



TECHNOLOGY NETWORK

DEVELOPER: Trends

As Published In

A Look at Rich Internet Applications

By Cameron O'Rourke



Looking at technologies for going beyond the aging HTML standard

For the last two years or so, interest has been building in the idea of a "rich client": a user interface that is more robust, responsive, and visually interesting than what can be achieved with HTML. Rich Internet Application (RIA) technologies allow us to deploy rich clients over the internet with Web-like simplicity. Whether RIAs will ever completely replace HTML applications is anyone's guess, but for organizations running complex applications with "fat client" technology, RIAs offer a low-cost alternative.

In this column, I sample current RIA products and technologies and provide some pointers for getting started. On the DevTrends Web site and in future issues of *Oracle Magazine*, I'll be delving into specific technologies and strategies for deploying RIAs, using the Oracle platform.

Why a Rich Internet Application?

HTML-based applications became popular because the cost of deployment was low, the architecture was simple, and HTML was trivially easy to learn and use. Many users and developers were willing to give up the user interface improvements brought by desktop computers in return for immediate access to new data and applications. The benefits of being Web-based outweighed the loss of significant UI functionality.

Certain applications, however, are not a good fit with HTML. Complex applications may require several page redraws to complete a transaction, and in certain fields such as medicine and finance, this can mean unacceptably slow interaction. Consider a project management system: It can be handled as an HTML application, but it certainly works better when users can see and manipulate charts, schedules, and hierarchies.

Furthermore, although HTML starts off simple, even simple interactivity can require a lot of scripting to get the job done. And even if an input form is carefully laid out and fully scripted, all it can send from the browser is simple name/value pairs. It would be nice if an HTML form could send and receive more-complex data structures in the form of XML documents.

RIAs utilize relatively robust client-side rendering engines that can present very dense, responsive, and graphically rich user interfaces. In addition to offering a wider variety of controls (sliders, date

pickers, windows, tabs, spinners, gauges, and so on), RIAs generally allow you to construct graphics on the fly with either Scalable Vector Graphics (SVG) or some other mechanism. Some RIA technologies can even provide full-motion animations in response to data changes.

Another benefit of RIAs is that data can be cached in the client, allowing a vastly more responsive user interface and fewer round trips to the server than with HTML. For wireless and occasionally connected devices, the trend is definitely toward rich clients and away from text-based Web clients. Applications running on laptops can be designed to work offline or to at least degrade gracefully when connectivity is lost.

The typical architecture for an RIA is shown in [Figure 1](#). XML is generally used as the data transfer format and is sometimes also used to describe form layouts. In many instances, the client can stay connected to the data source, so a server can update the client in real time. Access to an Oracle database is accomplished with Web service calls.

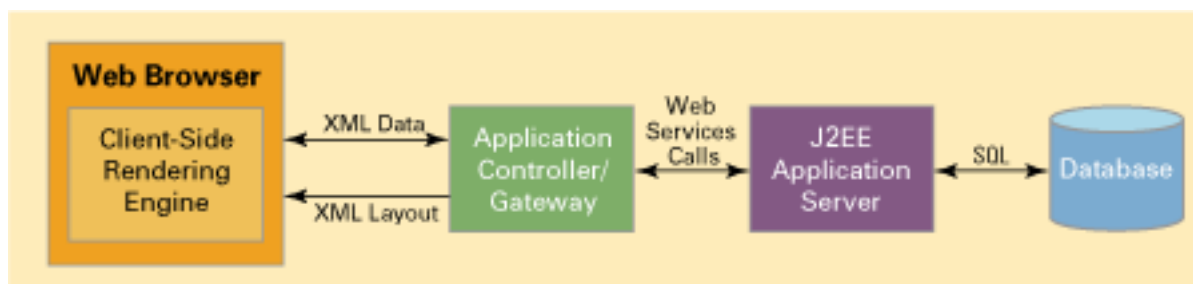


Figure 1: Typical RIA architecture

Technologies for Rich Clients

The following are some of the available RIA technologies:

Java. Some fairly complex client applications (Oracle JDeveloper, Eclipse) have been written in Java, so it stands to reason that Java can be used to create almost any rich client imaginable. Java is now several years old and has very comprehensive support for building form-based UIs. In addition to the user interface components found in the Java Foundation Classes (JFC/Swing), developers can also use the Standard Widget Toolkit (SWT) from the Eclipse Project and a raft of other third-party toolkits. For graphics there is the Java 2D API, a very complete—and very complicated—graphics API. Java also has unmatched support for XML and Web services. You can deploy applications by using either Java Plug-In software with a Web browser or with the newer Java Web Start technology included with the Java Runtime Environment. The main drawback of using Java for rich clients is its complexity (even simple forms and graphics require dozens of lines of nontrivial code). Advantages include Java's comprehensive support for Web standards and the depth of both the language and its class library.

