U5ZAJ15P0 : but I get the same error even if I deleted the whole ~ctx-binding line
U5ZAJ15P0 : I never got that error with uri# or datomic# by the way, it only occured when I added conn#
U5ZAJ15P0 : It's probably something silly
U0NCTKEV8 : do you have a var named conn# in that namespace?
U051SA920 : <@U5ZAJ15P0> What does macroexpand give you?
U5ZAJ15P0 : <@U0NCTKEV8> no, I checked. Let me check again just to make sure..
U5ZAJ15P0 : `java.lang.RuntimeException: No such var: wef-backend.test-util/conn#`
U0NCTKEV8 : you could fix it immediately by replacing most of the macro with a function that returns the context, and have the macro expand in to invoking that and then deleting it
U0NCTKEV8 : but that wouldn't tell you what is wrong
U5ZAJ15P0 : <@U051SA920>```
(let*
 [uri__16604__auto__
  (clojure.core/str "datomic:mem://hello-test-" (java.util.UUID/randomUUID))
  datomic__16605__auto__
  (integrant.core/init-key :wef-backend/datomic {:uri uri__16604__auto__})
  wef-backend.test-util/conn#
  (:conn datomic__16605__auto__)
  ctx
  {:auth nil :conn conn__16606__auto__ :db (datomic.api/db conn__16606__auto__)}]
 (try (+ 1 2) (finally (datomic.api/delete-database uri__16604__auto__))))
```

U5ZAJ15P0 : when running```
(macroexpand '(wef-backend.test-util/with-scratch-ctx ctx (+ 1 2)))
```

U051SA920 : What happens when you rename it to `some-other-conn#`?
U5ZAJ15P0 : tried that, didn't work. Trying again now just to make sure
U0NCTKEV8 : is that a real pound sign, or some unicode nonsense?
U051SA920 : Well the second usage of `conn#` expands properly. (in the map)
U0NCTKEV8 : yeah, which makes me suspect the first has some weird shenanigans
U5ZAJ15P0 : Wait, I think I found the issue
U5ZAJ15P0 : `some-other-conn# (:conn datomic#)]`
U5ZAJ15P0 : the "space" between # and (has charcode 160
U5ZAJ15P0 : instead of 32
U051SA920 : oh lol. :slightly_smiling_face:
U0NCTKEV8 : hah!
U0NCTKEV8 : unicode malarky
U11BV7MTK : did you copy past some of this from a github page?
U5ZAJ15P0 : nope, I wrote it myself
U5ZAJ15P0 : no idea what happened there
U5ZAJ15P0 : well, I copied an earlier version of that code, which worked
U5ZAJ15P0 : then added that line myself
U11BV7MTK : from where?
U5ZAJ15P0 : it might have picked up a character from somewhere
U5ZAJ15P0 : github
U11BV7MTK : oh
U11BV7MTK : weird
U5ZAJ15P0 : I just checked and none of the characters from the snippet I copy/pasted had character 160
U5ZAJ15P0 : oh well
U5ZAJ15P0 : sorry for the trouble
U0NCTKEV8 : there is actually an open jira issue that has had some recent activity about changing how clojure handles unicode whitespaces
U11BV7MTK : `char code 160 would be &amp;nbsp;`
U44SHEP4N : the fun thing about this is, thats 160 is not even considered whitespace by java
U0NCTKEV8 : oh, I guess the issue it got declined
U5ZAJ15P0 : Probably what happened is that OS X and/or Atom (the editor I am using) has some shortcut or some way to enter this type of whitespace, and it fat-fingered on the shortcut
U11BV7MTK : ```employee-resizer.core> (char 160)

\
employee-resizer.core> \
\space
```

U0NCTKEV8 : <https://dev.clojure.org/jira/browse/CLJ-2207>
U5ZAJ15P0 : it had been a while since I debugged a unicode issue; almost forgot how fun it is
U5ZAJ15P0 : :kappa:
U11BV7MTK : apparently atom will enter a non-breaking space with alt-space
U11BV7MTK : probably pretty easy to inadvertently hit
U5ZAJ15P0 : yep, I most definitely inadvertently hit that
U5ZAJ15P0 : If Clojure's reader threw an error saying "unknown character at position X" it would have been easy/easier to debug, but it considered it as part of the symbol
U0NCTKEV8 : that actually may be just be a bug in the reader
U5ZAJ15P0 : Now I know how to confuse the hell out of people though
```