Blink/Chromium Worker Implementation Notes

kinuko@chromium.org

NOTE: AS OF 2015 THIS DOC IS BECOMING OBSOLETE, please refer to this doc instead.

Web Workers spec:
http://www.whatwg.org/specs/web-apps/current-work/multipage/workers.html

Process Types and Threading for Workers

In Chromium they are implemented in the following way (as of Oct 2013):

- **Dedicated Worker**
  - Runs in the **same renderer process** as its parent document, but on a different thread (see Source/core/workers/WorkerThread.* in blink OR browse to third_party/Webkit/Source within chromium src folder for details). Note that it's NOT the chromium's worker process thread implemented in content/worker/worker_thread.*.
  - In Chromium the renderer process's main thread is implemented by content/renderer/render_thread_impl.*, while the Blink's worker thread does NOT have corresponding message_loop/thread in chromium (yet). You can use webkit_glue::WorkerTaskRunner (webkit/child/worker_task_runner.*) to post tasks to the Blink's worker threads in chromium.
  - The embedder's platform code in Chromium for Dedicated Worker is content/renderer/* (in addition to content/child/*), but NOT content/worker/*.
  - Platform APIs are accessible in Worker contexts via WebKit::Platform::current(), and it's implemented in RendererWebKitPlatformSupport (content/renderer/renderer_webkitplatformsupport_impl.cc) in Chromium.
  - Note that to call Platform APIs from Worker contexts it needs to be implemented in a thread-safe way in the embedder code (e.g. do not lazily create a shared object without lock), or the worker code should explicitly relay the call onto the renderer's main thread in Blink (e.g. by WTF::callOnMainThread).

- **Shared Worker**

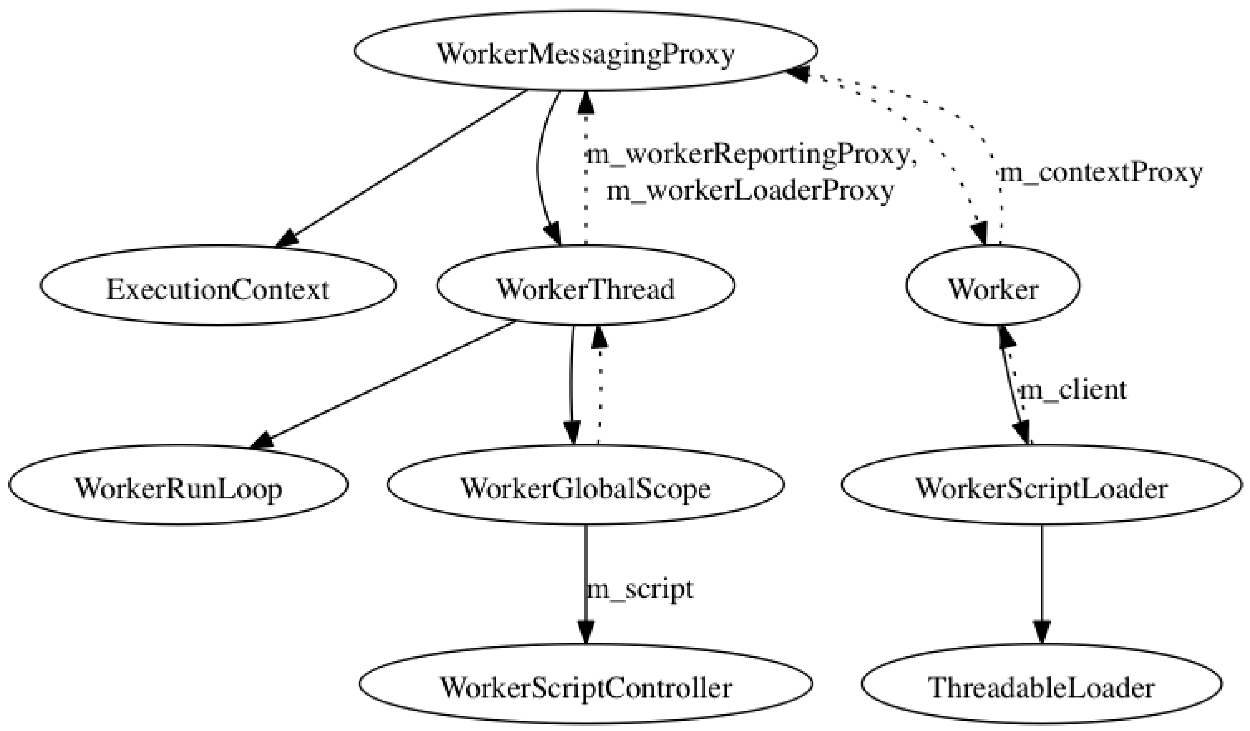
- As of Jul 2014, the below mentioned Shared Worker design has changed, with shared workers being created in the renderer process itself. For more details on the design refer the following issue and design document:
  chromium issue: http://crbug.com/327256
  design doc:
  https://docs.google.com/document/d/10P4ITgIUz8ujB3b0HRhPFhsGBVioyJg3_5uJj3Kotd4/edit

