2021-11-18 DSpace 7 Working Group Meeting

Date

18 Nov 2021 from 15:00-16:00 UTC

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

- More connection options available at DSpace Meeting Room

7.2 Release Plan

Tentative Release Schedule:

- *Thursday, Dec 23 (PR Creation Deadline):* All new feature (or larger) PRs should be created by this date. (Smaller bug fixes are welcome anytime)
- *Thursday, Jan 20 (Review/Test Deadline):* All code reviewers or testers should submit their feedback by this date. Code reviews must be constructive in nature, with resolution suggestions. Any code reviews submitted AFTER this date will be considered *non-blocking* reviews. This means feedback received after Jan 20 is *optional* to address (unless the team or PR developer decides it is required).
- *Thursday, Jan 28 (PR Merge Deadline):* All new feature PRs should be merged by this date. (Bug fixes can still get in, as long as they are small or important)
- *Monday, Jan 31:* Internal / Early release goal. If possible, we'd like to release 7.2 in late January or first week of Feb.
- *Monday, Feb 7:* Public Release Deadline. 7.2 *must be* announced/released by this date.

Ongoing/Weekly tasks:

- Tackle/Claim issues on 7.2 board (starting with "high priority")
- Review/test all PRs assigned to you for review/testing: https://github.com/pulls/review-requested (*Prioritize reviews of "high priority" PRs first*)

Agenda

- (30 mins) General Discussion Topics
    1. Cancelling or rescheduling next week's meeting. Thurs, Nov 25 is USA Thanksgiving & Tim will be unavailable to attend. We can either cancel or move it to Weds, Nov 24 at 15UTC
    2. Planning 7.2 Release Features
        a. Tickets have been assigned based on last week's meeting & moved to "In Progress" column of our board. By coincidence, both teams claimed the same amount of work.
            i. Atmire: 194 hrs estimated dev work (includes 100hrs for migrating Live Import sources, see below)
            ii. 4Science: 194 hrs estimated dev work
        b. *(Discussion)* Migrate extra DSpace-CRIS "Live Import" (Scopus, Web of Science, CrossRef & OpenAIRE) sources to DSpace core: https://github.com/DSpace/DSpace/issues/3359 (roughly 100 hrs?)
            i. Is there an updated estimate?
            ii. If we can pay for this out of the DSpace budget, would 4Science prefer to do this porting themselves? Or are they OK with others being paid to port features from DSpace-CRIS? If this is reassigned, we'll need to address the imbalance of work above.
    3. Upgrading to Angular 11 now (https://github.com/DSpace/dspace-angular/pull/1403) & Angular 12 later (likely in 7.3)
        a. Is anyone interested in starting an Angular 12 (or 13) upgrade in January, for possible inclusion at the very beginning of 7.3?
    4. Are there features which we know will need additional review/tester help?
        a. Testing DSpace 7 Pull Requests - Step by step instructions to help anyone (even non-technical folks) test a PR using Docker & GitHub CLI
    5. *(Other Topics?)*
- (30 mins) Planning for next week
    - Review the Backlog Board - Are there any tickets here stuck in the "Triage" column? We'd like to keep this column as small as possible.
    - Review the 7.2 Project Board - Assign tickets to developers & assign PRs to reviewers.
        - Paid (by DSpace project) developers must keep in mind priority. If new "high" or "medium" priority tickets come in, developers should move effort off of "low" priority tasks.
        - Volunteer developers are allowed to work on tickets regardless of priority, but ideally will review code in priority order.

Attendees

- Art Lowel (Atmire)
- Andrea Bollini (4Science)
- Tim Donohue

- Lieven Droogmans
- Giuseppe Digilio (4Science)
- Ben Bosman
- Paulo Graça
- Mark H. Wood

Current Work

Project Board

DSpace 7.2 Project Board: https://github.com/orgs/DSpace/projects/14

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 7.2

- **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).

Issue Triage process for 7.2

- **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 7.1, 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 *must* be resolved prior to 7.1. (Keep in mind however that priorities may change as the 7.1 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 7.1 timelines.)
        b. "medium priority" label = A feature is difficult to use, but mostly works.. These tickets *should* be resolved prior to 7.1 (but the 7.1 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 7.1. 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 7.1 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".

On Hold Topics

- Will we need subminor Release Numbering for 7.x?
    1. Should we potentially have subminor (e.g. 7.0.1) releases to fix major bugs or security issues? Or would those need to wait for the next minor (e.g. 7.1) release?
    2. Considering regular, scheduled releases instead of feature-based releases?
        a. Every **quarter** (3 months) release a new 7.x. Features will be implemented in priority order, but not necessarily guaranteed for a specific release.
- Discussing Release Support now that 7.x is out
    1. The DSpace Software Support Policy notes that we support the "most recent three (3) major releases" (where a major release is defined by the changing of the first number, e.g. 6.x 7.x). This would mean that 5.x, 6.x and 7.x are all supported at this time.
    2. Should we propose to Committers / Governance a change of policy to the "most recent two (2) major releases"? This would mean that we move to only supporting 6.x and 7.x.

Notes