U1KE7MFDY : ```(defn valid-users
  [username password]
  (->> users
       (map (juxt :user/name :user/password))
       (some #(= [username password] %))))
```

U1ALMRBLL : Now you can make your function:```
(let [valid-user? (valid-user-fn master-users-list)]
  ;; now use valid-user? as you wish...
  (if (valid-user? "bob" "abc123")
    ...```
and if you want to use a different set of username/passwords, you're not tied to any particular one. just make the new `valid-user?` function by calling `valid-user-fn` with your list

U0CGFT70T : <@U1KE7MFDY> <@U1ALMRBLL> thanks this is what I was looking for...  both great suggestions. <@U050MP39D> lol, right not a real system...lol... just creating an om-next tutorial, so just for edification purposes! :slightly_smiling_face:
U0W0JDY4C : hmm. i always seem to have a hard time pinning down when to use a protocl
U1LCB75M2 : generally, it's useful when 1) you're interop-ing w/ java (so you can use `extend-protocol` and do type-based dispatch) 2) you create a protocol + record to manage state lifecycle
U1LCB75M2 : otherwise, if you're just manipulating data (not state), simple data structures + the ad-hoc dispatch available w/ multimethods works nice and is more flexible/open
U1LCB75M2 : in other words... type-based (in Clojure case, actual Java types) dispatch = nominal typing, ad-hoc dispatch = more like structural typing
U0W0JDY4C : much to ponder. thanks for letting me take your time, btw
U1LCB75M2 : :+1:
U4SKJCP3K : How can I get the total bytes of an input stream in Clojure?
U0CAUAKCG : Is this a bug?```
(defn callable
  [fun]
  (proxy [clojure.lang.IFn] []
    (invoke [&amp;amp; args] (apply fun args))))

((callable +) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18) =&gt; 171

((callable +) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19)
1. Unhandled java.lang.UnsupportedOperationException
   invoke
```

U050MP39D : of an input stream? that's kind of impossible by definition
U050MP39D : <https://docs.oracle.com/javase/7/docs/api/java/io/InputStream.html>
U0CAUAKCG : <@U4SKJCP3K> count a byte-array
U0CAUAKCG : U4SKJCP3K change your name
U1LCB75M2 : <@U4SKJCP3K> I guess this works (if you have memory) `(-> stream slurp .getBytes alength)`
U1LCB75M2 : otherwise use `<http://clojure.java.io/reader|clojure.java.io/reader>`
U0CAUAKCG : aaah sorry, I got a slack bug, your guy's name was for a moment some hash haha
U4SKJCP3K : <@U050MP39D> Yeah, that does sound weird. What I am doing now is:
1. Make an HTTP request and get back a JSON file
2. Convert that to a ZIP file (gives me back an input stream)
3. Need to know the size of this payload

U4SKJCP3K : hcarvalhoaves: Nice, I will try this. Thanks :+1:
U11BV7MTK : <@U0CAUAKCG> i think you're running into the limitation that you can only have 20 arguments to a function
U4SKJCP3K : hlolli: Interesting, I will give it a shot. Thank you.
U11BV7MTK : and i 've heard someone talk about "trivial" functions through the repl are not invoked in the same way as normal functions
U1LCB75M2 : <@U4SKJCP3K> are you're trying to determine the file size to send on the response?

U0CAUAKCG : any hacks to bypass it? It's a bug in `overtone`, changeing the implementation where this is defined would be pain
U4SKJCP3K : <@U1LCB75M2> Precisely. I'm trying to do all these steps without performing IO.
U11BV7MTK : i'm way out of my depth on that one. I think <@U0NCTKEV8> or <@U051SS2EU> would know way more than me
U1LCB75M2 : <@U4SKJCP3K> you would have to hold all response in memory, use a buffered reader + this instead -> <http://greenbytes.de/tech/webdav/rfc2616.html#rfc.section.3.6.1>
U4SKJCP3K : hcarvalhoaves: Oh God, this looks crazy :slightly_smiling_face:
U0NCTKEV8 : yeah, proxy doesn't rest args like that
U0NCTKEV8 : doesn't support
U0CAUAKCG : *facepalm*
U1LCB75M2 : otherwise you have to make sure you don't OOM :wink:
U0NCTKEV8 : I surprised the first thing works at all
U4SKJCP3K : I am sending this payload to S3 and one of the requirements is to pass in the content length alongside the payload
U4SKJCP3K : I guess temp files are the simple solution to this here
U0NCTKEV8 : maybe proxy does sort of support that, I am not sure, depends on how you read the docstring, but I have never read it has supporting that
U0CAUAKCG : <https://github.com/overtone/overtone/blob/master/src/overtone/helpers/lib.clj#L146>here when the range is changed I bump into the same error as I posted above.

