Article
Rob Tweed · May 12, 2016 6m read

Announcing EWD 3: Integrating the JavaScript, Node.js and Cache communities

Those of you who keep an eye on developments in the mainstream of IT will be aware that a major upheaval has been occurring over the last 5 or so years, in which JavaScript has exploded in popularity and importance. Largely as a result of its server-side incarnation - Node.js - it has broken free of just being the scripting language that you use in web browser, to becoming the world's most popular language and enterprise technology of choice.

As soon as I started playing about with Node.js back in 2010, it was clear to me that not only was it going to be an important and major technology, but also that Caché's NoSQL database capabilities would make it a perfect partner. I saw the burgeoning popularity of Node.js as a great opportunity to bring Caché to the attention of a whole new audience, and I have focused most of my attention and free time since then on creating a way of integrating the technologies in such a way that not only would allow existing Caché developers to work comfortably in this new JavaScript-orientated world, but also would make access to and use of Caché seem natural to JavaScript developers.

Of course, many of you will know that Caché has included an interface to Node.js for some time: it's known as cache.node. However, on its own, it was not enough to meet my objectives. For Caché developers it was too low-level and required too much specialised JavaScript code to create a productive development environment. For JavaScript developers, it assumed they already knew about Caché, wanted to use Caché, and understood how Caché and its Global storage worked as a database. Sadly, in all the many JavaScript and Node.js meetings I've attended, I've yet to meet a single JavaScript developer who had previously even heard of Caché, let alone realised and understood how potentially powerful and useful it could be to them.

Some of you may be familiar with the framework/environment that I originally built around the cache.node interface: the EWD.js product. This was originally designed to make it easy to build and run Caché-based web applications, and it has been very successful, in use by quite a number of Caché-based enterprises. Over the years it has been extended in its roles and capabilities, including exposing legacy Caché applications as REST services and providing federated access to distributed Caché systems. It became clear, however, that the monolithic design of EWD.js was becoming an increasingly limiting factor, and that the time had come for a complete rethink and overhaul of its design and architecture, allowing use of the very latest JavaScript/Node.js design and coding techniques.

This work is now complete, and just this week it was launched. It's known as ***EWD 3***, reflecting the fact that it's the 3rd generation of the original EWD concept. EWD 3 is all Open Source (Apache 2-licensed), and you can use it today with pretty much any version of Caché (and Ensemble too).

Unlike the previous generations, EWD 3 isn't a single product, but a suite of Node.js modules, each of which does just one specific job. The core EWD 3 modules are designed to cleanly interoperate, but can also be used as standalone modules that can be integrated with other third-party Node.js modules. EWD 3 takes the core ideas of and lessons learnt from EWD.js and creates a new, modern and extremely powerful suite of building blocks.

So, where EWD.js was a set menu, EWD 3 is a buffet: you decide what components and building blocks you need and put them together in the way that meets your needs, whether that's to create a REST-based system, a complex run-time environment in which to run interactive, browser-based applications that communicate by Ajax and/or WebSockets, or some other environment of your choice.

For example, the ewd-xpress module is built using the core EWD 3 modules to create an application run-time

container environment that is very similar to EWD.js. However, if you look at the ewd-qoper8-vistarpc module, it uses a different assembly of the core EWD 3 modules to create a REST interface for the VA's VistA Electronic Healthcare Record (EHR).

One key design aim of the EWD 3 *ewd-xpress* module was to ensure that it could be used with the latest and most commonly-used JavaScript frameworks such as Angular.js. My own personal favourite these days is the immensely powerful and increasingly popular React.js framework from Facebook, and I've made sure that intuitive and easy-to-use React.js support is built-in from the start with EWD 3, particularly since it sets the scene for Caché-based mobile applications, developed using the same design patterns with Facebook's incredible React Native technology.

However, EWD 3 doesn't force you to use these big-name JavaScript frameworks – the same client-side JavaScript module can also be used as a standard JavaScript script file for traditional hand-crafted HTML/JavaScript applications if that's what you prefer. Indeed EWD 3 won't force you to work in any particular way or use any particular frameworks or design patterns. Just use the appropriate EWD 3 building blocks and work with them in the way you want.

So what does EWD 3 consist of? Well, the current set of EWD 3 modules includes:

Core Modules:

- ewd-qoper8: this provides the core master / worker process infrastructure for EWD 3
- ewd-qoper8-express: Express middleware interfacing for ewd-qoper8
- ewd-qoper8-cache: this provides the interface between ewd-qoper8 worker processes and Cache, via the cache.node interface
- ewd-document-store: abstracts Cache as persistent JavaScript Objects and a uniquely fine-grained document database
- ewd-session: builds on ewd-document-store to provide a Session Management environment for EWD 3 REST and interactive applications

Application-layer Modules (layered on the EWD 3 Core Modules)

- ewd-qoper8-vistarpc: provides REST access to the RPCs within the VistA EHR
- ewd-xpress: Express middleware and back-end, creating a run-time container environment for Ajax and WebSocket applications

Client-side Modules:

- ewd-client: allows registration of ewd-xpress applications and provides the messaging APIs for communication with ewd-xpress

Development Tools:

- ewd-xpress-react: React.js-specific client modules for ewd-xpress applications
- ewd-react-tools: Tools for managing and documenting React.js-based ewd-xpress applications

Management Tools:

- ewd-xpress-monitor: React.js-based ewd-xpress application for monitoring and managing an ewd-xpress / ewd-qoper8 environment

You'll find all these modules in my Github repository.

Some of the more recent modules currently only have minimal documentation. Expect that to change over coming months.

Over coming weeks, in the EWD Community Forum I'll be describing how to build Cache based ewd-xpress systems, and how to build React.js-based ewd-xpress applications, so make sure you visit this forum. Look for articles whose titles start with "EWD 3".

Expect to see further enhancement to the EWD 3 modules in coming months, and additional new application layer modules. I'm hoping others will also build new modules on top of the EWD 3 core – it's now a very powerful, modern, flexible, scalable and highly-performant set of building blocks.

I'm confident that EWD 3 meets my original design criteria: its modules such as *ewd-xpress* make it easy for existing Caché developers to create advanced mobile and browser-based applications using the very latest JavaScript techniques and frameworks. Through the *ewd-document-store* module, it also makes access to Caché intuitive for a JavaScript developer.

I'm hoping, therefore, that EWD 3 can be used to realise my original objective: to introduce the power, performance, scalability and unique capabilities of Caché to the now huge community of JavaScript developers - compared to the database technologies they're used to using, I believe it will be a revelation to them.

EWD 3. A new generation product, and an exciting opportunity to bridge the Caché and JavaScript communities to the benefit of each.

#Angular #Node.js #Open Source #React #Release #Tools #Caché

Source URL: https://community.intersystems.com/post/announcing-ewd-3-integrating-javascript-nodejs-and-cache-communities