Random notes from MVP Summit conversations on Expansion First

# In response to misunderstanding a Q from LWischik

The code needs to track the “current” template – the one it was created from. Thus the act of changing a template needs to include a step of “reapplying” it, which would change the code on the right.

My current vison of the stored artifact is the left and a pointer to the template fragment. The right is an in memory working copy. Changes there are pushed back to the left or template in one of three ways:

* The change, by its nature means a metadata change. In the sample, changing a property name is this type of change
* The programmer intended to change the template in one or more locations. There are two sub paths
  + the change should be reflected in al copies of the template. This change might involve the additional complexity of a conditional case (see note below). This is the common case.
  + A clone of the template should be created and a path to determining which other scenarios of the current template need the modification.
* The programmer wishes to make custom code. In the INotifyPropertyChanged sample, this would be all the custom code of the class which should move from left to right and back without alteration. It is possible that coloration marking is desirable of custom/metadata, and it’s possible this should happen only on a hotkey.
* The user has changed one occurrence of a symbol, and intends a rename of the symbol throughout the specific implementation (all right occurrences of \_xf\_Property\_dot\_Name and \_xf\_Property\_dot\_Name\_as\_CamelCase should be changed)
* The user made a mistake

That is the universe I have currently thought of.

NOTE on conditionals: I anticipate fragments will be conditionally applied. This will likely mean that the true template is sometimes a small number of variations selected on clear metadata or configuration rules. While conditionals in templates is sometimes required, most conditionals are easier to capture outside a template language.