



Efficient development of web applications for remote participation using Ruby on Rails

M. Emoto^{a,*}, M. Yoshida^a, C. Iwata^a, S. Inagaki^b, Y. Nagayama^a

^a National Institute for Fusion Science, 322-6 Oroshi, Toki, Gifu 509-5292, Japan

^b Research Institute for Applied Mechanics, Kyushu University, 6-1 Kasuga-kouen, Kasuga, Fukuoka 816-8580, Japan

ARTICLE INFO

Article history:

Available online 5 May 2010

Keywords:

Web
Ruby on Rails
Web-based applications

ABSTRACT

Large-scale experiments such as ITER require international collaboration, and remote participation plays an important role in carrying out such experiments. Web-based applications are useful tools for scientists participating in experiments remotely using personal computers. Since the participants typically download web-based applications to their computer each time they access the web servers, they do not need to install extra software in order to use these applications. In addition, the application developers do not need to distribute the latest program files each time they are modified, thus reducing maintenance costs for remote participation systems.

For these reasons, we have been developing web-based applications for the LHD experiments at NIFS. In a previous study, we showed the benefits of using Ruby on Rails (RoR) to develop web-based applications for analysis code. We thought this approach would also be useful for developing applications for remote participation. Therefore, we have developed several web-based applications using RoR for participating in the LHD experiments. These applications include a data viewer and a scheduler of experiments. The main reason to adopt RoR for this purpose is its efficiency for developing web-based applications. For example, to develop a data viewer, we used an existing program running on an X-Windows System. Using RoR, we could minimize the modifications of the existing programs to add web interfaces.

In this paper, we will report a web-based application developed using RoR for the LHD experiment. We will also discuss the benefits of using RoR in developing remote participation tools.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

In order to accomplish a large-scale experiment such as ITER, the institutes involved in the experiment must be able to cooperate effectively. Therefore, many institutes have begun introducing remote participation systems [1,2]. Web-based applications are an ideal method for implementing remote participation systems for the following reasons: (1) since web browsers are an integral part of current operating systems, the users do not have to install extra software to use web-based applications; (2) web-based applications are downloaded each time the client accesses the server. For these reasons, when developers implement web-based applications, they do not need to distribute the applications to the remote users.

In a previous paper [3], we demonstrated that the AJAX technique [4] is suitable for building a web interface for simulation codes because this technique does not require extra software such

as Java Virtual machine or ActiveX components. In addition, we showed that RoR [5], an open source web application framework, is an efficient tool to build AJAX applications.

After developing this application using RoR, we thought this approach can also be used to build other tools for remote participation. One advantage of using RoR is that the programmers can develop the entire application using only RoR; therefore, they do not need to use other computer languages, such as JavaScript, C++, or HTML. Another advantage is that RoR is an efficient development tool. This efficiency is a result of its design philosophy of “Convention over Configuration” and “Don’t Repeat Yourself.” The former means that the developer should follow conventional rules. For example, RoR provides naming rules for the classes and tables of its relational database. Although the developers do not have to follow these rules, if they do follow the rules, then RoR automatically associates Ruby classes with database tables. This makes it unnecessary for the developers to write extra code to access the data in the database. “Don’t repeat yourself” means that information is stored in a single location. For example, in other programming languages, in order to change the definition of a table, a developer must change the definition in various locations, such as the SQL files, the C++ class definition files, and libraries. By contrast, if the developers

* Corresponding author. Tel.: +81 572 58 2242; fax: +81 572 58 2624.

E-mail addresses: emo@nifs.ac.jp, masahiko.emoto@nifty.com (M. Emoto).

