Common Annotation,

Tagging, and Citation

at Harvard

CATCH

Library Lab Final Project Report

November 2013

By

Philip Desenne, Paolo Ciccarese and Martin Schreiner

Accomplishments

We developed a cloud-based working prototype of the backend Annotation Hub (AH) database and Application Programing Interface (API) for the Common Annotation, Tagging, and Citation at Harvard (CATCH) project (https://github.com/annotationsatharvard/catcha). The AH enables storing, searching, sharing and analyzing scholarly annotations, tags, including semantic tagging, produced on four digital media types: text, image, audio and video. The AH is designed to interoperate with third-party annotation multi-media tools to aggregate and associate contextualized annotation metadata with reference to persistent digital media in repositories, such as the Harvard Library DRS (see fig. #1).

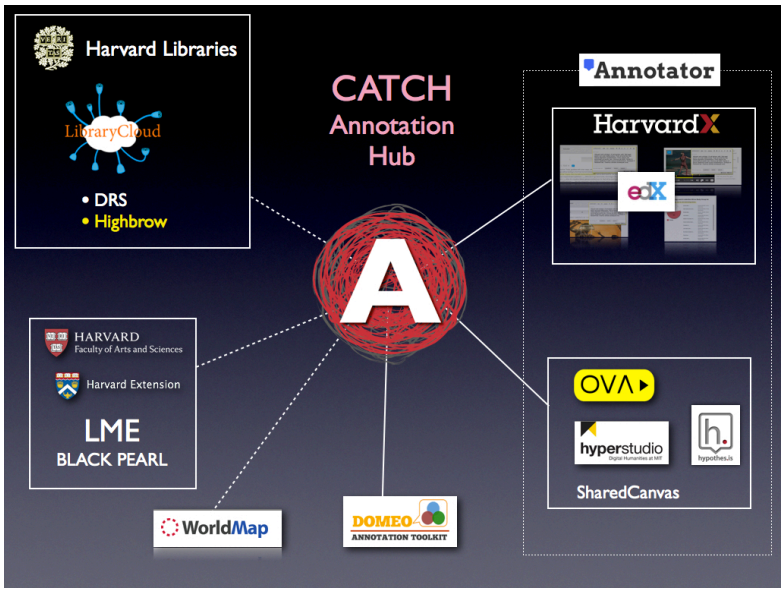

Figure 1: Annotation Hub connectivity

All annotation data stored inside the AH is normalized (translated) into the Open Annotation data model W3C standards (http://www.openannotation.org/spec/core/). The aggregated annotation metadata can then be queried and analyzed through the exposed API (see CATCH API detail section).

Along with the CATCH back-end we developed a media independent, customizable client side, Javascript Annotation Grid Display UI module to view and browse the aggregated annotation data across all media types stored on the database (see fig. #2). This UI is a window into the database that displays media-rich annotations and offers a link back to the original source of the media fragment that was annotated. It features simple sorting and filtering capabilities to browse and share annotations.

