

Technology Audit

Application Development

Canoo UltraLightClient 6.1

Written by: Michael Azoff

Date: October 2006

Abstract

Canoo UltraLightClient is a Java library for developing rich Web-based Java applications, also known as Rich Internet Applications (RIAs). An RIA differs from standard HTML or Dynamic HTML Web applications in that data is exchanged asynchronously through a channel which remains open throughout the user session. This allows optimised transmission of data, reducing network traffic by an order of magnitude, and thus speeding up the application. With RIAs the Web interface resembles a desktop interface and does away with the need to refresh the whole page when one item on the page needs refreshing. RIAs open up new opportunities for business applications, as well as allowing business to migrate traditional applications to the Web. Canoo's offering is unique in the marketplace for offering a pure Java solution, this reduces the learning curve for Java developers to write Web applications. UltraLightClient requires the Java Runtime Engine (JRE) to be present on the client side, whether as a browser plug-in or installed locally. This reduces the scope of the solution for purely Internet consumer applications. However, Canoo is targeted at business Internet and Intranet/Extranet applications, where JRE presence on the client-side can be assured. As a pure Java application, it offers robustness and strong security, making it also highly scalable. The product can be evaluated free for 30 days by downloading it from the Canoo Web site.

KEY FINDINGS

Key: ✓ Product Strength ✗ Product Weakness ⓘ Point of Information

| | | | |
|---|---|---|---|
| ✓ | Pure Java RIA solution based on Swing user interface library. | ✓ | Fast learning curve for Java developers. |
| ✓ | Highly scalable, suitable for enterprise-level business applications. | ✓ | Stand-alone mode simulates client/server interaction, suitable for testing and offline/online applications. |
| ✓ | Web applications can run within the browser or on the desktop as Java applications. | ⓘ | The licensing model is developer based, with no runtime restrictions. |
| ⓘ | A presentation layer RIA solution – the back-end relies on standard Java solutions. | ✗ | Not suitable for purely consumer Internet applications due to the need for JRE client-side. |

LOOK AHEAD

Canoo has recently offered a free open source tool to allow XML to be used against UltraLightClient. Soon to be released is a new version of UltraLightClient for mobile devices and thin-client terminals. Butler Group believes Canoo has strong prospects in an emerging market with many competitors.

► FUNCTIONALITY

Product Analysis

Developing Web-based applications is currently going through resurgence. Out of the post-millennium dot.com boom and bust have emerged genuine Internet enterprises that have exploited the Web and realised its possibilities. The point has been reached today where no business can afford *not* to have a Web presence, and where pure Web-based enterprises are worth billions.

The evolution of the Web can be characterised in terms of three phases: initially it was about putting textual content on the Web and joining these by hyperlinks. Then Web applications emerged that had database back-ends and interfaced with users through multi-page Web sites – the online shopping cart being a typical example. However, this second phase was still characterised by slow Web page refreshes every time the user clicked on a control widget. The browser always refreshed the entire page with each request-response transaction making optimised data transmission difficult. So a request for data never took account of the data already downloaded on the browser-side, ensuing in replication of network traffic, which further slows the application. This has all changed with the third phase: the facility to communicate between the browser and Web server asynchronously. In this third phase, the connection with the Web server is maintained open and while the user interacts with the interface on the browser, optimised data is exchanged asynchronously.

This third generation of Web applications is called Rich Internet Applications (RIAs) – it is characterised by their desktop-like features and look and feel. Demonstrating an RIA can be an anti-climax as it simply resembles an ordinary desktop application – its transmission via the Web is its significance, enabling advantages such as ease of maintenance and rollout, ease of administration and security (e.g. virus control), and of course the reach of the Internet.

One technology driving RIAs that has caught the attention of the developer community is Asynchronous JavaScript And XML (AJAX), the principle reason being its ubiquity as all modern browsers support JavaScript. Thus an AJAX-based RIA can be used by virtually anyone on the Internet. A JavaScript engine is downloaded into the client-side memory to run the session, its footprint is so small that this is not noticed. However, AJAX has its handicaps: coding in AJAX needs to support all the different Web browsers and their idiosyncrasies, although today this is mitigated by the availability of libraries that remove some of this burden. More serious limitations have to do with security and robustness under scaling.