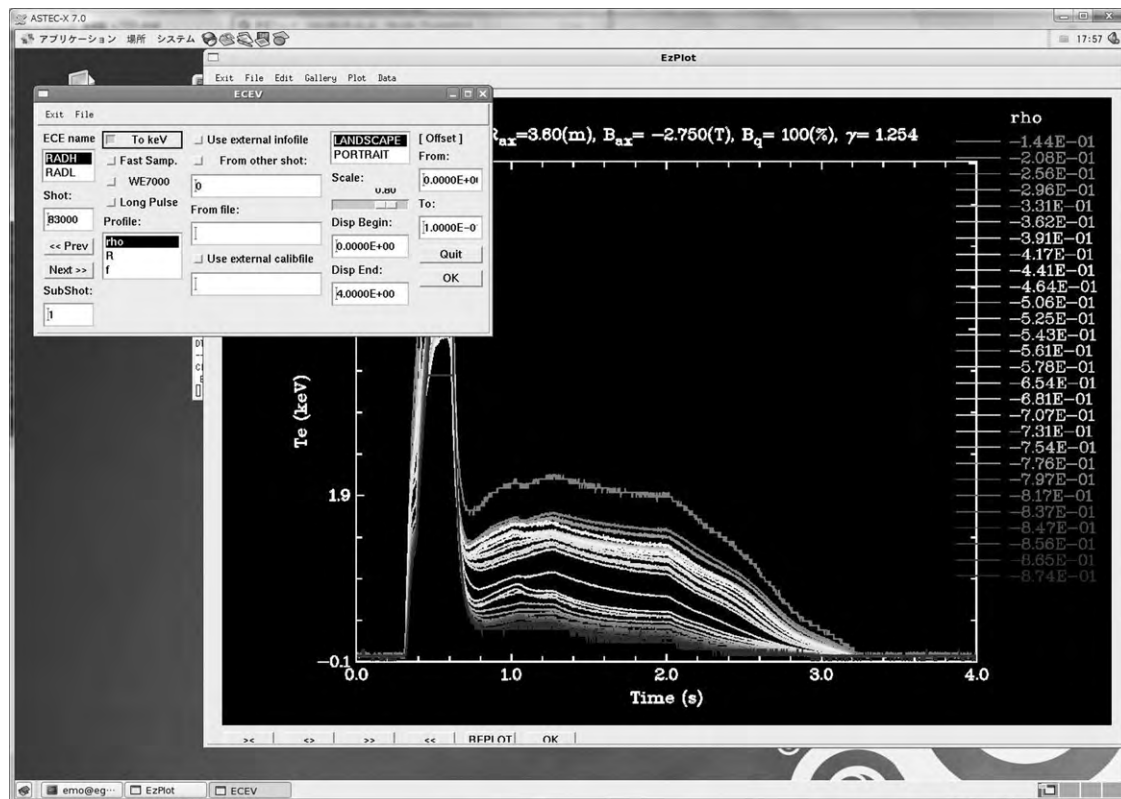


Fig. 1. ECEV, the data viewer that runs under X-Windows System.

use ActiveRecord of RoR, they only need to change a definition in one location, and the other parts are changed automatically.

2. Web-based applications using RoR

In this section, we introduce an actual web-based application that was developed using RoR for the LHD. This application enables the user to view raw signal data using a web browser.

2.1. Data viewer

For the LHD experiment, the raw signal data is acquired and archived by the LABCOM system [6]. In order to enable researchers to use the data archive of the LABCOM system, the developers provide the users with libraries and command line tools to register and retrieve the data. The scientists in NIFS use these libraries and tools to visualize the signal data. However, for those who are not familiar with the LHD experiment, such as visitors or co-researchers at other universities, it is difficult to use these libraries and tools. Therefore, user-friendly data visualization tools for the raw signal data are required. A visualization tool called ECEV (ECE Viewer) (Fig. 1) developed by Inagaki is available [7]. ECEV is an X-Windows based application running on Linux that provides a GUI to configure settings and draw graphs that can be viewed on the X-Windows system and can be stored as files. This application meets the basic requirements of visualizing the acquired data. However, it is a Linux-based application, whereas most of the researchers use other operating systems such as MS Windows and Mac OS. Therefore, we provide the users Linux servers in which ECEV and other necessary applications are installed, and the users can analyze the data without installing additional software on their computers. In order to use these applications remotely, the users have to login to the servers. For this purpose, NIFS provides virtual private network (VPN) access to enable the remote users to connect to the servers.

