USING THE VAADIN WEB FRAMEWORK FOR DEVELOPING RICH ACCELERATOR CONTROLS USER INTERFACES*

Wenge Fu#, Kevin Brown, Ted D'Ottavio, Seth Nemesure, Enrique Schuhmacher
Brookhaven National Laboratory, Upton, NY 11793, USA

Abstract

Applications used for Collider-Accelerator Controls at Brookhaven National Laboratory typically run as console level programs on a Linux operating system. One essential requirement for accelerator controls applications is the bidirectional synchronized IO data communication. Several web frameworks have made it possible to develop web based Accelerator Controls applications that provide all the features of console based user interface applications. Web based applications give users flexibility by providing an architecture independent domain for running applications. Security is established by restricting access to users within the local network. Additionally, the web framework provides the opportunity to develop mobile device applications that makes it convenient for users to access information anywhere and anytime. The Vaadin Java Web Framework is a tool kit being used to develop client side web interfaces. Vaadin provides Java developers a short learning curve overhead. Most Java Technologies, including JavaEE and third party packages work well within the Vaadin framework. This paper explores the feasibility of using the Vaadin web framework for developing UI applications for Collider-Accelerator controls at Brookhaven National Laboratory.

INTRODUCTION

"*Vaadin Framework is a Java web application development framework that is designed to make creation and maintenance of high quality web-based user interfaces easy*"[1]. First released in 2009, the Vaadin web application framework has been a fast growing API in terms of popularity among web developers for its rich functionality. The Vaadin framework has an advantage over other web development technologies because it uses the Java programming language which is more familiar to the application development community.

The Vaadin application framework provides two programming models: server side (Java) and client side. The client side framework is backed by the Google Web Toolkit (GWT). Program code is written in Java and resides on the server side. Server side program code helps make web applications more secure. Vaadin has a rich set of UI components. The server side and client side communicates via HTTP (or TCP when websockets are used) protocol and transfers data in JSON. The Vaadin framework supports all major web browsers without additional plugins[2] and works well with major IDEs. This makes the web application coding and debugging easier. In Vaadin, the look and feel of the web application is controlled by CSS themes. This makes web application GUI richer, and more configurable; Vaadin is best used for designing single page web applications which typically work like console level UI applications.

* Work supported by Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy.

fu@bnl.gov

VAADIN FOR ACCELERATOR CONTROL APPLICATIONS

Controls applications used in accelerator controls systems have many common characteristics, such as:

- They mostly require fast live bidirectional communications with many different systems such as hardware controllers, database servers, file servers, and other legend systems on different platforms (Unix, Linux, Windows etc.);
- Requires fast UI and interactive responsiveness.
- Rich UI for control data visualization for single or multiple GUIs.

These features can be relatively easy to implement with traditional languages such as C++ and Java. As Vaadin uses the Java programming language, server side java JDK (or JavaEE) APIs, third party APIs and jar packages can be used directly. This makes Vaadin a favorable choice when choosing a web application framework.

In a Vaadin web application, the server side programs (written by developers) and client side code (generated by Vaadin from server side code) have a common shared state, which helps enhance UI responsiveness and overall performance. For web based accelerator controls applications, the UI design and GUI layout are relatively easy to implement. It is critical for the client-server bidirectional communication layer to be effectively managed. Vaadin data push features make this kind of bi-directional communication easy to setup.

Vaadin push can be configured with Java annotations: @Push(PushMode, Transport) in program code or with a configuration file. There are three push modes:

- AUTOMATIC (default)
- MANUAL
- DISABLED

and 3 transport methods for communication in the request/response cycles:

- LONG_POLLING
- STREAMING
- WEBSOCKET

ISBN 978-3-95450-148-9

These push modes and transport methods can be either set in the program or set in a configuration file (e.g. web.xml).

Since STREAMING is very similar to LONG_POLLING and has been deprecated since V7.5.0, this paper just focuses on the long_polling (the "fake" push) and websocket (the real push) methods.

Long polling is a process whereby clients send requests to the server, and the server receive the requests, but holds it for a set period of time. The response happens when new data is available within the time period. Meanwhile, the client keeps the connection open and ready to receive data from the server. In this method, although all requests are initiated from client side, it effectively achieves a real time server push-like bi-directional communication.

Websocket, on the other hand, is a full duplex TCP connection that is independent of the HTTP request/response cycle. Once the connection is established, it is a true real time bi-directional communication.

For accelerator controls applications, both methods work well. But they do have some differences because of the nature with which the communication is established. The long_polling method is easy to setup and is supported by most web browsers. However, there may be a short delay in data transport, while requiring more server resources such as memory. Websockets use dedicated connections between server and clients, use fewer resources and is capable of handling large amounts of client/server connections. Figure 1 diagrams the relationship between web application (client), Vaadin application server and the back end accelerator control system.

