Web Enabling your Native Apps
With the WebKit Rendering Engine

Abstract

If you are leading a software development project today, you probably start with one very basic choice – can I make this work on the web? More and more, the default answer tends to be "yes". Find out how Trolltech has made it easy for developers to implement web content directly into their native applciations through the integration of the WebKit rendering engine.

Table of Contents

# Introduction: Build the web into your software (and vice versa)

At the outset of every project a software developer asks a very basic question. Can I make this work on the web? In a technology world dominated by digital audio and video, social networks and a complex web of interconnected devices, the answer most always is "yes." Platforms and applications that are connected make your products accessible to consumers and lower the cost of pushing out updates. Unfortunately, web applications are burdened with a long list of performance and security trade-offs that come with having to work inside a browser environment.

Now, with Qt 4.4, many of these decisions and trade-offs have become a thing of the past. You can create software that takes full advantage of the native desktop or device environment, and also include an embedded browser engine that allows you to mix and match content from both worlds. Qt 4.4, now with WebKit embedded, means you can create rich web applications on top of the Qt cross-platform user interface library for desktops or Qtopia embedded Linux platform for networked devices.

One of the best known examples of this best-of-both-worlds approach is Apple's iTunes client software. The rich, native application incorporates an embedded web client that allows customers to discover and download from the iTunes music store, in combination with a desktop client that organizes play lists songs and handles thorny issues like digital rights management. The embedded browser in iTunes is also based on WebKit, the open source embeddable browser derived from early work at Trolltech (see sidebar: "A Little History").

For the purposes of this paper, however, the important point is that Apple refined WebKit and made it the core of its Safari browser. Along the way, the company improved support for key web standards so that WebKit handles HTML, Cascading Style Sheets, and JavaScript. This leveled the playing field, putting WebKit and Safari on par with Firefox and Internet Explorer. Trolltech, in turn, has actively supported the WebKit open source project, helping improve its portability across operating systems. With Qt WebKit, we have made it easy to add and integrate with WebKit as a component of any application built with our user interface development frameworks and tools.

The integration between native user interface and web content actually runs both ways. In the example shown below, the web page rendered within the Qt WebKit browser window includes an image retrieved from the web, along with a number of standard HTML controls, but the slider shown at the bottom of the page is a native Qt widget. Manipulating the slider also invokes native image manipulation functionality for controlling the brightness of the image shown on the page.
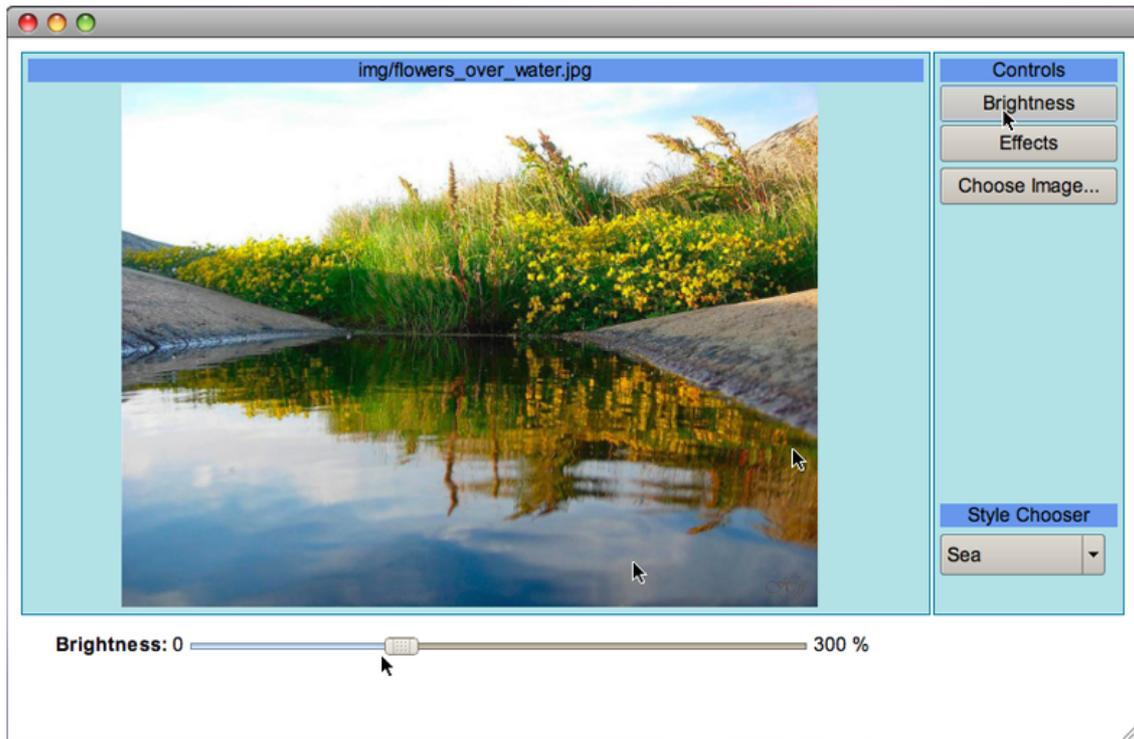
How is this possible? The same HTML <object> tag used in other browsers to represent content that should be displayed with the help of a plug-in is used by Qt WebKit for placement of native software components. The markup below would place the native widget referenced with the "EXAMPLE" class identifier within a web document at the location of this tag.

```
<OBJECT type="application/x-qt-plugin" classid="EXAMPLE" width="100" height="50"/>
```

