdrawn Items (by Title)* (e.g. XMLUI: http://demo.dspace.org/xmlui/admin/withdrawn , JSPUI: http://demo.dspace.org/jspui/dspace-admin/withdrawn)
      ii. *Browse Private Items (by Title)* (e.g. XMLUI: http://demo.dspace.org/xmlui/admin/private , JSPUI: http://demo.dspace.org/jspui/dspace-admin/privateitems)
3. Administrative Edit Item form
   a. Build a simple edit item form that mirrors the existing XMLUI and JSPUI edit item pages (e.g. XMLUI: http://demo.dspace.org/xmlui/admin/item, JSPUI: http://demo.dspace.org/jspui/tools/edit-item)
      i. This form should allow Administrators to add/update/delete any metadata field values for this item.
      ii. This form should allow Administrators to withdrawn/reinstantiate the item
      iii. This form should allow Administrators to make the item discoverable/private
      iv. This form should allow Administrators to add/update/delete bitstreams attached to the item.
      v. This form should allow Administrators to delete an item
4. Improve developer documentation / getting up-to-speed documentation.

Sprint Tickets

Angular UI tickets

All Angular UI tickets are managed in GitHub Issues (under the DSpace/dspace-angular project). *You will need developer access to claim individual GitHub tickets (contact Tim Donohue or Art Lowel (Atmire))*

There are two ways to view the available tickets – either in GitHub directly, or via our dspace-angular Waffle Board. Both views point at the same tickets, so changes in one are reflected elsewhere.

- Sprint #1 "Milestone" (Start with tickets which are labeled "Ready". Other tickets may have dependencies that need to first be completed.)
- Waffle board (Any ticket in the "Ready" column can be claimed, though we recommend those with the "Sprint 1" label)

REST API tickets

All REST API tickets are managed in DuraSpace JIRA (as the REST API is being developed on the current "master" branch of the DSpace codebase). *You will need developer access to claim individual JIRA tickets (contact Tim Donohue)*

JIRA tickets (All "REST API v7" tickets with the "community-sprint1" and "ready" label)

key	summary	type	updated	assignee	priority	status	resolution	labels

Unable to locate Jira server for this macro. It may be due to Application Link configuration.

Additional, unassigned REST API v7 tickets can be found in JIRA. However, we recommend selecting a ticket with the "community-sprint1" label.

Sprint Task Estimates (outdated)

This section has been replaced by the Sprint Tickets listed above. It has been kept just for historical purposes. Please claim tickets above.

UI Tasks

These are somewhat lower level than the goals above, in order to get an idea of the effort required, but need more detail before they can be turned in to github issues

Alphabetic jump to (specific letter) in results listing

- **Service**
  - **Effort**: 24h
  - **To do**
    - construct the Rest URL based on the params
    - parse the response
    - ensure everything gets cached properly
    - build remotedata objects for results
    - write tests
- **StartsWithComponent**
  - **Effort**: 16h
  - **To do**
    - show letters as links
    - show a startswith input
    - handle the case for phone screens (too small to show the alphabet)
    - call the service with the correct query
    - links should work with js disabled
    - component should be abstract enough to be used with any kind of list
    - write tests
- **Browse By Title Component**
  - **Description**
    - has an alphabetic selector at the top
    - shows an item list
    - can be paged and sorted
    - can be scoped by a community or collection
    - can be filtered by a metadata field
  - **Effort**: 8h
  - **To do**:
    - create the component
    - configure the routes
    - call the service with optional scope and metadatafield filters
    - configure existing components for alphabetic filtering, object lists and pagination to work with the correct data
    - write tests
- **Browse By Metadata Field Component:**
  - **Description**
    - has an alphabetic selector or a date selector at the top
    - shows a list of metadata values
    - can be paged and sorted
    - links go to a browse by title view filtered by the scope and the metadatafield of this component
  - **Effort**: 16h
  - **To do**:
    - create the component
    - configure the routes
    - configure existing components for alphabetic filtering, object lists and pagination to work with the correct data

- create a component to select by date
- reuse the search's date widget?
- Create an object list entry for metadata values
- the metadatafield it works with should be configurable: title, author, subject, etc.
- this should reflect in the route e.g. `/collection/:id/browse/title`
- and the i18n labels.
- It should work with or without a scope
- write tests

Build out Administrative Search/Browse tools (each of these tools is limited to Site Administrators)

