U23SA861Y : sry `(Result Error (List User) -&gt; msg) -&gt; Cmd msg`
U23SA861Y : look at the type signature for send and replace `a` with `List User`
U40QW928G : yeap that worked
U40QW928G : now idk what to do with the update
U40QW928G : how do I match fetchUser in my base
U40QW928G : case*
U31A7HG1E : I noticed this commit from a few days ago in elm-lang/elm-compiler:
<https://github.com/elm-lang/elm-compiler/commit/1361e2b6469be2af57a2694fae6c81bdba36c845>
U23SA861Y : `UsersFetched (List User)` would be one of your main message types
U31A7HG1E : Is that a sign that 0.19 is close to being released? Or maybe not?
U23SA861Y : you would then call `fetchUsers UsersFetched`
U23SA861Y : ```type Msg = FetchUsers | UsersFetched (List User)

update msg model =
  case msg of
    FetchUsers -> (model, fetchUsers UsersFetched)
    UsersFetched users -> (addUsers users model, Cmd.none)
```

U40QW928G : it still seems like poor architecture to have `(Result Error (List User) -&amp;gt; msg)` instead of `FetchUser` simply because we don't want to depend on that type because it's in a file that causes circular dependency
U40QW928G : idk
U40QW928G : seems like it's not well thought out
U23SA861Y : not at all
U40QW928G : because if I change FetchUser I'll have to change that signature everywhere
U23SA861Y : it's all about dependency injection
U23SA861Y : and seperation of concerns
U23SA861Y : the user code is about generating tasks
U23SA861Y : the Msgs are all about routing to where they need to go
U0JFGGZS6 : sylviecottrell: it's hard to know where to start of these kinds of data modelling questions sometimes, I can definitely relate..!
U40QW928G : so leave Msg type in the main part of the application?
U0JFGGZS6 : One place to begin is to really look at/think about what are your UI needs. What bits get shared between different views, what bits you aren't sure if they will be shared, what will be shared in some ways but not in others. That can help guide how you design your types _independently of whatever the database models look like_.  But it sounds like you may have done this already.
U5LFUHH19 : <@U0LPMPL2U> <@U48AEBJQ3> Can you tell me if I'm using this correctly?  I'm a bit confused because if the String-&gt;Int fails (e.g., a missing value) then I get nothing at all.  Is this right?
<https://gist.github.com/kyclark/fc42c06fb1b5e0252e5c3ac92c25b2fd>
U23SA861Y : if you split your view and update then it should be factored out on it's own to prevent view from depending on update.
U23SA861Y : <@U5LFUHH19> mind dropping it in ellie as opposed to a gist
U0JFGGZS6 : I found this talk/slide share really helpful on some of the general issues:
<http://fsharpforfunandprofit.com/ddd/>
U0JFGGZS6 : his example domain is in the same ballpark as what you are modelling too.
U5ABF3BH7 : Thanks a lot! Let me give you a concrete example. Let's say you are building the field for your view model.  You need to collect info about a organization and a person for example, in your viewModel, do you list the fields together as list like ``` orgname, orgbranches, personfirstName, personLastName ....``` or do you try to build a sub model like ```org : OrganizationView.model, person : Person.Model```
U5LFUHH19 : <@U23SA861Y> Yes, sorry: <https://ellie-app.com/3GndvL7jDbza1/0>
U0JFGGZS6 : what do you see as the advantages/disadvantages of one vs the other?
U23SA861Y : so the fromResult is designed to force the decoder to fail if value is not a string
U23SA861Y : not a valid int
U5ABF3BH7 : The first one is straight forward you don't have extra complicated layers, yet encoding is less straight forward.
U23SA861Y : what do you want to happen when the value is an empty string
U5ABF3BH7 : The second one adds more layer, but the link between the data and the view is very clear
U23SA861Y : or the value is an invalid integer
U5ABF3BH7 : How do you usually do it?
```