An alternative approach is to adopt a pure Java solution. Java is always attractive from the viewpoint that it is platform independent and is designed for network applications. For a pure Java approach that offers full RIA functionality there is a solution from Canoo, based on the Swing Java graphics library. It does require that the Java Runtime Engine (JRE) be present on the client side, although this is the case for large numbers of Internet users, the main target for Canoo is business applications, where there is control over the client machine and where installing JREs is easily done. The JRE can be a plug-in to the browser, in which case the application runs within the browser, or installed on the client machine, in which case the application can run outside of the browser.

Canoo's UltraLightClient is a Java library that enables using the standard Swing User Interface (UI) components in server-side Java Enterprise Edition (Java EE) architecture. Swing is designed for full-fledged desktop applications and offers a rich level of functionality for user interfaces. Conventional Swing-based clients do not fit into a Web architecture so UltraLightClient bridges the gap between Swing's rich UI components and a server-side Web architecture, leveraging both the advantages of client-side Java and server-side application management.

Product Operation

Technically, UltraLightClient is a server-side proxy library for Swing that realises a client/server split within the presentation layer of an application (see Figure 1). The developer uses the proxy classes that offer the familiar API of the Swing classes, but with a server-side programming model. As a result, the developer is alleviated of the difficult task of splitting up the application into a client part and a server part. Moreover, the application executes entirely on the server except for a Presentation Engine that serves any number of applications, such as a browser.

A view of the deployment options is shown in Figure 2, emphasising how the architecture is integrated into standard Java EE: UltraLightClient deploys either as a servlet in a Web container, or as a stateful session bean in an Enterprise Java Beans (EJBs) container; the communication between client and server is configurable within the options permitted by the Java EE architecture; and the client-side Presentation Engine requires just a Java Virtual Machine (JVM), which means that it can run either as an Applet within a browser or as a full Java application on the desktop.



Figure 1: The UltraLightClient Architecture: High-level View

UltraLightClient provides a server-side complement for each client-side component enabling client-side UI controls to be used as if they are running on the server-side. Canoo describes this as Half Object and Protocol Design Pattern, where each client object is matched by a corresponding server one. For development purposes it is possible to run both client and server objects within the same JVM, making the execution and debug cycles shorter. In this stand-alone mode, the client/server interaction is simulated.

An application communicates with the Presentation Engine via any of the standard Java EE channels (HTTP, HTTPS, and Remote Method Invocation over Internet Inter-Orb Protocol); the choice of protocol is merely a matter of configuration. The UltraLightClient itself is a lean library with just 50,000 lines of code (about 550 KB) that affects only the presentation layer of an application; for all other layers, the architecture is entirely open.

One of the primary features of UltraLightClient is its scalability. There are two key characteristics that enable scalability:

1. **Clustering:** UltraLightClient supports standard Java EE clustering and runs on any J2EE platform, which means that scalability is truly simple and ranges very far (because the choice in J2EE platforms is wide).
2. **Optimised communication:** UltraLightClient's built-in split between client and server is highly optimized with respect to client/server interaction and communication. Typical applications generate less than 10% of the network load compared to HTML applications, and are sufficiently responsive over a slow modem line, for the following reasons:

- a. Configurable user actions like syntactic validation or enabling/disabling of controls can be executed locally on the client and do not trigger a server roundtrip.
- b. Communication can be configured to run either synchronously or asynchronously: the asynchronous mode leverages the standard HTTP request/response channel for its advantages, such as firewall penetration, by maintaining the channel open logically with a lean polling timer, push solution that is sufficient for most applications. A second asynchronous channel can be opened when real-time push is necessary.
- c. Data is loaded from the server to the client lazily, i.e. only the data that is visible is loaded, and through client-side caching, data that has been loaded once will not be loaded again when the user revisits it.

As UltraLightClient covers only the Presentation Layer it is not solely responsible for the scalability of the whole application, however, it is unlikely to be the limiting factor for scalability in any application.

