Microservices for Score and State Saving of Aggregated Interactive Assignments

Cay Horstmann
San José State University
San José, CA
cay.horstmann@sjsu.edu

ABSTRACT

A set of microservices enables the inclusion of arbitrary interactive JavaScript elements in assignments, course/lab notes, and textbooks. Implementors of the JavaScript elements add calls to a simple API. The services handle aggregation, state saving, and the LTI protocol. A current implementation is the CodeCheck assignment feature that allows the creation of LTI assignments consisting of autograded coding exercises, code tracing exercises, and Parsons problems.

Author Keywords

online education, interoperability, smart learning content

CCS Concepts

•**Applied computing** → **E-learning;** *Learning management systems;*

INTRODUCTION

When writing digital textbooks, it is now possible to include complex interactive assignments that probe student understanding. These include code tracers (prompting for execution lines or variable contents), Parsons problems, and autograded coding exercises. Reading systems include Runestone [3], OpenDSA [7], proprietary systems (Wiley's zyBooks, Pearson's Revel), as well as EPUB3 readers (VitalSource, RedShelf). One important responsibility of a reading system is to track student progress and to share it with instructors.

These systems are compelling, but with the exception of EPUB3, which is an open standard with multiple implementations, their authoring formats differ greatly from each other. A prospective author will need to invest time to choose among them, and then to learn the chosen system's authoring tools. On the other hand, as discussed in the following section in more detail, course delivery based on EPUB3 is in its infancy.

Thus, an instructor who wishes to author interactive lecture notes, labs, or assignments, which might or might not grow into a book, is faced with a high barrier to entry.

Seventh SPLICE Workshop at SIGCSE 2021 "CS Education Infrastructure for All III: From Ideas to Practice", SPLICE'21, March 15-16, 2021, Virtual Event

A different barrier exists for authors of interactive exercises. Such exercises are typically authored in JavaScript, for display in a browser. Providing a remote storage facility for saving and restoring student work would be a major distraction for the exercise author.

In this paper, I report on the design of the CodeCheck Assignments feature which enables instructors to compose assignments from multiple interactive exercise types, provided they support an extremely simple score and state API which is proposed in the EPUB for Education draft standard.

EPUB3 AND EPUB FOR EDUCATION

EPUB3 [2] is an open standard for digital book packaging. Pages are authored in (X)HTML, CSS, and JavaScript. For offline reading, books are not supposed to contain external references. In practice, many EPUB3 readers do not block network connections in JavaScript code, thus enabling tools for offline code grading, speech analyis, and so on.Tools for packaging material into EPUB3 from sources in HTML, Markdown, or similar, are readily available.

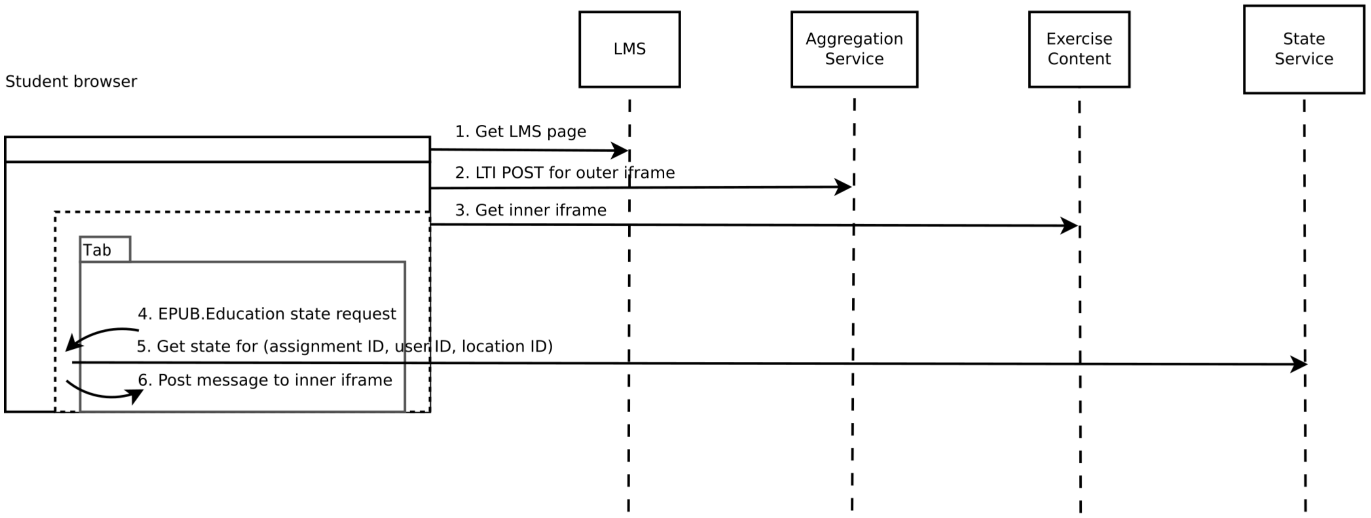
By itself, EPUB3 is not suitable for delivering material in a course. A reading system needs to provide enhanced services such as managing student enrollment, collecting scores, and managing student work state. The EPUB for Education [9] draft standard attempts to standardize on such services, but its progress has been slow and uncertain.

One encouraging aspect of the draft standard is the very simple API that interactive elements use for scores and state. An interactive element can entirely focus on visual appearance, user interaction, and score evaluation. The reader is responsible for transmission and storage of scores and state. The reader provides methods

```
EPUB.Education.reportScores([scores], callback)
EPUB.Education.getScores([locations], callback)
```

Each score is an object with properties `score` (between 0 and 1), `location` (a unique ID in the document), and `metadata` (arbitrary state data that the element saves and retrieves in order to restore student work). The first call saves and array of scores, and the second one returns it to the callback.