Find Item by Internal Item ID/Item Handle

- **Routing by handle**
  - **Effort**: 24h
  - **To do**:
    - Create an index that maps handles to self links
    - Modify the DSOResponseParsingService, to update the handle index every time an object is added to the cache
    - add hasByHandle and getByHandle methods to ObjectCacheService
    - modify RequestService.configure to check by handles as well to determine if something is cached.
      - perhaps make the fields to check for configurable, I could imaging a check by DOI for example
    - Add findByHandle to DataService
    - configure handle based routes, that redirect to the DSOType based route
    - write tests
- **Find Item admin component**
  - **Examples**:
    - XMLUI: http://demo.dspace.org/xmlui/admin/item ,
    - JSPUI: http://demo.dspace.org/jspui/tools/edit-item
  - **Effort**: 8h
  - **To do**:
    - write a component that allows you to enter the ID or handle
    - on submit, determine whether it's an id or a handle, and redirect to its edit page
    - configure the route, using a guard that limits access to authorized users
    - ensure the return button on the edit page returns you to the Find Item admin page
      - preferably by using the route history in the store (or even browser history) rather than specifying the source page as a param
    - write tests

Administrative Search

- **Current Browse Withdrawn By Title:**
  - XMLUI: http://demo.dspace.org/xmlui/admin/withdrawn ,
  - JSPUI: http://demo.dspace.org/jspui/dspace-admin/withdrawn
- **Current Browse Private By Title:**
  - XMLUI: http://demo.dspace.org/xmlui/admin/private ,
  - JSPUI: http://demo.dspace.org/jspui/dspace-admin/privateitems
- **Effort**: 12h
- **To do**:
  - configure the route, using a guard that limits access to authorized users
  - create the component based on the search page component
  - Use the set of facets for the administrative search. These will be configured on the rest side, and should contain a section to facet on withdrawn and private items
  - Add labels to the search result components for withdrawn and private items. Use bootstrap's warning style labels, add them to the front of the author row.
  - write tests

REST Tasks

These are somewhat lower level than the goals above, in order to get an idea of the effort required, but need more detail before they can be turned in to JIRA issues

Alphabetic jump to (specific letter) in results listing

- **Add REST method to compute the offset / specify the offset as a startWith**
  - **Effort**: 4h
  - **To do**
    - write the contract (see https://github.com/DSpace/Rest7Contract/pull/23)
    - add support for the new parameter
    - write tests
      - can be scoped by a community or collection
      - must work with both item level browse than metadata browse
      - must respect the error codes specified in the contract

Build out Administrative Search/Browse tools (each of these tools is limited to Site Administrators)

Find Item by Internal Item ID/Item Handle

- **Find a DSpaceObject by identifer (handle)**
  - **Effort**: 8h
  - **To do**:
    - write the contract (it should be a redirect to the specific items/communities/collections endpoint)
    - implement a new dedicated controller
    - write tests
      - valid identifiers
      - unvalid identifiers

Administrative Search

- **Effort**: 24h
- **To do**:
  - introduce special withdrawn and private prefix for discover configuration
  - enable the discover controller to search over withdrawn and private items
    - fix the "public search" configuration to exclude withdrawn / private items
    - write a default discover configuration that reuse the default filters / facets
    - write a SolrSearchPlugin to exclude withdrawn and private items to non-administrators
  - write tests

Build out Administrative Edit item

Metadata representation

- **Express the metadata in the DSpaceObject as a map**
  - **Effort:** < 8h
  - **To do:**
    - replace the List<MetadataEntryRest> metadata in the DSpaceObjectRest with the Map<String, List<MetadataValueRest>> metadata used in the DataDescribe section of the workspaceItem (see https://github.com/DSpace/Rest7Contract/blob/master/workspaceitems.md#single-item)
    - update all the tests to reflect the new structure
    - update the REST contract samples

Patch items

- **Add/Remove/reorder metadata**
  - **Effort**: 24h
  - **To do**:
    - write the contract (it must be based on PATCH)
    - implement the patch method in the Item repository using a support class so to be reusable for each DSpaceObject (Collection, Community, EPerson, ...)
    - write tests
- **Withdrawn, Reinstatiate, make private or discoverable an item**
  - **Effort**: 8h
  - **To do**:
    - write the contract (it must be based on PATCH)
    - implement the patch method in the Item repository
    - write tests

Delete an item

- **Effort**: 8hr
- **To do**:
  - write the contract (it must be a DELETE on the item endpoint)
  - implement the method in the item repository, it should consider where the item is used (it could be a template of a community/collection or wrapped in a workspace/workflow)
  - write tests

Manage bitstreams

- **Add bitstreams to an existent item**
  - **Effort**: 8hr
  - **To do**:
    - write the contract (it should be based on POST multipart)
    - implement the method in the Item repository, it should be possible to specify the Bundle where the bitstream will be created
    - write tests
- **Delete bitstreams in an item**
  - **Effort**: 8hr
  - **To do**:
    - write the contract (it must be a DELETE on the bitstreams endpoint)
    - implement the method in the bitstream repository, it should consider where the bitstream is used (it could be a logo of a community/collection)
    - write tests
- **Reorder the bitstreams of an item (out-of-scope?)**

- **Effort**: ???
- **To do**:
  - write the contract (it should be based on PATCH... is the current item representation good enough?)
  - implement the method
  - write tests