let me answer some of your questions, I hope it helps....

the viewmodel's (model of view) role , in my view, has UI logic and state of a view

The viewmodel should never have any UI logic or "view state" in it. For the purposes of this explanation, I would define view state as scroll position, selected row index, selected index, window size, etc. None of those belong in the viewmodel; things like SelectedIndex are specific to the way the data is shown in the UI (if you change the sort order of a DataGrid then the SelectedIndex can change, even though the SelectedItem is still the same). In this particular case, the SelectedItem can be bound to the viewmodel, but the SelectedIndex shouldn't.   
If you need to keep track of UI session type info them then you should come up with something generic (for example, I have persisted view state before by saving important stuff into a KeyValuePair list) which is then "saved" with a call to the viewmodel (via the interface I mentioned previously). The view has no idea how the data is being saved, and the viewmodel has no idea the data is coming from a view (it has simply exposed a call through its interface).

and the view's role is displaying some contents and synchronizing the viewmodel(having databinding code)

Yes, the view's responsibility is simply to visually display data presented by the viewmodel. The viewmodel gets the data from the model (the model is responsible for making database calls or WCF webservice calls, this will usually be done via a "service", but that is a whole other discussion). The viewmodel can then shape or manipulate the data, i.e. it may get a list of all customers, but only expose a filtered version of that list (maybe the current customers) in a public property which the view can then bind to.   
If the data is to be manipulated into something visual (a common example is an enum value being translated into a color), then the viewmodel still only has the enum value(s), and the view still binds to that value, but the view also [uses a converter](http://www.switchonthecode.com/tutorials/wpf-tutorial-binding-converters) to translate the pure data to a visual representation. By using the converter the viewmodel has still avoided doing anything UI related, and the view has avoided any real logic.