VAADIN CONTROL APPLICATION EXAMPLES

Vaadin is primarily designed for single-page web applications which work like desktop applications. The common life cycle for developing Vaadin web applications include, designing and writing web application UI Java code, tuning the look and feel of the UI in CSS, deploying the application to a web container (such as Glassfish server) and testing the application in a web browser. Since the UI design is very similar to strategies used with other languages, the focus of our tests addressed synchronized IO communication with simple GUIs and default CSS settings. We tested Vaadin with 3 different types of controls applications:

1. A single page application with no push (data polling , HTTP).
2. A single page application with manual push in long_polling. (HTTP)
3. A single application with automatic push in websockets. (TCP)

The Glassfish server v4.1 (and v4.0) are used as the application servers.

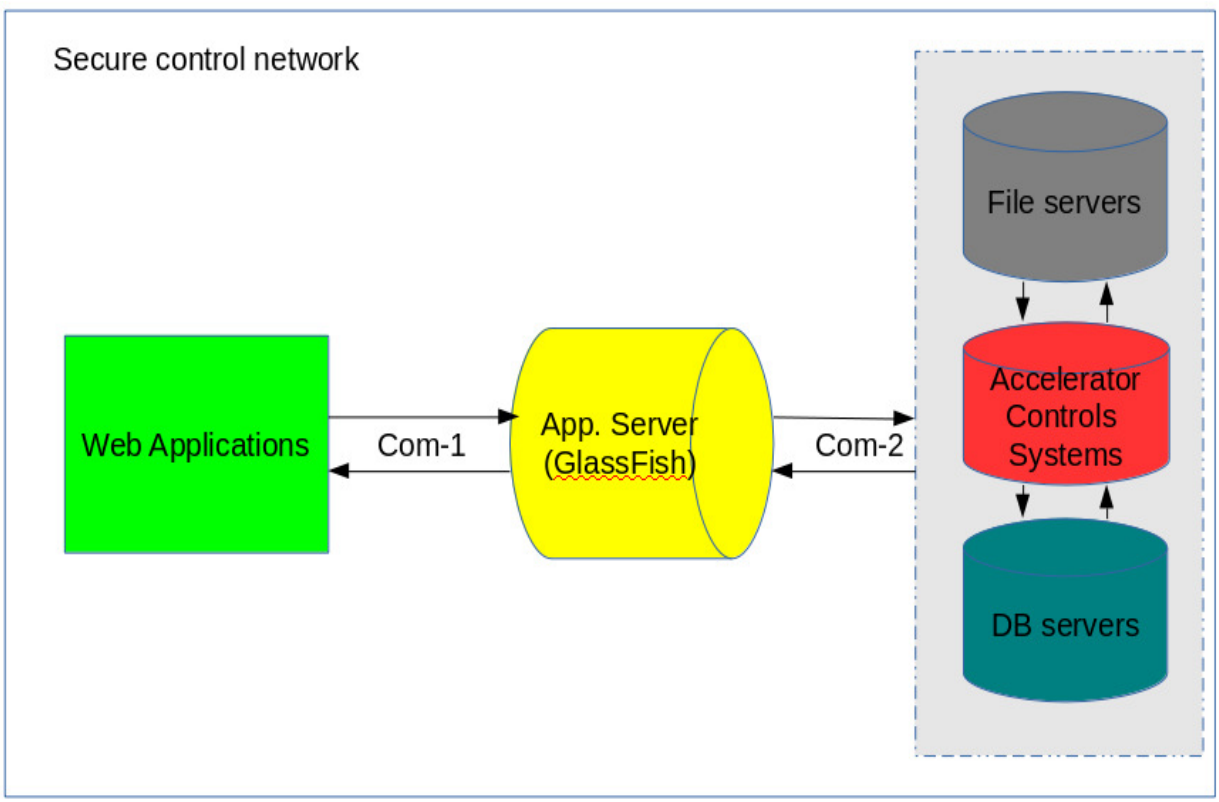

Figure 1: Diagram of Vaadin for accelerator control application

ISBN 978-3-95450-148-9

In the first case, we converted a C++ based controls application into a Vaadin web application. The application's name is SystemViewer. The basic function of this simple program is to display the current statuses of all monitored control systems based on live data in a back end database, and to highlight any system which may have problems or need System Administrators' attention. This program also displays the detailed system data for any monitored systems, and is capable of launching