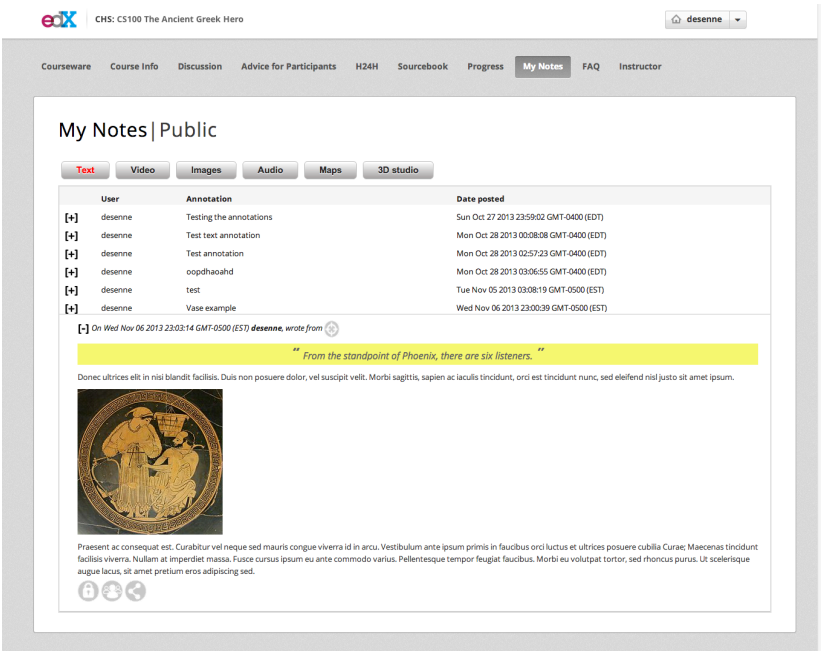

Figure 2: Example of Annotation aggregation in edX

Implementation of AH

Currently any annotation tool that implements the Open Knowledge Foundation Annotator.js library and plugin extensions is capable of interoperating with the CATCH AH. As a proof of concept of this interoperation we successfully connected the Open Video Annotation tool (http://www.openvideoannotation.org/) (see fig. #3) and several edX prototype platform modules for annotating text (see fig. #4), image (see fig. #5), video and audio to the AH database and the Annotation Grid Display.