- The following design was valid till April 2014.
  - Runs in its own separate process called **worker process** and is connected to multiple renderer processes via the browser process. Shared Worker also runs in its own worker thread.
  - In Chromium the worker process's main thread is implemented by content/worker/worker_thread.* (note that this is NOT the worker thread you refer in Blink!!). As well as in Dedicated Workers the Blink's worker thread does not have corresponding message_loop/thread in chromium.
  - The embedder's platform code in Chromium for Shared Worker is content/worker/* (in addition to content/child/*).
  - Platform APIs are accessible in Worker contexts via WebKit::Platform::current(), and it's implemented in WorkerWebKitPlatformSupport (content/worker/worker_webkitplatformsupport_impl.cc) in Chromium. As well as for Dedicated Worker, to use a platform API in Worker contexts the API needs to be implemented in a thread-safe way in the embedder code, or Blink-side code should explicitly relay the platform API access onto the main thread by WTF::callOnMainThread.

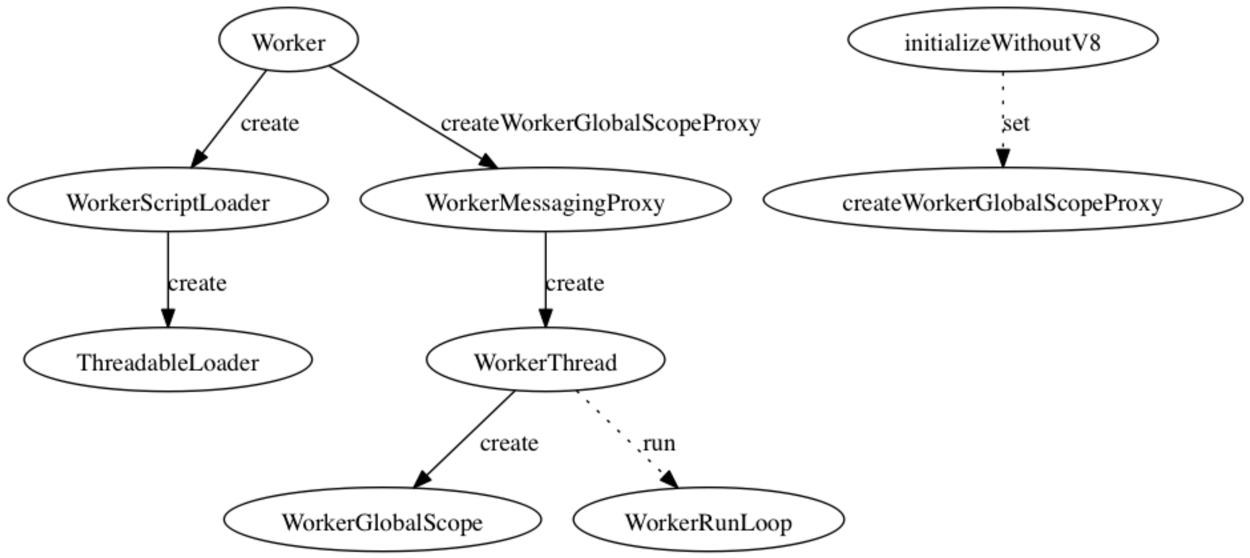
Dedicated Worker

WK::WebWorkerClientImpl == WebMessagingProxy == Worker{Loader,Object,GlobalScope}Proxy == WorkerReportingProxy