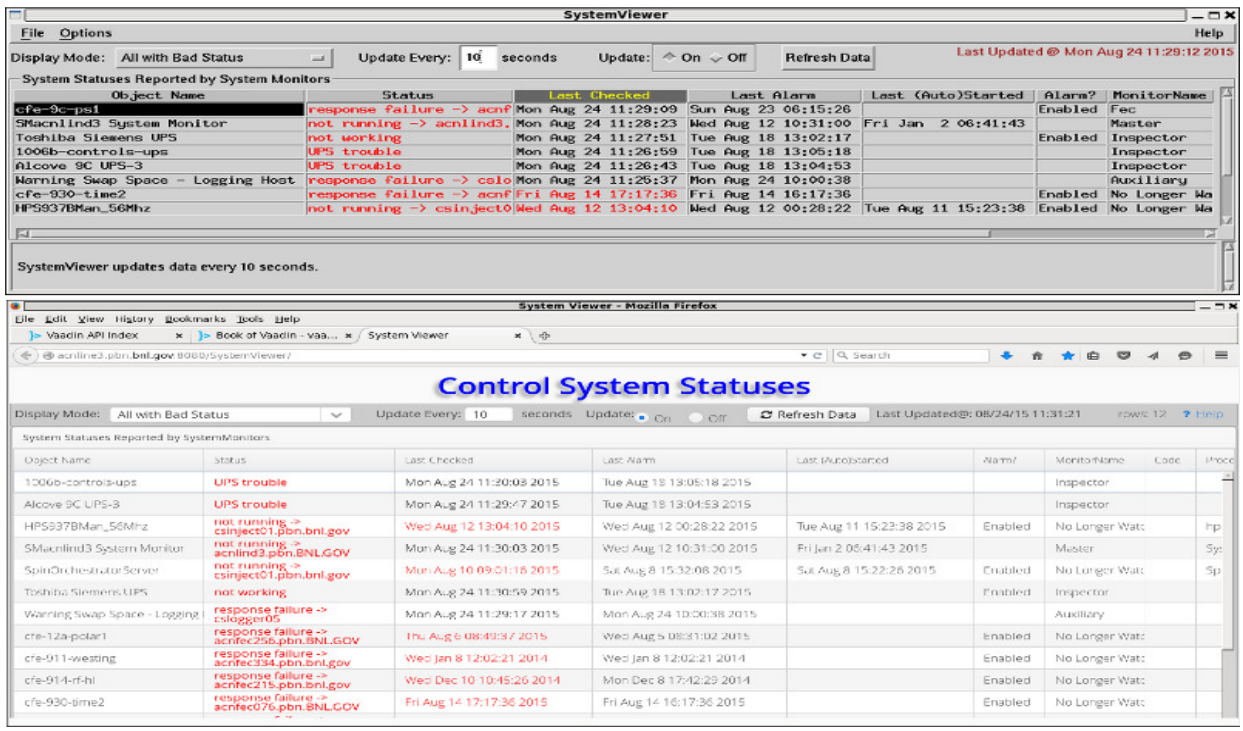

Figure 2: SystemViewer application: C++ (top) vs Vaadin (bottom)

diagnostic tools (other GUI programs) with related context directly from the web application. All features of the C++ version were implemented in the Vaadin version. This makes the program available anywhere within the security network. In this case, all data are periodically polled from the client side and the program works just as fast and as reliable as the C++ version. In fact, the Vaadin version includes more features such as allowing hiding and showing columns in the GUI. Figure 2 shows the GUI of the C++ vs the Vaadin web GUI.

In the second and third cases, we converted a GUI instance of a C++ application called PET (Parameter Editing Tool) into a Vaadin web application and use the data push approach with long_polling and websockets, respectively. In these two cases, the programs connect to control devices and display (or change) the live setting or measurement values of these devices. Two of the devices have data updating at various frequencies ranging from 1Hz to 1000Hz. In both cases, we found that the Vaadin web application works well when the data frequency development. It is recommended that websocket based push is used for long running web applications(>24 hours). Long_polling is supported by most of existing systems as it is an HTTP based "fake" push technology. Websocket is a relatively new technology, and requires new versions of application server software. Websockets are the recommended push technology for web based control applications.

In our Control System, we have successfully developed a DashBoard web application system using the Vaadin technology. It provides a flexible system for quickly setting up web based controls applications with a rich GUI to monitor accelerator operations.

Testing has uncovered some common problems with data push in Vaadin:

- After an application is running for some period of time, a "UIDetachedException" error occurs causing the web application to stop updating IO data. When this happens, re-loading page doesn't always help. The application server requires a restart. This problem may be caused by user session time outs.
- The Web UI seems unable to handle high frequency IO data with data update rates >

