VISTA Community Meeting
Day One: January 14, 2011
Afternoon Session

Topic: EWD

Presenter: Rob Tweed, M/Gateway

Mobile computing is on the rise. This year, it is predicted to overtake desktop/laptop for web interface. Part of why this happened was the app store, which has been imitated elsewhere. Result: everyone wants mobile apps.

Problem: you can't just take an application designed for a desktop machine and slap it onto a mobile device. It must be re-written from scratch. When writing for the iPhone and similar devices, programmers essentially have two choices: native apps or web apps.

When the iPhone first came out, the idea was that all apps would run through the browser, but Apple changed their mind and native apps became the way to go. Native apps, however, have downsides. Objective-C, though no longer actually required, is still needed. To get into the app store, an application must follow Apple's rules. Only registered, subscribing developers can create native apps. Results: long development times, increased cost.

Nobody really knows how the app store makes its decisions, so a programmer could spend a substantial amount of time developing a native app, only to have it rejected by the app store, essentially depriving it of any market.

Because of these limitations, web apps are making a comeback. They can be made to look and feel like native apps, but are easier to develop. No mac is needed to develop, and there is no need to be approved by the app store. Frameworks have begun to emerge: iUI, iWebkit, jQTouch (derived from jQuery).

However, web apps have their own downsides. They cannot access the device's camera or GPS; iUI and iWebkit are limited to CSS-based styling and animations. They are dependent on a floating viewpoint, and can't have separately scrolling panels or fixed toolbars;. Also, apps developed using these frameworks will only work on the iPhone.

Sencha Touch was released in June 2010. It works on all webkit-based mobile devices, and is now available free. It supports all HTML5 devices, including iPhone and Android.

However, Sencha Touch uses Javascript programming, which poses its own problems to developers. Javascript is complex, which means it is costly to maintain. Developers have different ways of doing things, and if a new developer doesn't understand the old code, he's likely to re-write the entire thing rather than try to figure out how to make changes.

This, therefore, is the dilemma: everybody wants mobile apps, but native apps take too long to develop and are costly to maintain, while web apps are either too simplistic or contain lots of Javascript.

Is it possible to have the best of both worlds? This is what EWD tries to provide. Coding is kept to a minimum, so programmers can focus on the “what” rather than the “how.” Everything else is tag-based.

EWD acts as an overarching framework over other frameworks (mostly Javascript). The idea is to minimize coding and automate everything that can be automated. All code consists of fetching data and putting it in the session; then moving data from the session back into the database. A "session" is simply a type of data structure.

On the front end, EWD automatically generates most of the javascript; the programmer only needs to set up event handlers. The front end is set up using EWD's custom tags. They can be "layered" so that tags can refer to other tags, which refer to other tags, and so on. The tags are parsed through XML DOM manipulation. Because EWD code is much simpler, it is easier and less costly to maintain.

Development using EWD is very quick; a developer at this conference wrote an EWD program in a day, including the time it took to learn the language.

EWD design is based around what Mr. Tweed calls "fragments," which can be thought of as small elements of the picture to be presented to users. A large UI can be broken down into several small fragments, each of which can be assigned to a different developer.

Events retrieve fragments and insert them into the DOM. This is secure and fast, since no one fragment is large or complex. Using fragments, developers can create as complex a UI as they want.

Mr. Tweed pointed out that VISTA programmers have a tendency to want to "roll their own;" to figure out what they want done and write the underlying code themselves. In Mr. Tweed's opinion, this is the wrong approach for mobile applications. Mobile devices have limited capacity and rely entirely on Javascript. Only seasoned Javascript programmers need apply—and most of them already work for Yahoo or Sencha or somebody.

Q: Where could EWD best be rolled out in VISTA now? CPRS? What are people suggesting?
A: I'm not a VISTA expert; you're asking the wrong guy. However, we don't want to be forced into a "big bang" situation where we're required to do a big rollout. But there's a middle-tier solution called node.js, a server-side Javascript. In combination with websockets, node.js allows us to have a VISTA implementation running at one end, and a Javascript-based VT terminal emulation in the browser. It's in the browser, so it's totally seamless. It doesn't use klunky techniques like polling; it writes things directly. So, we should get the people who know the business of VISTA to say which are the really high priorities and move those over to mobile. And instead of a big bang, we can phase it in seamlessly, and use the emulator for the bits that haven't been converted.

iWD encompasses jQTouch plus iWebkit+iScroll+Cubiq's Spinweels. It is for iPhone only.

EWD uses Sencha Touch custom tags to create apps for iPhone, iPad, Android, etc.

Q: What about security?
A: Well, these are web applications, so they're as secure as anything that goes over the internet. For example, Quest Diagnostics has an application called Care 360. Care 360 queries and sends test results, all using EWD, all web-based. Quest Diagnostics regularly brings in IBM's ISS security team to look at Care 360. Two years ago, the team found a very minor issue which was addressed; they haven't found anything since. EWD apps are actually more secure than native apps, because the information is not on the device.

An EWD demo program is available at http://184.73.30.147/ewd/selectAhead/index.ewd. A second program is available at http://184.73.30.147/ewd/selectAheadMaint/index.ewd, but that one needs a login and password.