Web Workers & Clojure Boot

NYC Clojure Users Group
Wednesday, March 30, 2016
by
Bill La Forge
laforge49@gmail.com
http://aatree.github.io/

What are Web Workers?

- Threads that run in a Browser.
- Web workers are isolated from the window thread and from each other, running in their own memory space. Data is not shared.
- Web workers do not have a Window object. Instead, each worker has a Worker object.
- Shared workers use the same js file; dedicated workers do not. With dedicated workers you may be loading multiple copies of clojurescript.

Why use Web Workers?

- To access additional computational resources.
- To prevent file i/o from hogging the window thread. Even asynchronous file i/o like IndexedDB will tie up a thread.

Web Workers are used to keep the aplication responsive to user input.

Clojurescript Issues

- Web workers only work with optimized code, as the unoptimized js references the Window object, giving rise to a null-pointer exception. If this is an issue, use a dedicated web worker.
- Even with optimized code, use of auto-reload injects a reference to the Window object. Again, use of a dedicated web worker is a solution.

Boot, a Simplified View

- Boot is an open-ended framework for defining, composing and running build tasks. Simple things are easy to accomplish while complex things remain uncomplicated.
- Boot works so well because it does not rely on conventions, but on clojure code that you provide.
- The boot.properties file provides a minimalistic configuration, specifying the boot and clojure versions you want to use.

Boot Internals

- Internal files are effectively immutable, each task building on a set of links to temp files that can only be read, creating a waterfall of data.
- There is no boot way of doing things, no reserved places where things are put. It is completely open ended.
- Pods can be used for complete isolation of dependencies. Each pod has its own classpath, allowing the use of multiple versions of code.

w3cschools webworker demo

https://github.com/aatree/aademos/tree/master/w3c-worker

- A very simple dedicated webworker written in javascript:
  http://www.w3schools.com/html/html5_webworkers.asp
- Adopted to use boot and hoplon:
  https://github.com/aatree/aademos/tree/master/w3c-worker

boot.properties

https://raw.githubusercontent.com/aatree/aademos/master/w3c-worker/boot.properties

```
BOOT_CLOJURE_VERSION=1.8.0
BOOT_VERSION=2.5.5
BOOT_EMIT_TARGET=no
```

- Used to configure boot. Typically the same for all projects.
- Clojure version must match the version given in the build.boot file.
- BOOT_EMIT_TARGET=no will be dropped in boot version 3.0.0.

build.boot

https://raw.githubusercontent.com/aatree/aademos/master/w3c-worker/build.boot

- dependencies
- source-paths
- require
- deftask
- comp
- reload
- cljs compiler optimizations
- boot-jetty init-params

worker.cljs.edn

https://raw.githubusercontent.com/aatree/aademos/master/w3c-worker/src/worker.cljs.edn

```
{:require   [counter-worker.counts]
 :init-fns [counter-worker.counts/main]}
```

- File name and location used to name and locate the js file defined by this file.
- Not required for the window js file, as the edn file is generated when using hoplon.

the w3c webworker

https://raw.githubusercontent.com/aatree/aademos/master/w3c-worker/src/counter_worker/counts.cljs

the w3c demo code

https://raw.githubusercontent.com/aatree/aademos/master/w3c-worker/src/count/index.cljs.hl

The Duracell Demo

https://github.com/aatree/aademos/tree/master/duracell

- IndexedDB accessed from a web worker.
- Web worker js file is provided by the durable-cells library.
- Demo used boot-reload and comiles the window js with optimizations none.
- Uses boot-jetty in place of boot-http. The web worker js is a static resource, which boot-jetty supports by default.

Duracell build.boot

https://raw.githubusercontent.com/aatree/aademos/master/duracell/build.boot

- Note in particular the init parameter on the serve task, needed to run on windows.
- Instead of providing the Jetty parameters like we did with boot-jetty, boot-http's serve task takes a function.

Initializing boot-http

https://raw.githubusercontent.com/aatree/aademos/master/duracell/src/client/duracell/strap.clj

```
(ns duracell.strap)

(defn jetty-init []
  (.put (System/getProperties)
        "org.eclipse.jetty.servlet.Default.useFileMappedBuffer"
        "false"))
```

Duracell Demo Code

https://raw.githubusercontent.com/aatree/aademos/master/duracell/src/client/duracell/index.cljs.hl

- Open-durable-cells! maps cell names to cells.
- The ready cell is set to true AFTER the txt cell has been loaded. Only then is the txt cell displayed.

Durable-Cells Library

https://github.com/aatree/durable-cells

- Provides a JS worker file which makes haplon cells durable.
- Easy enough to convert from cells to atoms, as both implement the same interfaces.

Durable-cells build.boot

https://raw.githubusercontent.com/aatree/durable-cells/master/build.boot

- Uses resource-paths instead of source-paths so that the code goes in the jar file.
- :dont-modify-paths? true prevents erroneous file duplication.
- :optimizations :simple needed by webworker.

dcells.cljs.edn

https://raw.githubusercontent.com/aatree/durable-cells/master/src/worker/dcells.cljs.edn

```
{:require   [durable-cells.dc-api]
 :init-fns [durable-cells.dc-api/start]}
```

- File name and location used to name and locate the webworker js file.
- dc-api/start is the function that is called when the webworker is loaded.

aaworker library

https://github.com/aatree/aaworker

- On receiving a requests, a worker sends back either a :success or :failure response.
- Workers can send a :notice message to the client at any time.
- Clients register notice processing functions, with :alert processing defined by default.
- A worker defines request processing functions using the deflpc! macro.
- Once startup is complete, the worker sends a :ready notice to the client.

tworker demo

https://github.com/aatree/aademos/tree/master/tworker

- A simple demonstration of the aaworker library.
- With both the window and worker js files compiled at the same time, optimizations can not be none and boot-reload will fail.
- The cljs.edn file for the window js file is generated by hoplon, so only the cljs.edn file for the worker is needed.
- Also, boot-html is not required as there are no static resources.

tworker build.boot

https://raw.githubusercontent.com/aatree/aademos/master/tworker/build.boot

There should not be any surprises here.

demo worker

https://raw.githubusercontent.com/aatree/aademos/master/tworker/src/worker/tworker/demo_worker.cljs

- deflpc!
- send-notice
- process-requests

index.cljs.hl

https://raw.githubusercontent.com/aatree/aademos/master/tworker/src/client/tworker/index.cljs.hl

- new-worker!
- Register-notice-processor!
- mklocal!

„For even the very wise cannot see all ends.“