Ownership relationships



Behavioral relationships



SharedWorker (Old implementation, valid till April 2014)

Renderer process side:

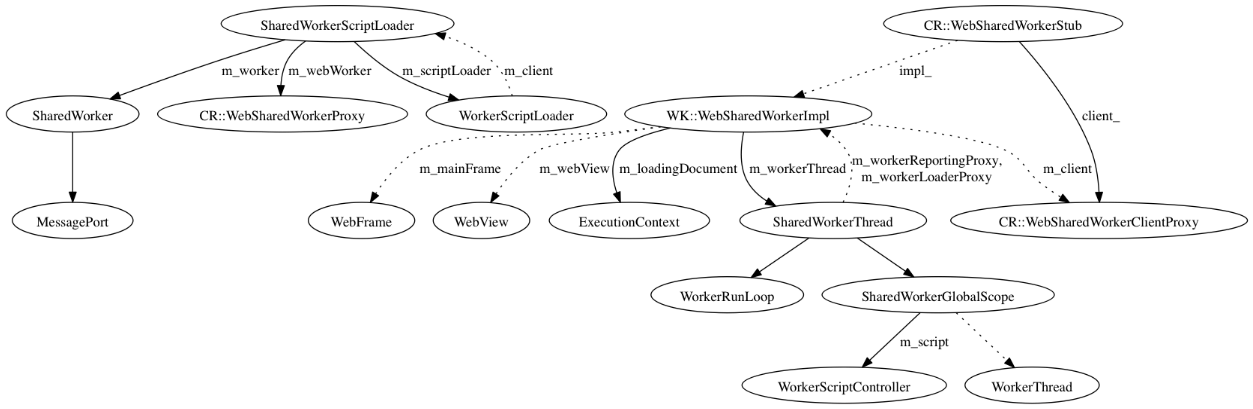
CR::WebSharedWorkerProxy == WK::WebSharedWorker

Worker process side:

WK::WebSharedWorkerImpl == Worker{Loader,Object,GlobalScope}Proxy == WorkerReportingProxy
WK::WebSharedWorkerImpl == WK::WebSharedWorker
WK::SharedWorkerImpl == WK::WebWorkerBase