VPN is a flexible way to use remote computer resources. However, VPN access has two disadvantages: it is complicated to use for users who just want to see the latest results of the experiment, and it can cause security problems. Therefore, we decided to develop a data viewer. As mentioned previously, a web-based application is suitable for this purpose. In order to reduce the development time, we chose to provide a web interface for ECEV rather than developing a new application.

The basic scheme is as follows: ECEV has a function that reads a configuration file in order to configure drawing parameters, such as the diagnostics name, layouts, x-axis, and y-axis. The newly developed application, EGView, provides web interfaces to create a configuration file (Fig. 2). This is a RoR application that runs on an Apache web server. The user can edit the parameters interactively. EGView creates a temporary configuration file and runs ECEV. In order to plot graphs, ECEV uses a graphic library, DISLIN [8]. We changed the plotting device of DISLIN from an X-Windows System to PDF format in order to produce PDF files. EGView converts the PDF files into PNG files and notifies the web browser of the completion of drawing and the URL of the files. When the web browser receives the notification, it redraws the graphs. Because the above sequence is executed asynchronously, the web browser is not stalled while producing graphs, allowing for a smooth user interface.

One of the main reasons to adopt RoR is rapid GUI development. The asynchronous call of EGView is realized using AJAX. Generally, developing AJAX applications can become complicated because the web page is dynamically changed by the complex communication between the client and the server. However, the encapsulation of JavaScript in RoR simplifies the implementation of AJAX and enables the developer to easily add an interactive web interface to the existing application. Another advantage of RoR is database management. The user can customize the drawing parameters, such as target data, layout, and axes. The configuration data is stored in