This API (or technically, its precedessor) is used the in eight exercise types of the activities in [5] (published by Wiley and delivered as EPUB3 through VitalSource). Adapting to the API was straightforward, and it would have been trivial if that had been part of the original design.



LMS
Aggregation Service
Exercise Content
State Service
Student browser
1. Get LMS page
2. LTI POST for outer iframe
3. Get inner iframe
Tab
4. EPUB.Education state request
5. Get state for (assignment ID, user ID, location ID)
6. Post message to inner iframe

Figure 1. Restoring state in an aggregated assignment

Unfortunately, the EPUB for Education effort seems to have stalled. There is only one known draft implementation (VitalSource). It provides state saving but no satisfactory services for score reporting or LMS integration.

CODECHECK ASSIGNMENTS

CodeCheck [4] is an autograder that is optimized for rapid authoring of small assignments. Its users requested LMS integration, but they wanted to assign problem sets that contain multiple CodeCheck assignments as well as other assignment types such as Parsons problems or tracing exercises. They also demanded state saving as a necessary feature for longer assignments.

One apparent solution is an LTI wrapper such as ACOS ([8]). However, typical LMS cannot include LTI exercises as part of a quiz assignment. (An exception is Moodle that has a plugin for this purpose [1].) Moreover, saving and restoring state is not a part of the LTI standard, nor is it otherwise supported by LMS.

The remedy was to provide two microservices, for state saving and for aggregating exercises into assignments.

The state service saves and restores the state (that is, arbitrary JSON) of an exercise ID, given a triple (assignment ID, student ID, location ID). When used with LTI, the assignment ID is the toolConsumerID + courseID, and the student ID is the opaque user ID provided by LTI. A use outside LTI is also supported, where assignment and student IDs are randomly assigned. (For FERPA reasons, the service does not use regular student IDs.) This is a straightforward service, which, to hold cost down, uses a shared NoSQL database (Amazon DynamoDB).

The aggregation service provides, optionally as an LTI tool provider, a web page with a tabbed interface. Each exercise is placed in an iframe, in order to isolate the JavaScript and CSS dependencies. Each exercise page must import a small JavaScript shim that implements the `EPUB.Education` class. Its methods for state saving and retrieval post messages to the parent frame, which then uses the previously described state service

Instructors are thus able to provide assignments consisting of multiple autograded code exercises, code tracing exercises, and Parsons puzzles. These exercise forms already supported the EPUB for Education calls, but it would be easy to add others.

As a compelling example, the Udacity/CS046 course [6] was turned into a set of assignments that, when placed inside a LMS, almost completely replicate the experience inside the Udacity MOOC environment, except that student progress data is now available to instructors.

A third service simply provides an LTI wrapper with state saving over a single exercise that uses the EPUB for Education API.

RESULTS AND CONCLUSION

Using the provided services, it is straightforward to integrate arbitrary JavaScript-based interactive assignments into larger assignments or lessons. The JavaScript code must be augmented to call the `EPUB.Education` services when the score changes (for example, on completion). Supporting state restoration in an existing element can be more complex, but it is optional.

There is value in separating LTI implementation, aggregation, and state saving as separate services. Providing these services requires a very different skill set than a designer of interactive content typically possesses. It remains to be seen whether designers of existing interactive elements will embrace a simple service-enabling API such as EPUB for Education.

Finally, we often think of building large systems that provide holistic solutions. In contrast, the services in this presentation are focused and tiny by comparison. For instructors, the cost of entry is extremely low. Perhaps as a consequence, there is a steady stream of users, both casual and committed.

REFERENCES

[1] 2020. Moodle LTI Question Type. (2020). Retrieved Feb 15, 2021 from `https://moodle.org/plugins/qtype_lti`

[2] Garth Conbo, Matt Garish, MURATA Makoto, and Daniel Weck. 2019. EPUB3 Overview. (2019). Retrieved Feb 15, 2021 from `https://www.w3.org/publishing/epub3/epub-overview.html/`

[3] Barbara J. Ericson and Bradley N. Miller. 2020. Runestone: A Platform for Free, On-Line, and Interactive Ebooks. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 1012–1018. DOI: `http://dx.doi.org/10.1145/3328778.3366950`

[4] Cay Horstmann. 2006. CodeCheck. (2006). Retrieved Feb 15, 2021 from `https://codecheck.io`

[5] Cay Horstmann. 2019. *Big Java Early Objects* (7th. ed.). Wiley, New York, NY.

[6] Cay Horstmann, Cheng-Han Lee, Sara Tansey, and Kathleen O'Brien. 2013. Udacity/SJSU CS046 Course. (2013). Retrieved Feb 15, 2021 from `https://horstmann.com/sjsu/cs046`

[7] Clifford A. Shaffer, Ville Karavirta, Ari Korhonen, and Thomas L. Naps. 2011. OpenDSA: Beginning a Community Active-EBook Project. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research (Koli Calling '11)*. Association for Computing Machinery, New York, NY, USA, 112–117. DOI: `http://dx.doi.org/10.1145/2094131.2094154`

[8] Teemu Sirkiä and Lassi Haaranen. 2015. Acos Server: Towards Smart Learning Content Interoperability. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research (Koli Calling '15)*. Association for Computing Machinery, New York, NY, USA, 169–170. DOI: `http://dx.doi.org/10.1145/2828959.2828981`

[9] David Stroup, Markus Gylling, Matt Garish, Yonah Levenson Hirschman, Tzviya Siegman, John Tibbetts, Rick Johnson, and Nick Brown. 2019. EPUB3 for Education Draft Report. (2019). Retrieved Feb 15, 2021 from `https://w3c.github.io/publ-cg/education/epub-education.html`