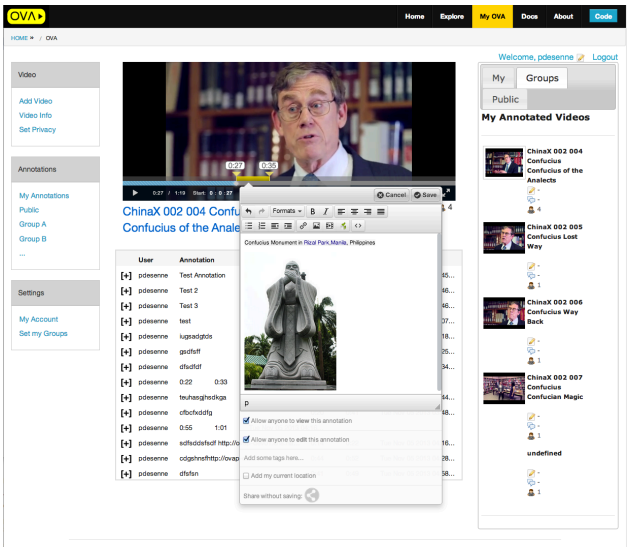

Figure 3: Open Video Annotation

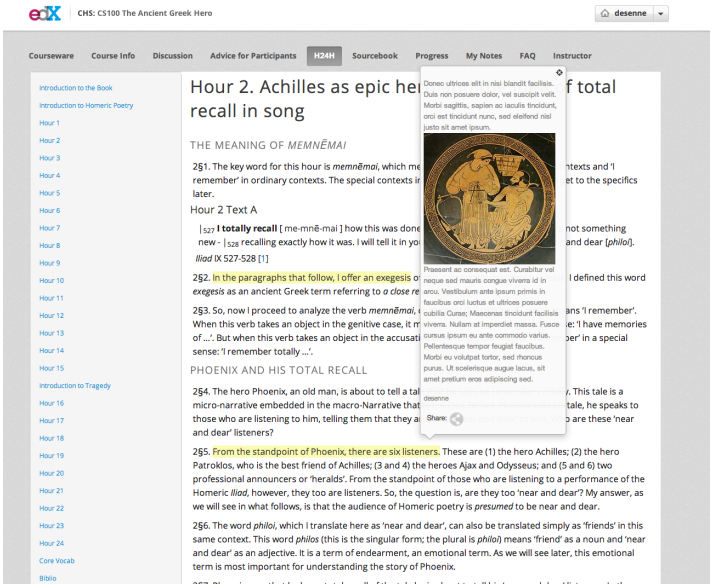

Figure 4: Text Annotation Module in edX

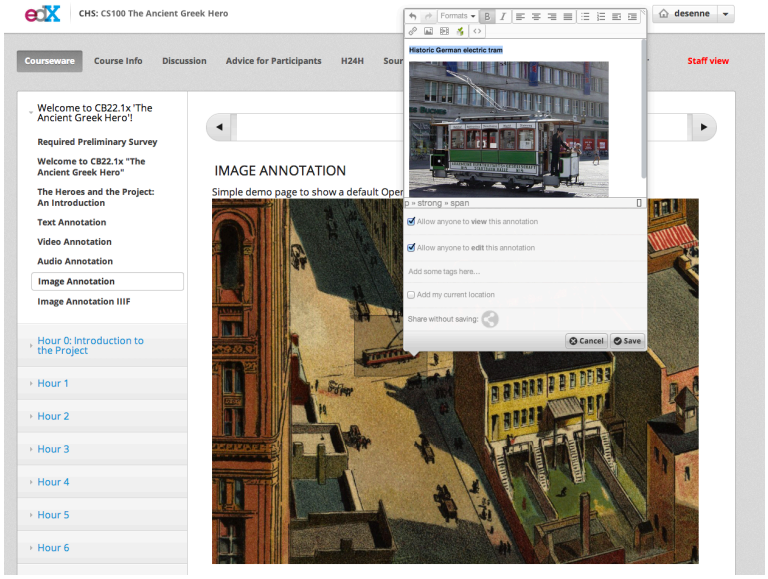

Figure 5: Deep Zoom Image Annotation Module in edX

Other tools that implement Annotator.js, such as Hyperstudio (http://hyperstudio.mit.edu/) from the Digital Humanities at MIT and Hypothes.is (http://hypothes.is/), have the capability to be connected to the AH.

Moreover, we managed to interoperate CATCH AH with non Annotator-dependent tools such as the Domeo annotation toolkit (http://swan.mindinformatics.org/) and Highbrow: A Textual Annotation Browser (https://osc.hul.harvard.edu/highbrow/) from the Harvard Library.

CATCH API details

Storage API

The AnnotatorJS Store API actions allow clients to perform CRUD operations (create, read, update, delete) for annotations they want to store in the CATCH backend. Storing the annotations in a remote backend allows clients to ignore the complexity of building their own backend. In addition, the CATCH API provides other services such as Security and Search which enables the clients to focus on making their applications more user-friendly and not on complex backend logic. The CATCH API also implements actions from the Harvard Annotation API Draft Spec (including archive and destroy).

Search API

The AnnotatorJS Search API is a basic search mechanism that allows clients to retrieve annotations by source, target, annotation text, and other attributes. The current Search API relies on a relational data model to pull out attributes that are available for searching. In other words, we store the JSON annotation document in a large text column and index on attributes that we want to expose to the search API. Therefore, we can only search on fields that we have indexed. In a future release, we will be moving to a document store which will allow us to provide a more flexible search API.

Security (Authentication / Authorization)

The CATCH API provides basic security using the JSON Web Token. The client is responsible for generating an auth token on every request. This is one of the more complicated pieces for client-side developers as it requires the developer to build their own token generator in the language of their choice. The auth token is basically a payload with authentication data (consumer public key, time-to-live, issued-at) signed with a secret shared between the client and server. On every request made by the client, the server checks for an auth token and verifies that the token has not expired. This provides minimal security between the software, but obviously does not handle user authorization and whether a particular user should have access to an annotation. At the moment, the CATCH API does not restrict access to annotations -- all annotations are public. However, the Permissions spec for AnnotatorJS allows clients to provide filtering on the client-side. In the future, the CATCH API will improve on the authorization mechanism and only allow authenticated users with proper permission to access annotations from the API.

Future Plans

The next phase of the CATCH project contemplates the implementation of a document-oriented database designed for ease of development and scaling, coupled with a distributed restful search and analytics engine that is schema free. For this we are considering open source solutions such as MongoDb (http://www.mongodb.org/) and ElasticSearch (http://www.elasticsearch.org). On the client-side we will extend and refine the usability of the Annotation Grid Display to accommodate multiple tool-independent annotation scenarios.

Furthermore, we will continue testing the implementations of the AH and Annotation Grid Display inside the edX platform and define APIs to connect the Library resources from the DRS inside the edX annotation modules.

Expenses and Project Execution
All the expenses for the CATCH project are included in the attached budget information and they were all posted in the general ledger. The allocated budget for the CATCH project was consumed entirely. The majority of the money was spent on independent contractor fees for front-end and back-end development.

We hired a total of three developers during the last trimester of the project. A minor fraction of the budget – 0.5% – was used to rent cloud-server space to run the database and server.

The difficulties we encountered during the execution of the project were 1) locating qualified independent contractors familiar with the topic - the process took more than two months; 2) coordinating front and back-end development to better synchronize data exchange formats.

In retrospect we should have begun development of the AH database and API earlier in the project timeline. Staggering front and backend development would have allowed a more mature data model scaffolding on which to build the front-end UI and enabled easier construction of client-side data handling methods.

Project Team
Paolo Ciccarese - Lead Technical Architect and Data Model Design
Phil Desenne - Lead Application Design and Instructional Technology
Martin Schreiner - Library Integration and Quality Assurance
Justin Miranda - Back-end Database Development
Daniel Cebrian - Front-end UI Development
Mark Soper - Front-end UI Development
Lynn Sayers - Project Budget and Finances