TOWARDS USER-DEFINED WEB APPLICATIONS IN ACCELERATOR LABS*

D. Liu†, Facility for Rare Isotope Beam, East Lansing, USA

Abstract

Most scientists and engineers in accelerator labs understand the basics of data types and data structures. They have in-depth knowledge about accelerator physics and other engineering domains. Some even develop software applications by themselves. However, they are not web application developers, and very few of them can implement web applications that support multiple users and data storage. In the approach of user-defined web applications, a user defines her/his own web application, test and use it first before sharing it to other users. It saves the communication efforts between developers and users, reduces the time from application design to production. Most importantly, users become the owner of the application and naturally the owner of the data that the application collects and produces. This will largely improve an application's quality and user experience.

SCIENTISTS ARE NOT DEVELOPERS

Most scientists and engineers in large scientific research labs are advanced computer users. They use general office software every day, and they need to use specific applications for their research and work. Some of them even develop software by themselves because available commercial software sometime does not satisfy their special needs.

Some applications developed by scientists are well known and used by the community. Some of such applications were released with an open-source licence, while some were provided with just the binaries. The application authors have to spend their own time to provide limited support. Reusing such software as libraries or services in a new development often is difficult due to intellectual property issues and source-availability issues. For the sake of the research community, there should be a platform to the scientists to easily **share** and manage their applications. The platform should lower the burden of software authors for managing and supporting users.

While some scientists have successfully delivered single-user desktop applications, very few have the technical skills to develop multi-user web applications. It is because this requires the software author to use multiple programming languages, develop both client- and service-side software, and know how to deal with aspects like security, data storage, and application protocols like HTTP.

USER-DEFINED WEB APPLICATIONS

In Section 1, we discussed the two problems in scientific application lifecycle:

1. difficulties to share and manage applications, and
2. difficulties to develop multi-user web applications.

In order to address these problems, we propose the approach of user-defined web application (UDWA).

What is a User-defined Web Application?

A web application has the following advantages compared to a traditional desktop application:

- Supporting multiple concurrent users
- Easy to release and update on diverse operating systems
- Easy to share to other users across organizational and geographical boundaries

Obviously, web applications help to address the first problem.

A user-defined web application is a data-centric web application generated by a web platform from high-level user defined specifications, and provides basic graphical user interface (GUI) and application program interface (API). Ideally, the platform provides an environment for users to compose and test the application specifications including:

1. the structure and types of data that application users will read and write, and
2. the data resources the application will consume.

The Epics process variable and other available UDWA data API's should be supported by the UDWA platform.

Minimal Requirements for Users

In order to use UDWA platform, and be able to define the application specifications, a user needs to understand

- basic data types like string and number;
- basic data structure concepts, like an object with properties and values, and an array of basic types, arrays, and objects;
- get and set the value of a Epics process variable; and
- read and update a resource via the HTTP protocol.

Technical Challenges

The first technical challenge to develop the UDWA platform is to have a data storage service that is able to save

1. the application specifications and their change histories; and
2. data instances saved by an application.

Note that the data instances of an application can be collected according to different data types and structures in the changing application specification. This makes it extremely difficult to store the data instances in a relational database, because the instances may require different data schemas. The same difficult is for the changing application specification if we want to save it in a database.

The second challenge is that the generated user interfaces of a piece of data can be rendered equivalently on

* This material is based upon work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DESC0000661, the State of Michigan and Michigan State University.
† liud@frib.msu.edu

ISBN 978-3-95450-169-4

both client and server sides. When a user composes an application in the application builder on the client side, the user should see the rendering result from specification. When the application specification is saved on the server, the server should render the same interface to the owner who defines it, and other users the owner shared with.

Technical Enablers

The answer to the first challenge is schema-less databases. With the development of NoSQL (not only SQL), some new databases do not require predefined schemas for the data to be saved and queried. MongoDB [1] is one of such implementations where data with different structures can be saved in the same collection that is like tables in relational databased.

The answer to the second challenge is JavaScript libraries that run on client and server sides. Node.js [2] is a JavaScript engine that runs on the server side and is able to execute many JavaScript libraries with equivalent functionalities as those running in browsers.

The Whole Picture

Figure 1 shows the major components of the UDWA platform and their interactions. Both client and server sides implement the model-view-controller patter.

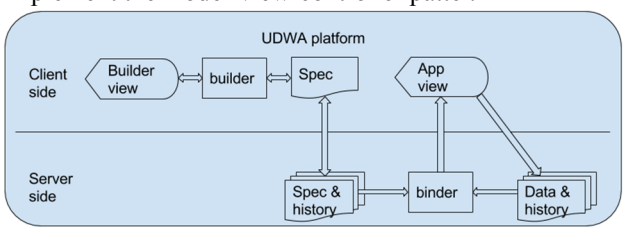

Figure 1. The major components of UDWA platform.

PROGRESS AT FRIB

At FRIB, we have started to develop web application platform where users can define their data and user interfaces for the data. The traveler application [3] was developed in the UDWA principle, and has been in operation for about three years. The traveler application allows users to design their data collection interface in a what-you-see-is-what-you-get (WYSIWYG) way, and to release it by sharing with other users and groups in the lab. Figure 2 shows a screen shot of the WYSIWYG form builder. We plan to develop a generic data store where users are able to define their own data type and data structure, to track structure and instance data changes, and to control the access to the data.

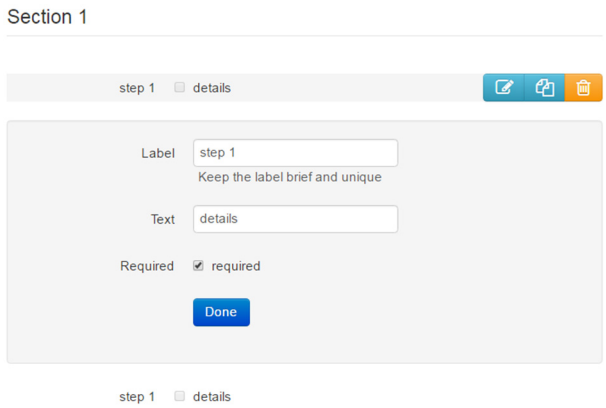

Figure 2. The traveler form builder.

CONCLUSION

The UDWA approach will benefit both the application users and the application developers in the aspects of user experience and application quality. It will reduce the time from application design to release, and therefore accelerate the construction of scientific research facilities like FRIB.

REFERENCES

[1] MongoDB, https://docs.mongodb.com/manual/

[2] node.js, https://nodejs.org/en/docs/

[3] traveler, https://github.com/dongliu/traveler/tree/FRIB