U09LZR36F : How is this? <https://github.com/aruberto/business-time/blob/master/README.md>
U69US348Z : &lt;deleted&gt;
U69US348Z : Ah, sorry, I guess I'll delete that and post it to <#C1B1BB2Q3|clojure-spec>
U04V70XH6 : Using that with <https://github.com/dm3/clojure.java-time> ought to get you what you need, fairly cleanly.
U04V70XH6 : FYI, at World Singles, we're slowly migrating away from clj-time to clojure.java-time (or just using Java Time directly).
U09LZR36F : That's interesting. We've been making a similar move too.
U3L6TFEJF : I have a Clojure design problem which I'd love some feedback on. I have a `SeriesStore` record and corresponding `SeriesStore` protocol. There are different stores for different accounts etc, so I need some way to create these stores. I'm considering two options:1. Have a regular fn that is a closure around all dependencies of the store, and then pass that fn around
2. Create another record which contains all the dependencies, e.g `SeriesStoreFactory`, and then pass an instance of that record around.
i.e regular fn closure vs. record, which is preferable and why?

U04V70XH6 : See this <https://github.com/clj-time/clj-time/issues/196#issuecomment-294067755> -- and read the whole thread for background. It is a discussion around moving `clj-time` to JSR-310 and the pros and cons... and why, ultimately, I came out against the switch of implementation and why I would encourage folks to use `clojure.java-time` _instead_ of `clj-time` if they want to switch from Joda Time to Java Time.
U07TDTQNL : Records almost always win <@U3L6TFEJF> for the simple reason that functions are opaque. Once you close over something with a function there's no way to introspect the closure to see what you closed over.
U07TDTQNL : Sometimes you need that for debugging, other times just to do `keys` or to otherwise get the state out inside some function.
U3QUAHZJ6 : hello everyone while using `clojure.test` its possible to share data between tests?
something like

```
(def data (atom :none))
(deftest t1 ... (reset! data 10))
(deftest t2 ... (do-stuff 10))
```

U3QUAHZJ6 : (do-stuff data) *****
U5ZAJ15P0 : If I wish to set up some data that should be accessible to all tests (either set up once before all tests or set up and tear down before and after each test), how can I do it with clojure.test? with-fixtures doesn't appear to let me pass data down to the test. I could use a global atom but that seems nasty
U5ZAJ15P0 : <@U3QUAHZJ6> oh wow, haha, I hadn't read your message before posting
U5ZAJ15P0 : looks like we have the exact same question
U3L6TFEJF : <@U07TDTQNL> sounds reasonable. but sometimes you want it opaque though? my main concern with records is that I want to keep it as functional as possible, and I worry that my code will start to look like OO if there are too many records all over the place. what's your experience on striking the right balance between records and fns?
U050MP39D : <@U3L6TFEJF> records are still immutable. they're still functional
U2J4YH6BG : <@U3L6TFEJF> in clojure Ive never been like, I wish I had less information at my disposal
U07TDTQNL : Right, you got to separate out "is a protocol" from "looks like OOP"
U050MP39D : polymorphism is orthogonal from "functional" imo
U2J4YH6BG : also I would recommend checking out stuart sierra's component or weavejesters integrant
U07TDTQNL : OOP = information hiding, local mutation, and inheritence
U07TDTQNL : records and protocols don't do any of that.
U3L6TFEJF : true, true. and I actually have polymorphism in this case, so I'll go with records. thanks for the input!
U3L6TFEJF : <@U2J4YH6BG> I've tried the whole gamut of state management libs in clojure, I'm currently experimenting with <https://github.com/vspinu/commix> which I think is fantastic so far!
U2J4YH6BG : :slightly_smiling_face: there are alot, I havent even heard of this option
U3L6TFEJF : It's less than a week old :smile:
U2J4YH6BG : sounds ready for production to me
U3L6TFEJF : I wouldn't, but hey, I won't judge you :wink:
U07TDTQNL : Better than most node libraries! #burn
U2J4YH6BG : &gt; Commix was built as a response to a range of limitations in Integrant which in turn was designed to overcome limitations in Component.
U2J4YH6BG : turns out dependencies are hard?
U5ZAJ15P0 : I've been using integrant for a couple of days, it has been great so far

U2J4YH6BG : I liked it as well when I compared it with component
U3L6TFEJF : I tried using Integrant but I immediately found myself working around it instead of with it, then I stumbled over commix here: <https://github.com/weavejester/integrant/issues/21>
U3L6TFEJF : I recognized some of the problems I encountered in the "differences from Integrant" section
U3L6TFEJF : anyways, more options on the table, more innovation = win for everyone
U5ZAJ15P0 : Does anyone have a recommendation for me on how to share some stuff between tests? e.g. a database connection
U3QUAHZJ6 : well if is something static i think you can just
```
(def db-spec (..))

(deftest test
  (jdbc/query db-spec ..))
```

i really would like to know how i can share data that is produced by one test in another test

U5ZAJ15P0 : <@U3QUAHZJ6> why would you want to do that?
U3QUAHZJ6 : im trying to do some sort of all-or-nothing integration api integration test
imagine that i do a request to create an user, and it returns me an id
afterwards in another test id like to user this id to create a bank account for this user (or something like that)

U5ZAJ15P0 : <@U3QUAHZJ6> ah, I was planning on doing the same thing. It seems to me this should be done within one test block? I don't think there are much guarantees as to in which order tests will run
U5ZAJ15P0 : and you would rely on order there
U3QUAHZJ6 : im trying to avoid a giant let with tons of implicit tests, but i seems that i got no other option
U5ZAJ15P0 : <@U3QUAHZJ6> I am a newb, but what do you think would be the issue? You can nest `testing` calls to create context
U050MP39D : dynamic vars and a fixture would be another option but probably not a very good one
U5ZAJ15P0 : and from what I just inferred from the source of `testing`, it returns its last expression
U5ZAJ15P0 : so you could use that to pass values around