ECLIPSE RCP ON THE WAY TO THE WEB

M. Clausen, J. Hatje, DESY, Hamburg, Germany
J. Rathlev, University of Hamburg, Germany
K. Meyer, C1 WPS GmbH, Hamburg, Germany

Abstract

Web-based applications are becoming an increasingly important part of the IT strategy for many organizations. Eclipse supports the development of web applications with its Rich Ajax Platform project and aims to further improve this support in the next major version of Eclipse, which is currently in development and planned for release in summer 2010. Using these technologies, users can run the same Eclipse application on different platforms. At work for instance operators could use the desktop environment to control the machine while at home they configure the alarm system in a browser or cellular phone. The applications for both platforms can be built from the same source code. This paper will give an overview of the Eclipse strategy to support applications on the web and its impact on existing Eclipse RCP applications like Control System Studio.

INTRODUCTION

For many users and developers, web applications are becoming increasingly important. While traditional, native applications have many advantages, the strength of web applications is that they can be accessed from anywhere, including from mobile devices, and are easier to deploy because they do not require installation on the client. Building web applications usually requires developers to learn a different set of skills than those required for the development of desktop applications because web applications use different technologies than desktop applications.

The Eclipse Rich Ajax Platform (RAP) is a framework developed by the Eclipse Foundation which can be used to develop web applications based on existing Eclipse technologies. RAP application development is similar to traditional Rich Client Platform (RCP) development, so developers can reuse their existing RCP knowledge. The promise of RAP is to enable the development of both web and rich client applications from a single code base with a high degree of code reuse.

Supporting the development of web applications is also an important goal for the next major version of the Eclipse platform, which is currently being developed in the E4 project.

For this paper, we have surveyed the RAP project as well as the E4 project. We describe how web applications can be developed using RAP, point out the major differences to traditional RCP development, and list the current limitations of RAP. We also give a short overview of the E4 project. Finally, we assess the impact of the Eclipse web efforts on existing RCP applications by using the Control System Studio as an example.

THE ECLIPSE RICH AJAX PLATFORM

The Eclipse Rich Ajax Platform (RAP) is an extension for the Eclipse Rich Client Platform (RCP) for creating plug-in-based, interactive web applications [1]. RAP reuses existing Eclipse concepts and technologies, such as the Standard Widget Toolkit (SWT) and JFace, as well as plug-ins, extensions and extension points, OSGi services, etc. The HTML and JavaScript parts of the web application are provided by RAP, so developers are not required to be familiar with those technologies.

Because developers who write RAP applications continue to use the same frameworks as they would use when writing a traditional RCP application, developing a RAP application on the surface looks very similar to developing an RCP application. However, there are still some important differences that developers should be aware of.

In this section, we describe the basic way in which RAP works and the most important differences to RCP.

User Interface Architectures

Because RAP is based on the same APIs as RCP, the architecture of a RAP application's user interface is structurally very similar to that of an RCP application. Figure 1 compares the architectures of RCP and RAP. In RCP, the user interface (SWT) is implemented based on the operating system's native UI widgets. RAP provides its own implementation of the SWT API, which implements user interface widgets based on web technologies running in the browser.

However, while there are few structural differences, there are major differences in behaviour. A traditional RCP application is started once per user (that is, each user runs his or her own instance). RAP applications, by contrast, are multi-session applications: the application is started only once on the server and can be accessed by multiple users simultaneously. Therefore, applications must be developed with multi-session support in order to work with RAP.

Multiple Sessions

The multi-session nature of RAP applications has several important consequences.

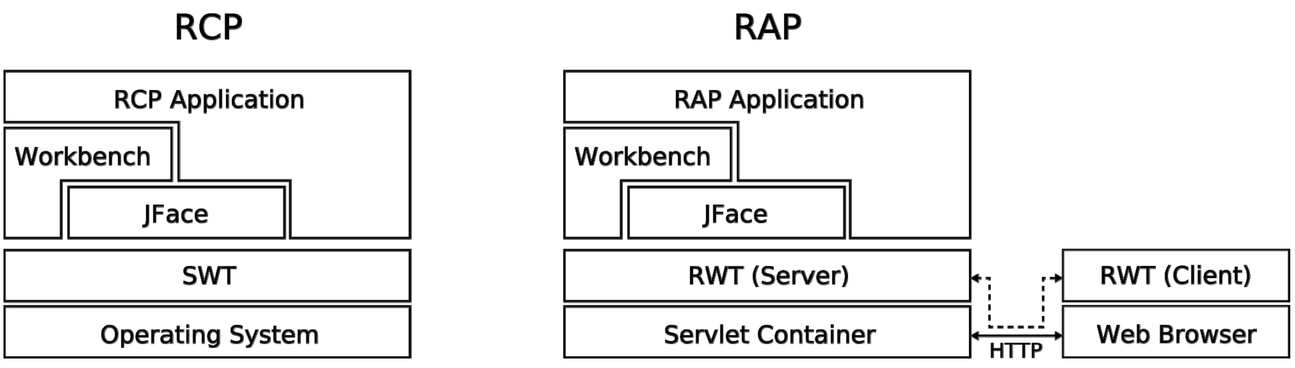

Figure 1: The user interface architectures of the Rich Client Platform and the Rich Ajax Platform. (Source: [2])

Obviously, the application's own application logic must support multiple sessions. For example, an application should not store user-specific state or session state in Singletons or use other kinds of "global" state like static variables. Singletons are instantiated only once per application, not once per user session, so they are necessarily shared across all users.