Using a JavaScript API and the Web's Document Object Model, developers can manipulate the visibility and placement of this object just like any other element on the page (for example, to show or hide a user interface widget). JavaScript code also can be used to invoke functions that native software components will execute.

Could the same effect be achieved entirely with JavaScript and AJAX?

We should never say never, given the image handling routines engineers at operations like Flickr have managed to build into their Web 2.0 applications. But clearly there comes a time when you're packing in too much into JavaScript code that must be downloaded and interpreted on the fly, particularly if you have the choice of substituting fast compiled software.

Your native software application also can invoke functions within the web environment, for example to navigate to a specific page. In this way, a mobile phone operated on a pre-paid calling plan could detect a low balance condition and automatically send the user to an e-commerce site offering the ability to re-charge the account by credit card in a secure environment.

The code your developer would have to write for this purpose could be as simple as:

```
QWebView webView;
webView.load("http://www.host.com/");
webView.show();
```

This paper is not intended to give detailed coding instructions, just to give you an idea of what is possible. Still, the code to send application messages back and forth between JavaScript and native code is only slightly more complicated than what we've shown here. Qt WebKit methods can be invoked from either C++ or Qt Script for Applications.

Other reasons to use Qt WebKit:

- Simplify testing by using *one* browser engine across multiple platforms. When you embed Qt WebKit in your application, you have the luxury of dictating a single browser environment for all users. This eliminates much of the typical web application development hassles associated with CSS or JavaScript code that gets interpreted differently by different browsers, and versions of browsers, on different operating systems.

- Give your Internet application access to resources that are outside of the web environment "sandbox," such as access to the underlying file system, and make your own independent decisions on how to ensure application security (just make sure you get it right).

- Tweak the browser environment to behave however you want it to work, whether that means adding custom media types and other new capabilities or restricting navigation and the content types you want displayed.

- Integrate Web Apps into an application, such as weather, stock data, or some of the more popular social networking sites like Facebook, etc.

Putting Qt WebKit to Work

What can you do with Qt WebKit?

Create an iTunes-like experience. Partly because of the history with WebKit in iTunes, one obvious application is a similar online store for downloadable media – not just music but video, games, or anything else that can be distributed as bits. In this way, you can combine a native widgets for your media player and utilities for storing and organizing files with a web content display panel to manage the shopping process and show all the latest promotions.

Embed reporting, documentation, and help functions. If your application must display data, documentation, or help screens, Qt WebKit provides a convenient method for displaying text, graphics, and multimedia content that takes advantage of web standards. Qt also provides the QtextBrowser for programmatic formatting of rich text for display within your application and also includes some web-like hypertext capabilities. But in many cases, building your application to display standard web content is easier.

Enhance virtual worlds. While networked games, training, and social avatar environments like Second Life have dramatized the power of 3D worlds to let us do things we couldn't do in a 2D environment, there are also tasks for which an embedded browser is the more powerful and practical solution. Examples include subscription renewal or displaying player rosters and lists of events for your avatar to attend.