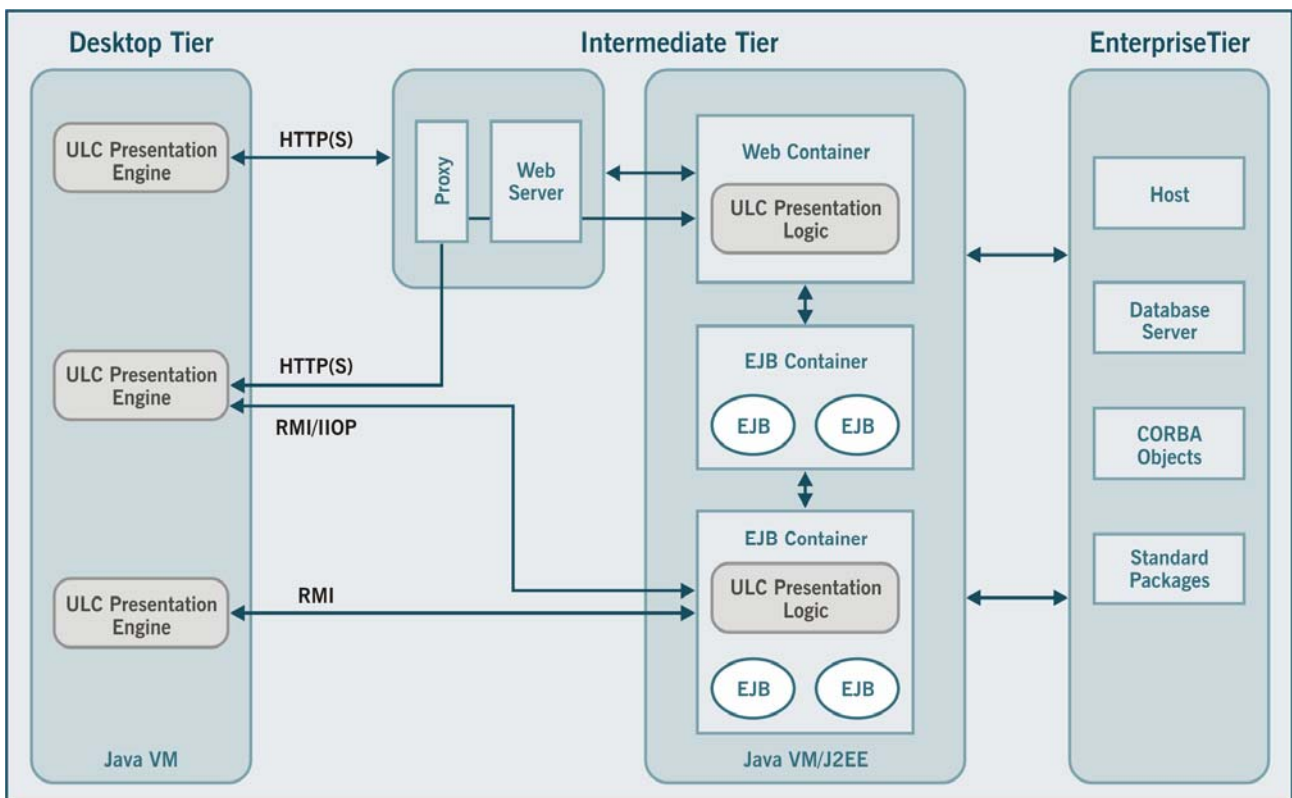


Figure 2: The UltraLightClient Architecture: Deployment View

For fault tolerance purposes the status of user sessions are held on the server, including the status of the complete user interface, so if the client crashes then the status of the user interface can be recovered from the server. Furthermore, the status of user sessions can be saved to disk on the server such that they can be recovered after a crash of the server. Since UltraLightClient supports application deployment as stateful session beans, the EJB container’s built-in support for ‘passivisation’ can be used for recovery purposes.

Product Emphasis

The key differentiator of UltraLightClient in the RIA market is that it relies solely on standard Java infrastructure, whether Java Standard Edition or Java EE. Competitor technologies are either non-Java (such as AJAX and Adobe Flash-based solutions) or they re-implement standard Java functionality.