U0CAUAKCG : the proxy is defined on line 100.
U0NCTKEV8 : I was going to say, that is defrecord which is a different beast entirely from proxy
U0CAUAKCG : I think this hack could have been done better when this was written way back.
U0CAUAKCG : a macro that takes all arguments and puts them into a map or vector.
U0NCTKEV8 : it is also likely really old code, given that it uses proxy instead of reify there
U0CAUAKCG : Yes, reify a protocol would maybe be a better solutin?
U0NCTKEV8 : no, I mean, those are all interfaces it is proxying so reify is likely a better choice
U0CAUAKCG : ah
U11BV7MTK : `(~invoke_fn this# ~@args)`. does this have to many arguments with `(range 21)`?
U0NCTKEV8 : I am not familiar with the codebase, but from scratch I might prefer to pass around a map with a key that maps to a function
U0NCTKEV8 : the issue is, the last arity of invoke needs to call applyTo, or something like that, you should check out AFn.java
U0CAUAKCG : the limit is 10 arguments because in overtone they come in pairs.
U0CAUAKCG : this limit has been there for long time, nobody has taken the time to fix this
U0NCTKEV8 : if it is going to use proxy, it shouldn't proxy IFn, it should proxy AFn
U0NCTKEV8 : there are invoke arities from 0 to 21, the 21st arity takes 20 args + an array as the 21st
U0CAUAKCG : I see what you mean, but I can't see how that would be implemented. It's macroexpanding to a argument pyramid of invoke.
U0CAUAKCG : Or just somehow collect all the arguments as one parameter.
U0CAUAKCG : ah ok, read better, last arity invokeing applyTo, I see
U0NCTKEV8 : you can't do it with proxy
U0NCTKEV8 : or anything really
U0CAUAKCG : this macro `defrecord-ifn` has always some function or macro wrapped around it, there an &rest sequence could be used.
U0CAUAKCG : no, sorry, forget what I wrote, it's the defrecord that gets called in the end
U0CAUAKCG : macros, brainfuck
U0W0JDY4C : does anyone have experience with clojure.walk/postwalk? it says it walks on each form but sometimes there's strange vectors showing up where there should be maps.. not really sure how to write what I'm trying to write
U0W0JDY4C : looking for a generic way to walk a structure and convert something like```
{:a 1 :b {:c {:d 2}}}
into
{:a 1 :b {:c {:d 2 :id :c} :id :b}}
```

U0W0JDY4C : but to do this seems like postwalk needs some sort of "look back"
U051SS2EU : <@U0W0JDY4C> hash-maps are made of two element vectors (entries) - you can see it if you call seq on one
U051SS2EU : you can do that transform above by checking for hash-maps inside each hash-map, and updating them
```

to include their id key before returning it
U051SS2EU : so your conditional would check if the arg was a map, then if it is check for maps in vals of the map, and update those vals
U46LFMYTD : Hey <@U61HA86AG> , thanks for the links
U46LFMYTD : Sorry for the late reply, I took some time to read the articles. I think I get the gist of what is meant by data-driven. I never learned Object Oriented design patterns anyway, so its hard for me to contrast what I'm doing in clojure against say, java.
U46LFMYTD : I am curious about the following, isn't creation of a record and association protocols a more object oritented, less data-driven approach?
U46LFMYTD : For example, a data-driven approach to me would just be an associated map. Maybe it has a type keyword. I can write a function which looks up :type in the map and does something with the corresponding value
U46LFMYTD : whereas creating a new record seems similar to creating a new object, and writing protocols seems to me to be like creating methods