ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUSH101

CREATING INTERACTIVE WEB PAGES FOR NON-PROGRAMMERS*

T. D'Ottavio†, P. Dyer, G. Marr, S. Nemesure, Brookhaven National Laboratory,
Upton, NY, USA

Abstract

This paper describes a new web page creation system that allows web developers with limited programming experience to create interactive displays of control system data. Web pages can be created that display live control system data updating in real-time, as well as data stored within our logging/archiving and database systems. Graphical, tabular, and textual displays are supported as well as standard interaction techniques via buttons, menus and tabs. The developer creates a web page using a custom web page builder. The builder presents a web page as a user-defined grid of tiled cells. The developer chooses the display style of each cell from a list of available cell types and then customizes its data content. Final polish can be applied using HTML and CSS. Specialized tools are available for creating mobile displays. This paper shows examples of the web pages created, and provides a summary of the experience of both the web developers and users.

INTRODUCTION

The Operations group within the Collider-Accelerator department at BNL has the job of setting up and running our many machines in a way that satisfies the physics experiments currently in progress. As part of that job, they need to communicate the status of the facilities to experimenters and internal machine specialists, as well as scientists around the world that are interested in our research.

Our Operations personnel are trained as physicists and generally have little or no web programming experience. So, they have had a difficult time over the years putting together web pages that provide user interaction and live displays. This paper describes work done to address this problem. The goal was to put together a web page construction tool that would let a developer create interactive and live updating web pages without having to understand web programming beyond some familiarity with HTML tags and CSS styling. A similar system was developed at DESY for building synoptic displays [1].

WEB PAGE BUILDER

The approach taken was to construct a web-based tool that lets the user create and view a web page in a way that closely matches the way that web page would be displayed. The user lays out the web page as a grid of tiled cells of varying sizes. Then the content of each cell is adjusted so that it either 1) displays data (label, table, chart, etc.), or 2) provides UI interaction (link, button, menu, etc.). The finished product is saved in an XML file. An example builder window is shown in Fig. 1.

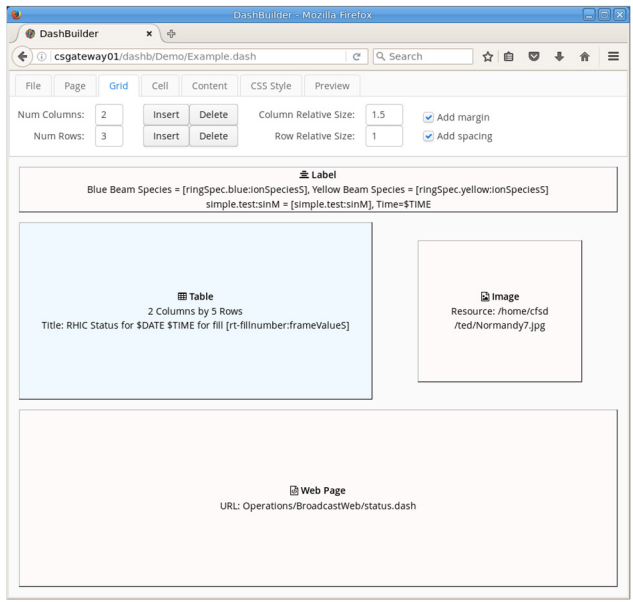

Figure 1: Web page construction interface

As seen above, the web page construction tool, which we call DashBuilder, is itself a web page that contains several tabs where the user defines various aspects of the web page that is to be built. Below is a summary of the functionality on each of the tabs:

- File – The web page description is stored in a file. This tab shows the file currently being edited and allows for saving, versioning, and switching the file.
- Page – Here a user indicates how frequently the web page should be updated (1-60 secs). The user can also specify the tabs to be shown and what should fill them. A theme can also be applied.
- Grid – This tab specifies the size (number of rows and columns) of the cell grid that should be used. Here the user can also specify the relative size of the selected row and column.
- Cell – Here a user defines the size of the selected cell by specifying the number of rows and columns that the cell spans. Adjustments can also be made in terms of width/height percentage and alignment.
- Content – The content of the selected cell is specified here. Choices include Label, Image, Video, Table, Link, Web Page, Slide, PPM Select, Time Select, Web Select, Tree Select, Pet Page, Gpm Monitor, Logged Data, FDA Plot. Once a choice is selected, the user can specify details, for example, the path to a Web Page or the contents of a Table.
- CSS Style – Final polish to the page can be specified on the CSS tab. Here, all the normal CSS styling rules apply. Each cell type has a known CSS class name, or the user can specify a CSS name for a cell.

* Work supported by Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy.
† Email address dottavio@bnl.gov

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUSH101

- Preview – When selected, this tab allows the user to view the web page as it would be viewed within a separate browser tab or window.

Cell Content

Much of the UI for the DashBuilder program is devoted to laying out the interface for the constructed web page. But it is the number and quality of the cell content types that are most important in determining the quality of the web pages produced. Broadly speaking, the cell types fall into three categories: user interactions, generic displays, and controls-specific displays.