XUL. XUL (pronounced "zool") is an XML-based UI language from the Mozilla open source project. It is used to create forms applications that run in the Mozilla browser as well as in other rendering engines such as Zulu (a Flash MX component) and **Thinlets** (a Java implementation). XUL rendering engines are typically very small (under 100K) and can both consume and produce XML

data. As with Java, there is a fairly large user community with plenty of open source tools such as [the Theodore Thinlet Editor](#) (see [Next Steps](#)), a Java application that lets you graphically lay out a UI and generate the corresponding XUL. XUL's main disadvantage is that it is not backed by a major commercial entity. XUL's big advantages are its integration with the Gecko engine (opening access to a huge array of Web standards) and the fact that XUL is a very expressive and compact language compared to most of the other XML UI description languages.

Macromedia Flash and Flex. Flash is a mature commercial product for bringing interactive graphics to Web pages. It has recently been upgraded to include features for building forms-style applications. Flash is arguably (depending on the Flash Player version you are referring to) the most widely deployed front-end technology on the Web, claiming up to 98 percent penetration across all desktops. The Flash facilities for creating animated graphics are powerful and visual (as opposed to requiring low-level graphics code). A graphics designer will be at home in this environment. The scripting language is ActionScript, a variant of ECMAScript 1.5, otherwise known as JavaScript. The Flex product adds an XML description language to Flash, so UIs can be compiled and rendered on the fly by the Flash Player. Flex makes Flash more familiar and accessible to traditional development shops. The primary disadvantages of both Flash and Flex are the limited support for XML and Web services standards and the relative immaturity of the environment as an application development tool. The primary advantages of Flash and Flex are the ease with which you can create complex animated displays and the availability of third-party add-ons.

Oracle Forms. Oracle Forms is a mature commercial product for building database-centric internet applications. With Oracle Forms, you create forms by using a visual designer that outputs form module files. The module files can be formatted in either the proprietary FMT format or as XML for further processing outside the design tool. The module files drive a Java runtime environment that renders the forms. In addition to all of the standard forms widgets, you can integrate additional Pluggable Java Components and custom JavaBeans for more functionality. The scripting language is PL/SQL, the same scripting language used in the Oracle database. An interesting feature of Oracle Forms is the Java API for creating, editing, and compiling Forms module files—developers can create scripts to generate numerous Forms applications or make global changes. Oracle Forms' main disadvantage is that it requires licensing of Oracle Application Server for Web deployment. Its primary advantages are its tight integration with the Oracle database and the rest of the Oracle platform (such as Single Sign-On and Enterprise Manager), its extensive support for internationalization, and its sheer productivity for building data-centric applications.

Getting Started

I've discussed only a representative sampling here of the technologies available for creating Rich

Next Steps

ASK Cameron

Technologist Cameron O'Rourke [answers your questions and provides additional developer resources.](#)

READ about

[Eclipse Project](#)

[Java Web Start](#)

[Mozilla XUL](#)

[Macromedia Flash](#)

[Macromedia Flex](#)

[Oracle Forms](#)

[Theodore Thinlet Editor](#)

[Thinlets](#)

Internet Applications. There are many others. When choosing an RIA technology, you need to weigh the following factors:

- Open source versus commercial product
- Mature functionality versus cutting-edge features
- Lightweight footprint versus UI richness
- Media-centric versus data-centric applications

Regardless of the technology you choose, I can offer these best practices for creating RIAs:

1. Fetch data in a background thread. The performance expectations for a rich client are high, and if the application pauses while collecting data from a Web service, it will seem unresponsive.
2. Keep the client in sync with the remote data. Because you are no longer constantly refreshing pages, it's important to "push" data changes to the client asynchronously, if possible.
3. Hire a graphic artist or at least a good UI designer. Along with the ability to create a visually interesting and functional UI comes the opportunity to make a complete mess of things.

Drop me an e-mail message, let me know how you plan to use Rich Internet Applications within your organization, and be sure to check devtrends.oracle.com for additional resources.

Cameron O'Rourke (cameron.orourke@oracle.com) has been in the IT industry since 1982 and is a 12-year Oracle veteran.

Please rate this document:

Excellent Good Average Below Average Poor

[Send us your comments](#)