Unlike AJAX, Canoo UltraLightClient is not restricted to the browser: it can operate as a fully-fledged Java application on the client machine, exploiting Java Web Start to initiate the application. The flexibility also extends to the server side, where the application can integrate with EJBs. In the stand-alone deployment, the entire application executes within one JVM – apart from development and testing benefits, this mode is also useful for running applications that need to be online and offline.

Furthermore, with AJAX the presentation appears as HTML or Dynamic HTML output in the browser, and uses the JavaScript language and Cascading Style Sheets. On the server side there is a need for a framework such as Struts to tie the Web application together. Thus developers need multiple skills to work with AJAX. However, with UltraLightClient, the Web application is developed as a pure Java application – this makes the learning curve for building RIAs much shorter for skilled Java developers.

The robustness of Java lends itself to business-to-business, whether Internet or Intranet, as well as business-to-consumer applications. AJAX applications can be difficult to scale up as the interaction between client and server is reliant on a number of technologies which do not optimise uniformly. There is also the benefit of a stronger security model with Java.

► DEPLOYMENT

The technical know-how necessary to implement UltraLightClient is good expertise in Java, up to Java EE infrastructure level. However, the level is lower than for normal Java-based Web development, because UltraLightClient resolves two essential issues that are difficult:

- Split of code between client and server, including object synchronisation and communication.
- Integration into the Java EE infrastructure. UltraLightClient offers abstract Application Programming Interfaces (APIs) that cover the low-level details and allow, for example, configurative deployment of an application as a servlet or a stateful session bean.

Projects teams are typically split into a framework team and several developer teams, where the former embeds the UltraLightClient library into a larger application framework that provides the complete infrastructure for development, including support for persistency, logging, and testing.

The implementation time depends on how UltraLightClient is being used. For a project that uses UltraLightClient “out-of-the-box” without customisations, implementation can be done in a matter of hours because UltraLightClient is just a Java library and can therefore be used in any Java development environment and production environment. If the Eclipse-based Visual Builder (ULC Visual Editor) or the load testing tool (ULC Load) is used, time for implementation takes a little longer; ULC Visual Editor and ULC Load are additional options to the UltraLightClient base product. However, much depends on the size of the application. ISVs who want to develop or migrate a sophisticated product to UltraLightClient or corporations who want to roll out UltraLightClient to lots of developers will need longer implementation times. In these scenarios UltraLightClient is often integrated into a larger framework with add-on components.

Most customers of UltraLightClient, up to 80%, do not require any training from Canoo. The product essentially offers a server-side API to Swing, so this flattens the learning curve. Canoo provides extensive documentation, a tutorial, a programmer’s guide, and numerous code samples, and its email-based technical support is usually sufficient for experienced Java developers.

The following courses are available from Canoo:

- Introduction to UltraLightClient (2 days).
- Programming with the ULC Visual Editor (1 day).
- Optimising ULC Applications (1 day).
- In-depth training on Tables, Trees, and TableTrees (1 day).
- Developing a ULC application from end-to-end (1 day).

Canoo also provides consulting and customised courses and on-site training. Ongoing technical support is provided through:

- A free e-mailing list.
- Paid e-mail support with guaranteed response times and prioritised handling of requests.
- Customers can hire Canoo consultants or partners for on-site support.

The pricing of UltraLightClient is as follows:

Standard:

- UltraLightClient library: US\$ 1499 per developer seat.

Optional:

- ULC Visual Editor: US\$ 499 per developer seat.
- ULC Load: US\$ 2999.
- Source Code: US\$ 50,000.

With the developer licenses the customer can develop and deploy software indefinitely, as long as each developer has a license. A yearly maintenance fee of 20% entitles customers to upgrades. Customers who have not purchased maintenance can buy upgrades for a fee of 60%. The support model will soon be changed to a flat-rate model with “platinum service”, “gold service”, etc.

► PRODUCT STRATEGY

The target for UltraLightClient is horizontal, Canoo has customers who are developing software for financial institutions, healthcare, logistics, government, public transport, manufacturing, and others. Return on investment in using UltraLightClient is gained through an improved Web application architecture. Many businesses rely on the Web for revenue and UltraLightClient offers a high degree of security and reliability compared with alternative RIA approaches.

