2023-06-29 DSpace 7 Working Group Meeting

- Date
- Attendees
- Current Work
  - Project Board
  - New Feature development process for 7.6
  - Issue Triage process for 7.6
- Notes

Date

29 Jun 2023 from 14:00-15:00 UTC

Location: https://lyrasis.zoom.us/my/dspace?pwd=RTk4QUhISnhPRi9YenVrTFJKbDllQT09 (Meeting ID: 502 527 3040)**. Passcode**: dspace

- More connection options available at DSpace Meeting Room

Agenda

- (60 mins) General Discussion Topics
  1. DSpace 7.6 release retrospective
     a. Anything we feel we'd like to do differently as we move towards 8.0?
     b. Any processes/procedures we'd like to keep in place for 8.0?
  2. Disbanding the "DSpace 7 Working Group". Moving back to general "DSpace Developer" Meetings
     a. Proposing simply renaming this meeting & keeping the meeting time the same.
     b. Wiki obviously will need to be updated to note this change and make it easy to find where the developers meet
  3. Planning for 8.0 (Summary from Steering meeting yesterday)
     a. Goals for 8.0
        i. Move forward major features which missed 7.x.
           1. COAR Notify support (4Science & Harvard)
           2. OpenAIRE integration with notification broker/claim service (4Science)
           3. Porting "REST-Based Quality Control Reports" from old REST API to new one. (U of Laval, Canada)
           4. Duplicate Detection in Submission ported from DSpace-CRIS (The Library Code)
        ii. Include new features which empower users in the admin UI. Make things easier for Admins.
        iii. Improve documentation, training to allow for greater community contributions. (Ease setup/install/customization, etc.)
           1. Angular upgrade/maintenance. Spring upgrade/maintenance. Solr upgrade/maintenance, etc.
           2. Possibly need cleanup of Submission Refactor to support Angular upgrade. https://github.com/DSpace/dspace-angular/issues/858
              a. Library used to create the Submission form may need updating?
     b. **Timeline for 8.0 release: April 2024**
     c. In parallel, proof of concepts / planning regarding modularization (e.g. 4Science angular proposal) and OCFL/preservation storage (Lyrasis proposal to be discussed in more detail).
  4. Planning for 7.6.x releases - bug-fix only.
     a. Tim will bring to Steering the suggestion to switch post-7.6 release numbering to 7.6.1, 7.6.2, 7.6.3 (for eventual bug fix release). This clarifies that 7.6 is the final feature release, and that every later release is a minor upgrade.
     b. Two development branches: `dspace-7.x` and `main`
     c. Two project boards in GitHub: DSpace 7.6.x Maintenance and DSpace 8.0 Release
     d. Revisiting code review process brainstorms: See Incentivizing Code Reviews and PR Testing
  5. *(No Updates)* Demo Site migration to Lyrasis (https://demo7.dspace.org/ and https://api7.dspace.org/server/)
     a. Tim will work with Lyrasis to make this happen as soon as reasonably possible now that 7.6 is released.
     b. Demo site will be renamed back to "demo.dspace.org" (instead of "demo7.dspace.org").
  6. Future meeting discussions for 8.0
     a. 4Science proposed to present
        i. COAR Notify on July 13th 2023
        ii. ORCID Login improvement on July 20th 2023
        iii. Angular : library-based architecture proposal updated proposal on July 20th
  7. (Other topics?)

Attendees

- Tim Donohue
- Art Lowel (Atmire)
- Andrea Bollini (4Science)
- Paulo Graça
- Mark H. Wood
- Grazia Quercia (4Science)
- Corrado Lombardi (4Science)
- Julian Timal (eScire)
- Martin Walk
- Melissa Anez

Current Work

Project Board

DSpace 7.6 Project Board: https://github.com/orgs/DSpace/projects/23

To quickly find PRs assigned to you for review, visit https://github.com/pulls/review-requested (This is also available in the GitHub header under "Pull Requests Review Requests")

New Feature development process for 7.6

Per a decision of DSpace Steering, **new features for 7.6 are only welcome if they used to exist in 6.x.** Everything else must be a *fix* that would normally be acceptable in a "bug fix only" release.

7.6 is a "transition" release, where we are transitioning back to our release numbering scheme. As of 8.0, new features will only be allowed in major releases (8.0, 9.0, 10.0) and minor releases will only include bug/security fixes (8.1, 8.2, 8.3).