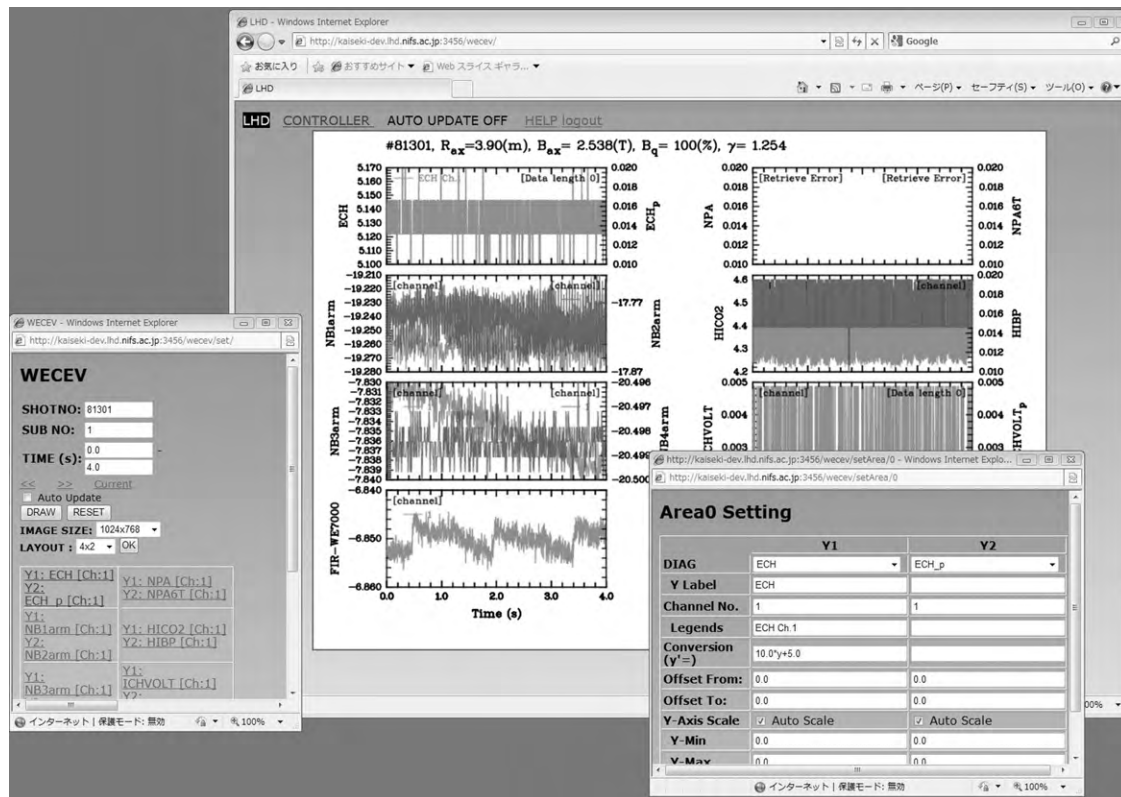


Fig. 2. EGView, the web interface of ECEV.

the relational database that is associated with the user account. Each time the user connects to the server, the server retrieves this information from the database and restores the configuration. User authentication is used to protect the settings from being modified by other users; the user information is also stored in the database. An additional advantage of using RoR is its ActiveRecord class. When using a relational database, it is normally necessary to convert variables into database objects. However, RoR provides the ActiveRecord class, which abstracts database functions such as SQL query and relations of tables. This feature enables the programmers to consider database objects as Ruby variables, thus making it easier to develop database applications.

3. Discussion and conclusion

In this paper, we showed the benefits of using RoR to build web-based applications for remote participation. As shown in this paper, we used RoR to add web interfaces to existing programs. This approach is an efficient method to quickly build a web-based application. While it is usually necessary to cooperate with the database in order to build a web-based application, Ruby's ActiveRecord class enables the programmers to relate to the database objects as Ruby variables. Therefore, RoR is suitable for developing the following types of web-based applications: (1) an application that uses relatively simple but dynamic GUI; (2) an application that needs to access databases. On the other hand, some types of web-based applications are not easy to develop using RoR. For example, an application that uses socket communication other than HTTP is difficult to develop because JavaScript does not support general network functions. Implementing complex GUI is another problem because HTML provides only simple GUI components. It is

possible to develop such an application using other software components, but it ruins the simplicity of RoR. We have developed a real-time data monitoring application. This is a web-based application, but it must receive IP multicast packets and have real-time plotting capability; these features are difficult to implement using RoR. Therefore, we chose to develop this application using the Java programming language. In conclusion, RoR is an efficient tool to develop certain types of web-based applications, but it is not always the best choice. The developer should choose other tools if they are more suitable.

Acknowledgments

We thank all the staffs of the LHD project for their assistance and encouragement. This work was supported by the NIFS budget code NIFS08ULHH512.

References

- [1] J. Vega, E. Sánchez, A. Portas, A. Pereira, A. Mollinedo, J.A. Muñoz, et al., Overview of the TJ-II remote participation system, *Fusion Eng. Des.* 81 (2006) 2045–2050.
- [2] Y. Nagayama, M. Emoto, H. Nakanishi, S. Sudo, S. Imazu, S. Inagaki, et al., Control, data acquisition, data analysis and remote participation in LHD, *Fusion Eng. Des.* 83 (2008) 170–175.
- [3] M. Emoto, S. Murakami, M. Yoshida, H. Funaba, Y. Nagayama, Web interface for plasma analysis codes, *Fusion Eng. Des.* 83 (2008) 453–457.
- [4] <http://www.adaptivepath.com/publications/essays/archives/000385.php>.
- [5] <http://www.rubyonrails.org/>.
- [6] H. Nakanishi, M. Ohsuna, M. Kojima, S. Imazu, M. Nonomura, Y. Nagayama, et al., Portability improvement of LABCOM data acquisition system for the next-generation fusion experiments, *Fusion Eng. Des.* 82 (2007) 1203–1209.
- [7] Private communication, not yet published.
- [8] H. Michels, The Data Plotting Software DISLIN: Version 10, Shaker Media GmbH, 2010.