U06F82LES : you can issue a global exclusion for ring
U06F82LES : <https://github.com/technomancy/leiningen/blob/master/sample.project.clj#L86>
U0K1UT6PQ : hm. the problem being that lein-gorilla does need ring, just not that old one ...
U06F82LES : that's ok, you can specify the latest version
U06F82LES : exclusion just means "ignore whatever transitive dependencies there are relating to this"
U0K1UT6PQ : thank you, will try that :slightly_smiling_face:
U051SS2EU : to look at plugin deps there's a separate plugin tree command `lein deps :plugin-tree`
U06F82LES : <@U051SS2EU> did not know that
U0K1UT6PQ : <@U051SS2EU> nice one, that should help
U0K1UT6PQ : thank you!
U5XMV6DQT : ```lein ancient``` can help with outdated deps (it's a separate plugin though)
U0K1UT6PQ : well, it looks like it's something funky alright
U0K1UT6PQ : (as in it still doesn't show up)
U050M5F75 : question-- does anyone have a homoiconic datetime solution/approach that they use in their projs?
U050SC7SV : I like my dates served as longs, nothing else (I rarely have to care about TZ)
U06F82LES : I use only java.util.Date and js/Date
U050M5F75 : just everything UTC -> epoch <@U050SC7SV> ?
U06F82LES : with the occasional goog.time and clj-time for formatting
U4TE22XR8 : <@U050M5F75> I use this: <https://github.com/dm3/clojure.java-time>
U050SC7SV : date as number works well as long as you don't need to be super precise with "date math"
U050SC7SV : <@U050M5F75> yes
U050SC7SV : otherwise java 8 api seems decent
U050M5F75 : not a fan of `#object[java.time.LocalDate ...]`
U050M5F75 : seems strange those don't get read
U051SS2EU : that would require clojure to require java 8 right?
U051SS2EU : I guess it could do that conditionally?
U050M5F75 : would be interesting to have those dump into a constructor/factory format like `(java.time.LocalDate. &amp; args)`
U051SS2EU : the more likely option would be what clojure does today with java.util.Date - using a literal representation that the reader accepts
U050M5F75 : oh rllllly
U050M5F75 : i did not know that
U051SS2EU : ```+user=> (java.util.Date.)#inst "2017-07-04T13:25:33.376-00:00"
+user=> #inst "1492-01-10T12:11:44.000-00:00"
#inst "1492-01-10T12:11:44.000-00:00"
```

U051SS2EU : Date is a less than great API, but clojure makes it readable
U050M5F75 : does this `#inst` mean <https://clojure.github.io/clojure/clojure.instant-api.html>?
U051SS2EU : no, it's how Date objects are printed, its the instant reader
U0K1UT6PQ : I confirm the fishiness, was pulling a dependency in the code :facepalm:
U0K1UT6PQ : so that was fun
U050M5F75 : this is cool. didnt know about `#inst` and `#uuid` and stuff
U050M5F75 : thx
U3QUAHZJ6 : hello everyone, i have 4 heavy database queries that are running sequentially
```
    (benefit-db/transition-benefits-to-ongoing db/db-spec)
    (benefit-db/transition-benefits-to-consumed db/db-spec)
    (benefit-db/transition-benefits-to-ended db/db-spec)
```

is there an easy way to run this guys in parallel and do something else when they *all* finish?

U2PGHFU5U : plins: `(let [results (map deref  [(future trans1) (future trans2)  (future trans3)] do-something-else))`
```