Turbo-charge your mobile browser. By using Qt WebKit in combination with the Qtopia environment for Linux-based devices, you can deliver a web browsing capability that makes intelligent use of the native capabilities of the device. For example, the "home screen" display when you activate you phone might be customized by location – taking advantage of built-in GPS capabilities – to conveniently connect you with local services. If you take a photo with the built-in camera, a brower-based display could guide you through the steps to post it to a photo-sharing service on the web.

These capabilities can be built into other mobile devices as well as phones. For example, a camera with Wi-Fi or cellular network access could include its own browser for posting photos to the web or sending them to a friend's camera phone.

A Little History

WebKit came into being as a fork, or derivative, of the KHTML browser engine created for the KDE desktop used on Linux and Unix systems and used as the core of the Konqueror web browser. Our embrace of WebKit is a sort of homecoming for this codebase. Trolltech employs a number of the key developers that have worked on KHTML, including project founder Lars Knoll.

In 2002, Apple adapted KHTML and the related KJS JavaScript interpreter for use in the Safari web browser. KHTML was published under the LGPL open source software license (a more lenient version of the Free Software Foundation's Gnu Public License), which required Apple to share its modified version of the codebase, which Apple called WebKit. WebKit became the core of the Safari browser, but it also became an important embedded component of other applications such as iTunes and entered the mobile realm on Apple's iPhone smart phone.

Because Apple invested significant resources in improving the handling of HTML, Cascading Style Sheets, JavaScript and other web content, WebKit has emerged as an important technology for embedding web browsing and web application capabilities in all sorts of software. It's being used in smart phones from Nokia (our parent company, as of January 2008) and Google's Android mobile software platform, and in the Adobe AIR rich Internet application framework.

Trolltech is an active participant in the WebKit open source project, helping make the code more portable. So the browser engine that started on Linux and moved to MacOS now runs on Windows and many other desktop platforms, as well as phones and embedded devices. Of course, open source projects have many parents, but we still think our role in giving birth to this technology gives us an edge in understanding how to use it to the fullest. By working with Trolltech, you can have that edge, too.

Create store kiosks and other embedded environments. Many stores now offer an in-store version of their online store as an alternative way of browsing for merchandise. Often, this is just a matter of providing a PC for customer access, setting Internet Explorer for full screen display, and locking it on the store website. But a store kiosk based on an embedded environment with Qt WebKit could take advantage of custom features, such as the ability to tie in with card readers and bar code scanners. A consumer could scan the barcode off a shirt to find the product in the online catalog, browse to find a color or size not available in the store, and then complete the order with the swipe of a credit card.

Again, we're offering the store kiosk as just one example of the type of embedded environment where Qt WebKit could prove valuable. Just as easily, you could provide documentation or performance measurement in the control system for a complex piece of equipment on a factory floor.

Tap web services. In addition to displaying Web content, Qt WebKit lets you tap into any sort of services that can be accessed using web protocols, XML, JavaScript/AJAX services invocation, or other methods. Create your own web services or tap into public ones like Google Maps, Yahoo Mail, or RSS syndication.

Link to social networks. By taking advantage of the application programming interfaces offered by the likes of Facebook, LinkedIn, and MySpace, you can make your own applications more social. For example, in the context of mobile phone development, you can offer your phone's users with a way of keeping up with their Facebook friends while they're on the move.

Next Steps

By now, we hope you have a sense of what you can accomplish with Qt WebKit – just about anything you can imagine that takes advantage of some combination of the web and native software. Like the other user interface components in the Qt family, Qt WebKit is portable to all major desktop operating systems and to the Qtopia Linux environment for mobile devices. We provide it to you as open source software, with services and support available from Trolltech.

Find out more at http://trolltech.com/webkit/

About Qt

Qt is a cross-platform application framework for desktop and embedded development. It includes an intuitive API and a rich C++ class library, integrated tools for GUI development and internationalization, and support for Java™ and C++ development. Qt enables companies to build software applications that run across multiple desktop operating systems and embedded devices, without rewriting the source code. For companies from a broad spectrum of industries, Qt shortens time-to-market and increases productivity by allowing them to leverage software investments made on one platform across many others.

For more information, visit **www.trolltech.com**