The Link and the various Select cell types are examples of user interaction cells. In some cases the interaction affects other cells on the page. For example, a Web Select cell loads an associated Web Page cell. Generic display types include the Image, Video, and Web Page cell types.

The controls-specific cell types provide the most important functionality. They allow the user to display live or stored data from the control system within labels, tables and charts. We use a bracket notation, [controls name], to specify where a live device value should be inserted within the text of a Label or Table.

Most of the leverage for this builder, however, comes from reusing existing files and file formats that we have used for years to support our heavily used, generic applications. We have three programs that are used for generic controls display – 1) PET – displays live controls data in a tabular format, 2) GPM – displays live controls data in a graphical format, and 3) LogView – displays stored controls data in a graphical format. Each of these programs has a file format that specifies the specific data to load, and each has a large set of existing files generated over many years. Our web page builder is able to read these files and display them within the Pet Page, Gpm Monitor, and Logged Data cell types. This capability made it much easier to build a powerful web page construction tool.

Another important feature of this editor is that the constructed web pages can be nested. That is, one saved web page can be embedded inside another by specifying the path to the DashBuilder file within a Web Page cell description. This means that very complex web pages can be constructed using this design.

System Design

This section describes the infrastructure that is used to support both the construction and the display of the web pages produced by this system. As seen in Fig. 2, the system uses clients (web browsers) that talk to Java EE servers. We are currently using a Payara [2] server, which is based on a GlassFish [3] design.

A key to the system design is the use of web user interface tools provided by the Vaadin Toolkit [4]. This is a powerful set of tools that allows the developer to construct web user interfaces by writing code in Java that runs on the server. At runtime, the Java descriptions are translated into widget descriptions used by the Google Web Toolkit [5], which itself translates those descriptions into HTML and Javascript. Vaadin also supports a wide variety of add-on tools. We are using the chart and mobile add-ons. The basic Vaadin tools are free, but add-ons and support require purchase.

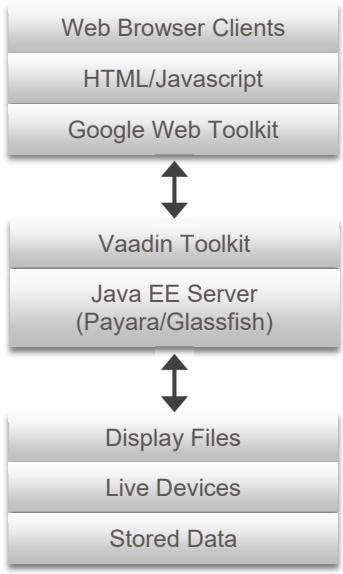

Figure 2: System diagram and tools.

In addition to the Vaadin tools, the system is supported by our own set of HTTP services, which are used for the retrieval of data from live devices, and data stored by our logging and database systems [6].

The server also needs access to the file system. To display a web page, the server is passed a relative path to a saved web page description file that contains information about the page layout and cells to display. And, as noted above, those cell descriptions often contain pointers to files created by other generic programs that describe table or graph displays.

VIEWING WEB PAGES

To display a web page constructed with this system, the user constructs an HTTP request that points to the saved web page description file. Fig. 3 shows an example.

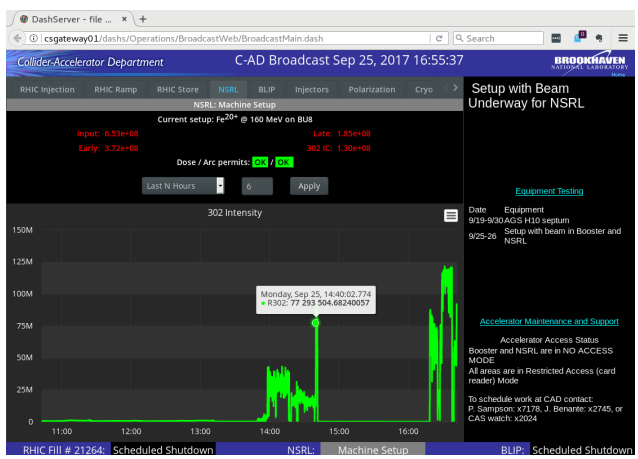

Figure 3: A displayed web page.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUSH101

A specialized web server, which we call DashServer, was setup to display the web pages described by the files created by the DashBuilder program. The same Java EE technology described in the previous section is used to display the web pages.

As can be seen from Fig. 3, web pages produced by this system can be very complex. This is made possible by the ability to nest saved web descriptions inside other descriptions. The example above actually uses 11 separate description files to create the web page. Most of those files describe what is to be viewed when the user selects one of the 8 tabs.