- **For brand new UI features**, at a minimum, the UI ticket should contain a description of how the feature will be implemented
  - If the UI feature involves entirely new User Interface interactions or components, *we recommend mockups or links to examples elsewhere on the web*. (If it's useful, you can create a Wiki page and use the Balsamiq wireframes plugin in our wiki)
  - Feature design should be made publicly known (i.e. in a meeting) to other Developers. **Comments/suggestions must be accepted for TWO WEEKS, or until consensus is achieved** *(whichever comes first)***.** After that, silence is assumed to be consent to move forward with development as designed. (The team may decide to extend this two week deadline on a case by case basis, but only before the two week period has passed. After two weeks, the design will move forward as-is.)
  - This does mean that if a UI feature is later found to have design/usability flaws, those flaws will need to be noted in a bug ticket (to ensure we don't repeat them in other features) and fixed in follow-up work.
- **For brand new REST features (i.e. new endpoints or major changes to endpoints)**, at a minimum we need a REST Contract prior to development.
  - REST Contract should be made publicly known (i.e. in a meeting) to other Developers. **Comments/suggestions must be accepted for TWO WEEKS, or until consensus is achieved** *(whichever comes first)***.** After that, silence is assumed to be consent to move forward with development. (The team may decide to extend this two week deadline on a case by case basis, but only before the two week period has passed. After two weeks, the design will move forward as-is.)
  - This does mean that some REST features may need future improvement if the initial design is found to later have RESTful design flaws. Such flaws will need to be noted in a bug ticket (to ensure we don't repeat them in other features) and fixed in follow-up work.
  - *REST API Backwards Compatibility support*
    - During 7.x development, we REQUIRE backwards compatibility in the REST API layer between any sequential 7.x releases. This means that the 7.1 REST API must be backwards compatible with 7.0, and 7.2 must be compatible with 7.1, etc.
      - However, deprecation of endpoints is allowed, and multi-step 7.x releases may involve breaking changes (but those breaking changes must be deprecated first & documented in Release Notes). This means that it's allowable for the 7.2 release to have changes which are incompatible with the 7.0 release, provided they were first deprecated in 7.1. Similarly, 7.3 might have breaking changes from either 7.1 or 7.0, provided they were deprecated first.
    - After 7.x development, no breaking changes are allowed in minor releases. They can only appear in major releases (e.g. 7.x8.0 or 8.x9.0 may include breaking changes).
- **No new Entity Types will be accepted in 7.x**
  - Because new out-of-the-box Entity Types require strategic planning, we have decided that we will be unable to accept new Entity Types in any 7.x release. That said, any newly suggested Entity Types will be passed along to Steering / Leadership so that they may be considered during the planning of the 8.0 release.
  - Enhancements, improvements or bug fixes to the Configurable Entities feature itself, or existing out-of-the-box Entity Types are still welcome in 7.x. We want to ensure that Configurable Entities is made as stable and usable as possible in 7.x, in preparation for discussions of new entity types in 8.x and beyond.

Issue Triage process for 7.6

- **Overview of our Triage process:**
  1. *Initial Analysis*: Tim Donohue will do a quick analysis of all issue tickets coming into our Backlog Board (this is where newly reported issues will automatically appear).
  2. *Prioritization/Assignment*: If the ticket should be considered for this release, Tim Donohue will categorize/label it (high/medium/low priority) and immediately assign to a developer to further analysis. Assignment will be based on who worked on that feature in the past.
     a. "high priority" label = A feature is badly broken or missing/not working. These tickets must be implemented first, as ideally they *should be resolved* in the next release. (Keep in mind however that priorities may change as the release date approaches. So, it is possible that a "high priority" ticket may be rescheduled if it is a new feature that cannot fit into release timelines.)
     b. "medium priority" label = A feature is difficult to use, but mostly works.. These tickets *might* be resolved prior to the next release (but the release will not be delayed to fix these issues).
     c. "low priority" label = A feature has usability issues or other smaller inconveniences or a non-required feature is not working as expected. These tickets are simply "nice to have" in the next release. We'll attempt to fix them as time allows, but no guarantees are made.
  3. *Detailed Analysis*: Developers should immediately analyze assigned tickets and respond back within 1-2 days. The developer is expected to respond to Tim Donohue with the following:
     a. Is the bug reproducible? (If the developer did not understand the bug report they may respond saying they need more information to proceed.)
     b. Does the developer agree with the initial prioritization (high/medium/low), or do they recommend another priority?
     c. Does the bug appear to be on the frontend/UI or backend/REST API?
     d. Does the developer have an idea of how difficult it would be to fix? Either a rough estimate, or feel free to create an immediate PR (if the bug is tiny & you have time to do so).
     e. Are you (or your team) interested in being assigned this work?

4. *Final Analysis:* Tim Donohue will look at the feedback from the developer, fix ticket labels & move it to the appropriate work Board. If it is moved to the Project Board, then the ticket may be immediately assigned back to the developer (if they expressed an interest) to begin working on it.
    a. If the ticket needs more info, Tim Donohue will send it back to the reporter and/or attempt to reproduce the bug himself. Once more info is provided, it may be sent back to the developer for a new "Detailed Analysis".

Notes

- **Feedback on 7.x process**
    - **Scope:** Way too much in 7.x (too long to get out - 4 years is a long time)
        - Kept adding more features. Too much "scope creep". Not strict on release dates
        - Took time to improve what is there which resulted in larger refactors (couldn't avoid all "scope creep" that occurred & there were positives out of it.)
    - Early releases we may have taken too much time to **discuss** solutions. Got better about that later on.
    - We learned the hard way: move features, not release dates. We should keep this.
    - DSpace 7 is a massive improvements over 6.x.
    - Massive overhaul (UI and REST API) is not something we'll need to do frequently. Was necessary in 7.x.
    - **Code Reviews:** Lose a lot of time keeping PRs up-to-date without being able to anticipate when reviews will occur.
    - Originally noted that 7.x needed more than just a flashy UI. But, maybe that was all we needed all along
    - **Community involvement:** Did we do too much work to bring them along early on? It wasn't easily attainable to bring community members along until we "finalized" the 7.x platform.
        - Hadn't documented the design principles well enough (and the design principles sometimes changed)
    - **Funded Development:** absolutely **necessary** to ensure this level of effort could occur for 7.x. Allowed service providers to balance work on open source DSpace much better.
    - **Staffing:** More support for Tim/Tech Lead.