The target market by size for Canoo divides into three categories: Large enterprises, ISVs, and IT service providers. Key market opportunities exist with ISVs who need to migrate their applications to RIA technology because their customers expect applications to work over the Internet, and companies who want to move their applications to RIA technology, because they want to use the same applications in the Intranet and Extranet both online and offline.

Canoo's route to market is to sell direct and through partners. Key business partnerships that support the product include Genigraph (France), C1 WPS and Innexu (Germany), Tata Consultancy Services, and Tinext (Switzerland).

Canoo issues one to two major releases per year, and one to four maintenance releases in between.

► COMPANY PROFILE

Canoo, a privately-owned company, is headquartered in Kirschgartenstrasse 7, 4051 Basel, Switzerland, with additional offices in Lugano (Switzerland), Mumbai (India), and Amsterdam (The Netherlands). The company was founded in 1999 by 18 senior engineers who had worked together in various contexts.

Canoo has currently 33 employees, and 5 subcontractors. One employee is located in India, one in Amsterdam, one in Lugano and the rest are located in Basel, Switzerland. The personnel split by category is: Research & Development – 8, Sales & Marketing – 4, Support & Services – 24, and Administration – 2.

Examples of key clients include: UBS, Navis, Farm Credit Canada, Otto, Münchener Verein Versicherungen, Abacus, Siemens, Dekabank, Credit Suisse, and Würth. The implementation at UBS covers the entire Swiss IT organisation, comprising hundreds of developers, three deployed applications, and many more are being developed, with thousands of users. At Würth the deployment is global, with 60 developers, thousands of users, and the company has commissioned offshore software shops with development. Overall, dozens of applications, many of which serve thousands of users, are deployed world-wide.

Currently, Canoo has 106 customers with purchased licenses for UltraLightClient. In addition, there are about a dozen universities that have free educational licenses. The company's total customer base is about 120 paying customers. In addition, there are thousands of non-paying customers who are using the open source WebTest product, and thousands of non-paying customers who are using the free on-line service –Canoo.net, which has about 30,000 visits per day.

► SUMMARY

RIAs open a new possibility for business. UltraLightClient is frequently used to consolidate applications that run across the Intranet and the Internet with just a single application. UltraLightClient also creates multi-purpose applications by leveraging its multiple deployment options: Intranet for employees, Extranet for contractors or customers, and online/offline usage over the Internet. The product also allows business to move traditional client/server applications to the Internet: fat-client applications can be migrated to UltraLightClient and offered in “on-demand” scenarios over the Internet, an option favoured by ISVs. The possibilities for RIAs are still being explored and reflect the emerging nature of this market.

Butler Group believes that Canoo has a distinct position in the RIA market in being able to offer a pure Java solution, a major attraction to organisations already invested in Java. The robust features of Java also make it ideal for enterprise applications. For business-to-business, as well as other RIA scenarios, Butler Group believes Canoo UltraLightClient should be considered.

Contact Details

Canoo Engineering AG

Kirschgartenstrasse 7
CH 4051 Basel
Switzerland

Tel: +41 (61) 228 94 44

Fax: +41 (61) 228 94 49

E-mail: ulc-sales@canoo.com

www.canoo.com



Headquarters:

Europa House,
184 Ferensway,
Hull, East Yorkshire,
HU1 3UT, UK

Tel: +44 (0)1482 586149

Fax: +44 (0)1482 323577

Australian Sales Office:

Butler Direct Pty Ltd.,
Level 46, Citigroup Building,
2 Park Street, Sydney,
NSW, 2000, Australia

Tel: + 61 (02) 8705 6960

Fax: + 61 (02) 8705 6961

End-user Sales Office (USA): Important Notice

Butler Group,
245 Fifth Avenue, 4th Floor,
New York, NY 10016,
USA

Tel: +1 212 652 5302

Fax: +1 212 202 4684

This report contains data and information up-to-date and correct to the best of our knowledge at the time of preparation. The data and information comes from a variety of sources outside our direct control, therefore Butler Direct Limited cannot give any guarantees relating to the content of this report. Ultimate responsibility for all interpretations of, and use of, data, information and commentary in this report remains with you. Butler Direct Limited will not be liable for any interpretations or decisions made by you.

For more information on Butler Group's Subscription Services please contact one of the local offices above.