The DashServer supports continuous updating of Controls data displayed anywhere on a web page. This is accomplished by a client/server polling mechanism built into the Vaadin toolkit. The server updates its user interface as needed, and the client periodically polls for a refresh of the user interface. The polling frequency defaults to every 3 seconds, but the user can adjust this value within a range of 1 to 60 seconds.

MOBILE WEB PAGES

Over the last few years, there has been a lot of interest in developing status displays for smartphones. Much of the system as described allows for web page displays on modern smartphone web browsers. In addition, Vaadin offers a mobile toolkit containing specialized user interface widgets that work nicely on mobile displays.

This allows a user to build web pages targeted at mobile devices. We set up a specialized web server (DashMobile) that we use to display these mobile web pages. The developer builds pages targeted at desktop or mobile displays, and then directs users to one or the other of the servers to get the correct display.

Several screens from the mobile web application built by our Operations group are shown in Figs. 4 and 5.

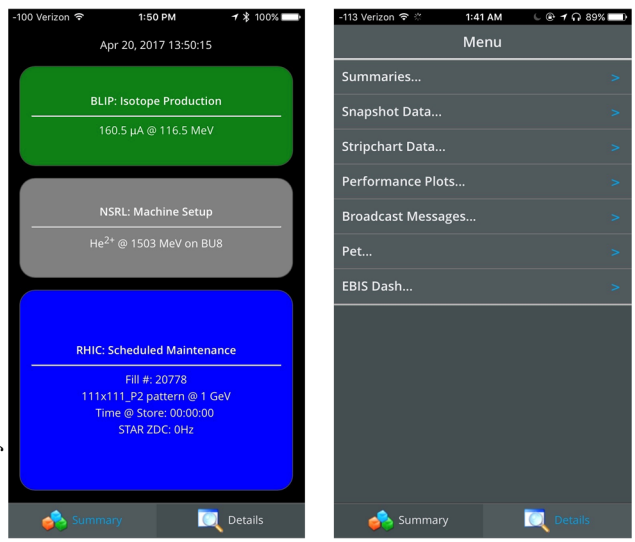

Figure 4: Mobile application start page and menu.

The start page for the application shows the status of the three major physics programs currently run within the accelerator complex. The background colors indicate the status of each program. Clicking on one of the colored areas will bring up a summary screen for that program. Even more detail can be obtained by clicking on the Details tab, which brings up the menu screen shown in Fig. 4. Fig. 5 demonstrates the quality of the tabular and graphical displays that can be produced by this system.

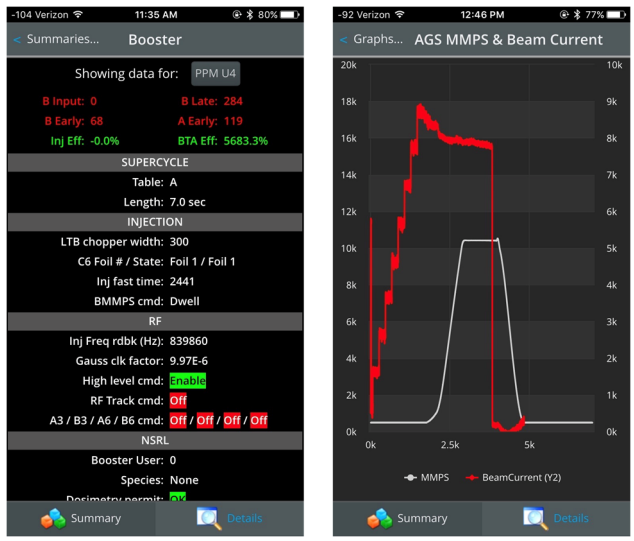

Figure 5: Mobile application tables and chart.

CONCLUSIONS AND FUTURE WORK

With a few month's effort, it is possible to put together a web page creation tool that allows non-programmers to create status displays that are interactive and complex. Leverage for us came from utilizing Vaadin and Java EE technology and from reusing tabular and graphical file formats that already existed within our facility.

Currently, the developed web pages do a good job of adjusting to various window sizes on desktop displays and generally do the right thing when the user adjusts a window size. Ultimately, however, we would like the web page developer to be able to build web pages whose content automatically adjusts to the platform where the content is being displayed – mobile or desktop.

REFERENCES

[1] Bacher, R. "Light-Weight Web-Based Control Applications with the Web2cToolkit" in *Proc. ICALEPCS'09*, Kobe, Japan, Oct. 2009, paper THP110, pp. 889-891.

[2] Wikipedia, GlassFish, https://en.wikipedia.org/wiki/GlassFish

[3] Payara, https://www.payara.fish

[4] Vaadin, https://vaadin.com

[5] Wikipedia, Google Web Toolkit, https://en.wikipedia.org/wiki/Google_Web_Toolkit

[6] D'Ottavio, T, Brown, K., Fernando, A, Nemesure, S, "Building Controls Applications Using HTTP Services" *ICALEPCS'17*, Barcelona, Spain, Oct. 2017, paper THUPHA157, this conference.