ISBN 978-3-95450-148-9

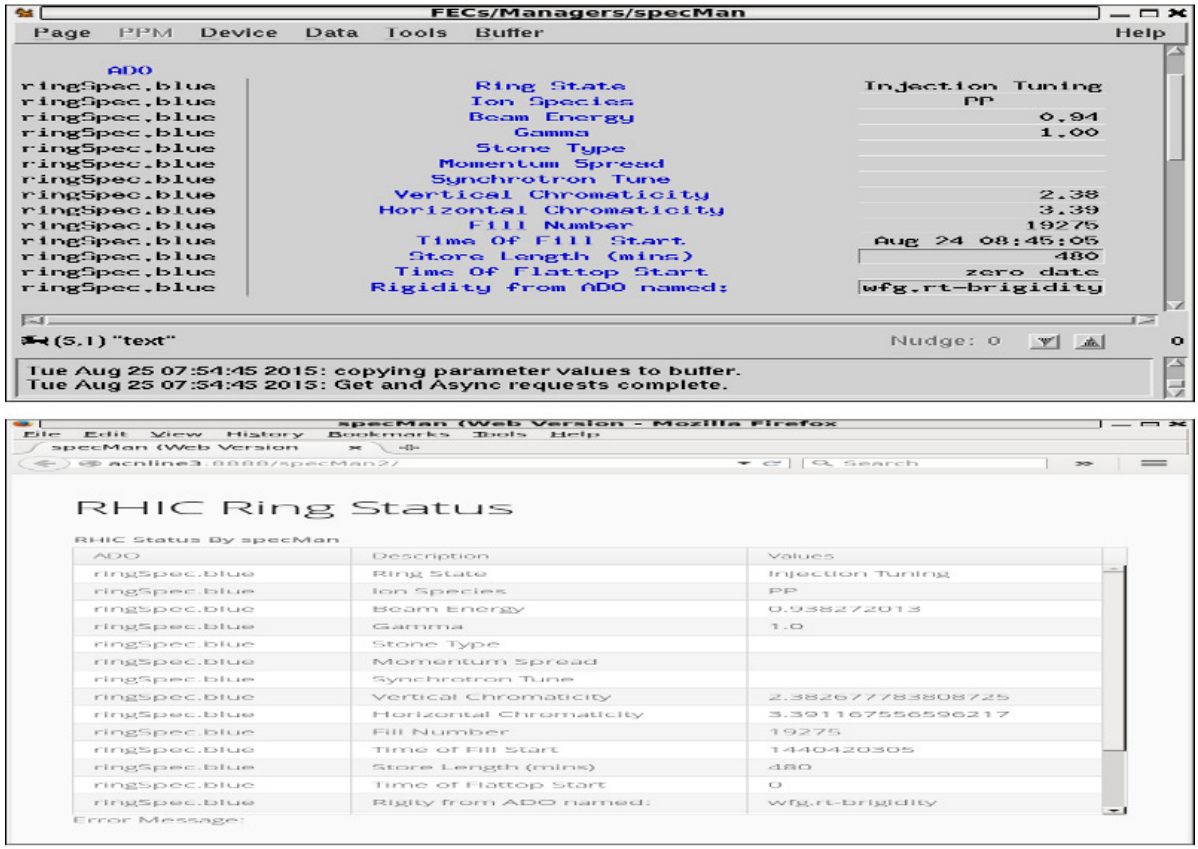

Figure 3: specMan: C++ (top) vs Vaadin (bottom)

~100Hz. Developers need to keep this in mind when deciding if Vaadin is good for the type of target applications which requires high frequency data updates.

- The IO connections between front end application and Vaadin application server, and the IO connections between Vaadin application server and the rest of control system have to be properly controlled and coordinated, otherwise, the control system may be over burdened.

Some of the problems can be resolved by using proper system settings and program logic controls; Others may require software updates. For example, after upgrading from Glassfish 4.0 to v4.1, the UIDetachedException problem disappeared.

It is noticed that, with Vaadin, we can develop web applications without explicit knowledge of HTML and Javascript. However, since all client side web applications are essentially based on HTML, Javascript and CSS, familiarity with HTML, Javascript and CSS will help developers make web applications more flexible.

SUMMARY

The Vaadin web application frame work offers web application developers rich sets of UI components[3], add-ons, Java APIs, and powerful console level application like features. It makes accelerator control web application development simpler for Java developers without the prerequisite of HTML and Javascript knowledge. The key aspects of accelerator control applications include a fast and responsive bi-directional IO connection and UI interactions on web GUIs. The Vaadin web framework does a good job in this regard with server push features that support long_polling or websockets. The testing of accelerator control web applications developed with Vaadin technology shows that, it is easy to convert console level control applications to HTML5 supported web platforms, and the web applications can be just as robust as the OS console level applications. As Vaadin technology and application server technology evolves, this web application framework will become more reliable. These technologies will aid in making web based accelerator control application development easy, powerful and more convenient to end users. The testing work with Vaadin showed some problems that needed attention during application development. Avoiding these pitfalls will help to develop robust and high performance web applications.

REFERENCES

[1] Book of Vaadin: https://vaadin.com/book/
[2] Wikipedia: https://en.wikipedia.org/wiki/Vaadin
[3] Vaadin: http://demo.vaadin.com/sampler/

ISBN 978-3-95450-148-9