Callable or Runnable or Both

The runnable example kicked off a task iterating all states and all contacts for each state sequentially. This means it takes longer for the result to be posted to sessionScope, but the user can continue working in the meantime. This is good for long tasks where the user can come back later for the outcome or doesn't need to see the outcome.

The callable example kicked off a thread for each state, processing the contacts in parallel over (potentially) 10 threads. Note that the actual number of threads available at any one time may be fewer. This means it is quicker to process all the contacts, but the user has to wait for all states to be processed. However, the outcome can be posted to viewScope, because we know they're on the same XPage still when the tasklets have all completed. This is good for tasks that can be chunked, but the overall running time will not be excessive and so the user can wait for the outcome. Obviously there is no point using a callable if the user never needs to see the outcome.

But of course the two could be combined. Code could kick off a runnable to get the states and then kick off callables for getting the contacts in each state. The runnable could then be tasked with the for loop iterating over the results of the Futures and write them to a sessionScoped variable. This is useful if the user can come back later for the outcome or doesn't need to see the outcome, but sub-tasks can be chunked and completed quicker in parallel. However, this will have an impact if other tasklets are triggered by the same user or other users. Xots only has 10 threads available, so the other tasklets may need to sit and wait, which may then have a more significant impact on that particular user. So if it does not matter if there's a single thread taking longer or a bunch of smaller threads running at the same time, the driver for making the best architecture decision may be what else may be running on the server at the same time.