Tale of Two Editors: Sinopia
Introduction

- Linked data editor and supporting backend API and data storage
- Developed by LD4P (Linked Data for Production) funded by Andrew W. Mellon Foundation
- help.sinopia.io
- stage.sinopia.io
History

- Initially a fork of the original Library of Congress BIBFRAME Editor
- Major Refactoring began in 2018
- Initial Sinopia Stack
  - Included a separate forked copy of the Profile Editor
  - Linked Data Platform Trellis for storing RDF
  - Deployed on Amazon Web Services
  - Integration with Question Authority
- Subsequent work cycles
  - Migrated from Trellis to MongoDB
  - Refactored templates to RDF, removed Profile Editor
  - Copied UI elements from current LOC editor
  - User dashboard for accessing recently used search queries, templates, and resources
Community-Focused & Collaborative Editing

- From Sinopia’s inception, design and implementation supports multiple groups
- Groups can be institutions (i.e. Stanford University, Cornell University, Library of Congress) or organizations (PCC, MLA)
- Existing Sinopia implementation allows universal editing by any authenticated user BUT
- Current work-cycle is adding permissions to restrict editing by users in specific groups, still universal read access
General RDF Editor and Datastore

- Users can use any ontology in their resource templates
- However, Sinopia is optimized for BIBFRAME
  - Search
  - Rdf2marc
  - UI enhancements
- Examples of other ontologies
  - Schema.org
  - Performed Music Ontology
  - RDA FRBR
- Users can also Sinopia API for non-BIBFRAME record management, backend datastore agnostic to what RDF is stored, however RDF resources do have BIBFRAME-specific metadata
Agile Software Development

- Sinopia team organization and workflows based on Agile software development Best practices:
  - Work in Short 1-2 week sprints that together make up a work-cycle, typically 6-8 weeks in length
  - Unit and integration tests for functionality and bug fixes
  - Continuous integration, tests are run and must pass to merged into main code branches
  - Get immediate feedback from users for UI and functionality changes
- Iterative process, flexible to changing requirements, resource availability, and developer experience
- Stakeholders and users drive requirements
- “Kanban”-like project boards track tasks through identification, assignment, development, and deployment
Third-Party Integrations

- Focus of latest LD4P grant, “Closing the Loop”
- Sinopia PCC resources will be sent to
  - Casallini’s ShareVDE
  - OCLC Entity back-bone
- Current work-cycle building extract-transform-load workflows
  - **Extract** Sinopia resources
  - **Transform** RDF to MARC21 and JSON records
  - **Load** records into ILS and Library Services Platforms
Thank-you! Questions?

Sinopia Environments:

- Development at https://development.sinopia.io/
- Stage at https://stage.sinopia.io/
- Production at https://sinopia.io/

Code Repositories:

- Sinopia Editor at https://github.com/ld4p/sinopia_editor
- Sinopia API at https://github.com/ld4p/sinopia_api
- Other supporting projects at https://github.com/ld4p
LC Bibframe Editor

BIBFRAME Workshop in Europe 2021

Matt Miller
Network Development and MARC Standards Office
Library of Congress
Starting Point

- An editor to advance BF100
- Designed by the catalogers
- Bibframe opinionated
- Make use of recent web development practices: reactive, stateful, etc.
Timeline

- Late 2019 - Early 2020: Contracted SAMHÆNG to work with pilot catalogers to design new editor
- Mid 2020 - Development began
- Mid 2021 - Soft launch
- Rest of 2021 - Stabilization, bug fixes, feature development
Technical Considerations

- Powered by profiles (uses the old profile editor)
- Communicates in RDF/XML - compliments LC’s ecosystem
- Client heavy, builds XML payloads client side
- Reactive, uses Vue.js
- Simple Dockerized stack:
  - MongoDB powered LDP backend
  - Node Express backend API
  - Nginx routes and servers client
Devil is in the details, rubber meets the road, <insert idiom here>

- When your horizon is production you will unlock new layers of requirements.
- In the new editor’s case that includes
  - Design / Usability requests
  - Features that support existing workflows
  - Hidden requirements
- In our case we are building in an environment of having the same cataloging system for the last 20 years.
- Need to support converting BF records back to MARC
Our development cycle

- Message board with all participants, they interact and ask questions give suggestions, feature requests, etc.
- Monthly all participants meetings: announcements, updates, demos, etc.
- Monthly division BF meetings
- Weekly Office Hours meeting
- These generate issues and feature requests, we have a list where people "vote" on what they want prioritized
- New versioned release every couple weeks
Examples: Supporting Workflows

- Copy cataloging workflow.
- Needed to quickly validate a subject heading to determine if it needs review.
- Checkout Kevin Ford’s talk on this from LD4: https://sched.co/joAW
Examples: More(?!?) complicated interface

• More textual based subject editing interface

• Designed for catalogers who know what headings they want to use and just want to type them quickly

• Needs to support all possible pre-coordinated heading construction

• Can be turned off. Make UI features toggleable
Examples: Hidden/Soft Requirements

- Non-latin input is very important. An ecosystem of solutions/workarounds had developed for our ILS to input diacritics

- Something the new editor had to support, 20 years of muscle memory is a high hurdle

- Toggleable shortcut “packs” that replicate existing diacritic shortcuts used in the ILS
Examples: Reducing New Complexity

- Cataloging now involves keeping track of Works vs Instances vs Items. Feedback that is was hard to do in the editor interface.

- A little mini-map of the layout of the resource to help communicate what is going on in the record/resource.
Next Steps

• Continued development cycles

• Once stabilized, development of more ambitious feature requests

• Starting to get feedback that is more “It would make things easier if the editor did…” Making the editor a way to make cataloging easier/quicker.

Thanks for listening!

Matt Miller
mattmiller@loc.gov
@thisismmiller