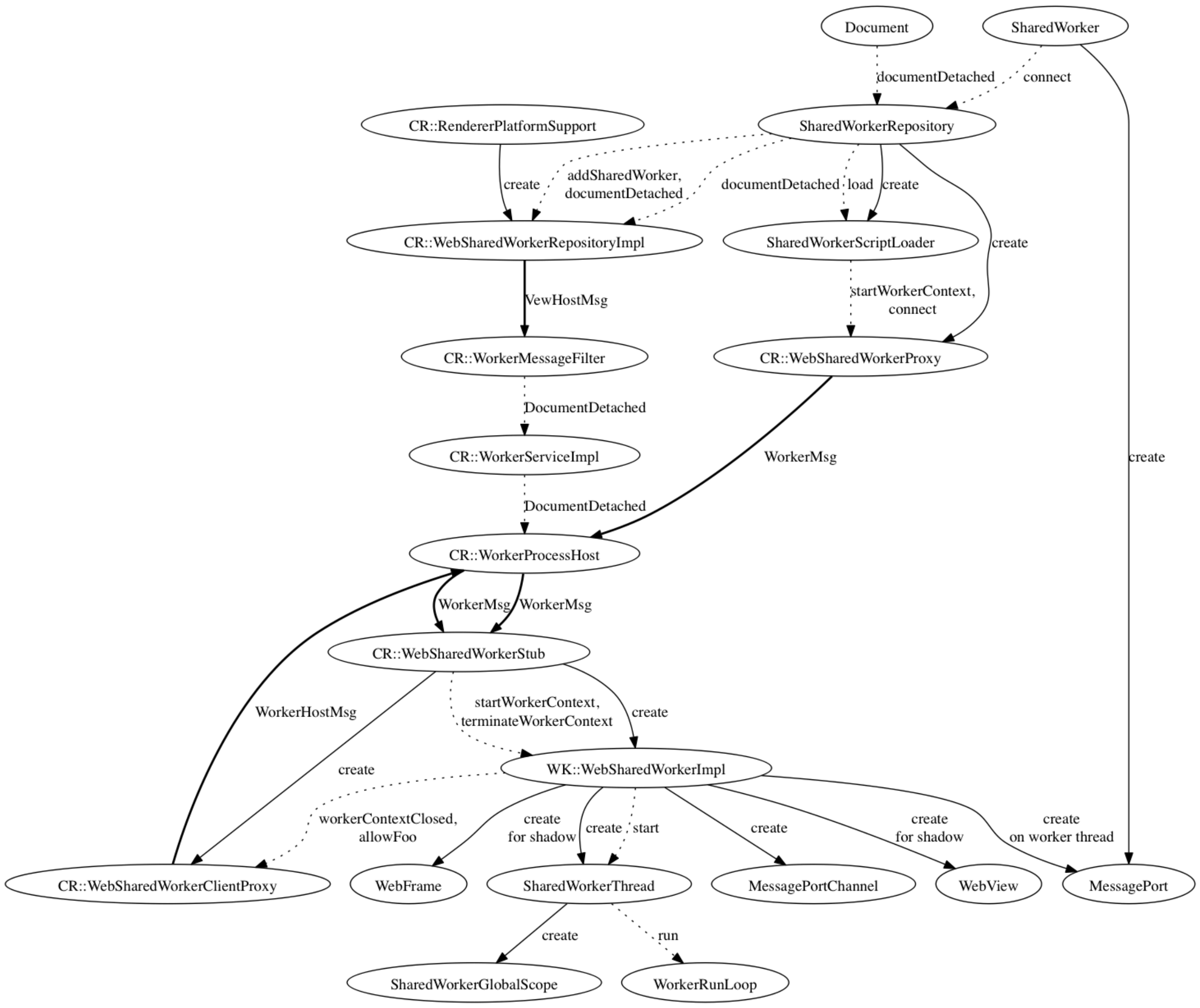
Ownership relationships

external link (for bigger image):
https://docs.google.com/a/chromium.org/file/d/0B0VUbfole2fwSUVndmFuRTR0NFE/edit?usp=sharing



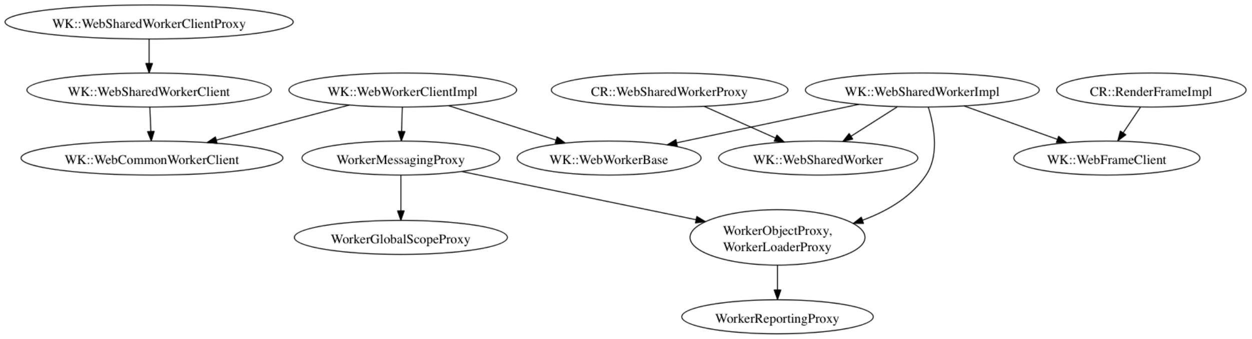
Behavioral relationships

external link (for bigger image):
https://docs.google.com/a/chromium.org/file/d/0B0VUbfole2fwNjA3QjNTS3FjQmc/edit?usp=sharing



Blink Worker Class Inheritance Tree

external link (for bigger image):
https://docs.google.com/a/chromium.org/file/d/0B0VUbfoIe2fwakpZcmc2Y19IM0k/edit?usp=sharing



Inheritance relationships