There are also some RCP features that behave differently when used in a RAP application. Most importantly, a RAP application has only one workspace which is shared by all users. Therefore, users cannot use the workspace to customize their application and all users of a RAP application share the same preference settings.

It is also not possible to use Eclipse's localization features to run the application with different languages for different users.

Communication and Performance

RAP uses the browser as a thin client which is responsible only for rendering the user interface. All application logic as well as handling of user interface events happens on the server.

The communication between the user interface in the web browser (client) and the application on the server is based on HTTP. The client uses JavaScript to send an asynchronous HTTP request to the application when the user interacts with the application, a technique known as Asynchronous JavaScript and XML (Ajax). This communication is handled automatically by RAP. Developers are not required to implement any communication between the browser and the server themselves.

Because all UI events are handled on the server, RAP sends an HTTP request to the server every time the user interacts with the web application. This requires a large amount of communication between the web browser and the server. RAP tries to minimize the network traffic by transferring only the changes to the user interface instead of the whole page. Despite this optimization, RAP is still very sensitive to network latency. Because every user interaction requires a corresponding network communication, RAP applications tend to become unresponsive if network latency is high [3].

Deployment

RAP-based applications can be deployed either as standalone server applications (based on the OSGi HTTP service) or as a Servlet-based web application running in a standard Java Enterprise Edition application server. On the client side, only a web browser is required.

LIMITATIONS

Most of the SWT, JFace and Eclipse Workbench user interface features can be used in RAP-based web applications. This includes perspectives, views, editors, dialogs, tool bars, context menus, and menu bars. However, one limitation is that RWT does not provide a graphics context object, so it is not possible to use code which calls drawing operations.

This means that custom SWT widgets which use primitive drawing operations cannot be directly used in a RAP application. To create custom widgets for RAP, knowledge of JavaScript is required because a JavaScript-based implementation of the widget must be provided to RWT.

RAP does also currently not support the Eclipse Graphical Editing Framework (GEF).

Another limitation is that RCP applications that are ported to the web using RAP will usually not have the look and feel which is typical for web applications. The style of interaction that is typical for the web is a page-based model, while RAP simply uses the windowing model used by the Eclipse Workbench. Whether or not this is seen as a disadvantage of course depends on the specific application and the expectations of the users.

THE E4 PROJECT

The E4 project is an Eclipse project which aims to develop the next major version of the Eclipse platform [4]. The project consists of many subprojects which deal with different aspects of improving the Eclipse platform. One of the goals of the E4 project is to improve the portability of Eclipse applications to different platforms, including the web.

Model-based UI

The rigid structure of the workbench is replaced by an EMF-based model which describes the layout of the user interface parts. Widgets are instantiated based on this model. This opens up the possibility to create not only SWT widgets but also other kinds of widgets, such as web-specific ones.

Additionally, the model will relax the structural limitations of the containment hierarchy. For example, it is no longer required to use perspectives. This might enable developers to design applications that have a look and feel which is more appropriate for the web.

Experimental Projects

There is an experimental project in the E4 context which evaluates compiling RCP applications from Java source code to Flex instead of Java byte code. The compiled application would then be able to run as a Flash application within a web browser. It is currently not clear which features of RCP would be supported in such applications.

IMPACT ON CONTROL SYSTEM STUDIO

Control System Studio [5] is a suite of many different applications. Based on our previously described analyses of RAP, we expect varying degrees of portability for the different applications.

Applications that make use of drawing operations to display for example charts or plots will likely not be portable to the web based on currently available Eclipse technologies. In particular, because the Graphical Editing Framework is not supported by RAP, we expect that it will not be possible to make the Synoptic Display Studio (SDS) available as a web application.

In RAP, a single workspace is shared by all users. This will impact all applications that use the workspace to store files or user-specific information. If those applications assume that access to the workspace is exclusive, they will have to be adapted to work with concurrent usage patterns.

CSS applications make extensive use of the Eclipse preferences service. Due to the shared workspace in RAP applications, preferences are also shared among all users. These applications will have to be adapted if different settings are required for different users or user groups.

We expect that simple diagnostic or monitoring tools which are based on standard SWT widgets can be ported to a RAP environment without requiring large changes.

CONCLUSION

The Eclipse Rich Ajax Platform is based on the same APIs as the Rich Client Platform. By using RAP, developers can leverage their existing RCP experience when developing web applications. Developers are not required to learn new frameworks or programming languages in order to use RAP productively.

There are, however, some conceptual differences between RAP and RCP applications that developers have to be aware of. The most important difference is that RAP applications run multiple user sessions within a single application instance. Therefore, in order to be portable between RCP and RAP, applications have to be developed with multi-session support in mind.

To ensure a high degree of portability when developing new rich client applications, our recommendation is to maintain a clear separation of the user interface and the application logic. Developers should also take care not to conflate user-specific data with global application data. We expect applications which follow these guidelines to be portable to the web without requiring a large amount of work, using either RAP or some other, future technology.

REFERENCES

[1] Eclipse Foundation, “Rich Ajax Platform (RAP)” (project web site); http://www.eclipse.org/rap/.

[2] Eclipse Foundation, “RAP Project – Introduction”; http://www.eclipse.org/rap/introduction.php.

[3] D. Rubel and M. Russell, “RAP or GWT – Which Java-Based AJAX Technology is for You?” (video), EclipseCon 2009; http://live.eclipse.org/node/722.

[4] K. McGuire and M. Oberhuber and S. Northover and J. Krause and B. Galbraith, “e4 Project in Review” (video), EclipseCon 2009; http://live.eclipse.org/node/737.

[5] “Control System Studio – CSS” (project web site); http://css.desy.de/.