An interesting wrinkle developed after we had built the first iteration of this property-binding framework. The first screen we tried it out on was the customer registration form. Customer registration is fairly straightforward, just a bunch of text fields, one checkbox, and a few buttons. The second screen, the album screen, is much more visual and interactive. It uses numerous GUI widgets: two proof sheets, a large image editor, a slider, and several command buttons. Even here, the form makes all the real decisions about selections, visibility, and enablement entirely through its properties. So the album form knows that the proof sheets' selections affect the central image editor, but the screen is oblivious. Keeping the screens "dumb" helped us eliminate GUI synchronization bugs and enabled much stronger unit testing.

## IS ONE ENOUGH?

On some screens, proof sheets allow multiple selections; on others, only single selection. Worse yet, some actions are allowed only when exactly one thumbnail is selected. What component would decide which selection model to apply or when to enable other commands based on the selection? That's clearly logic about the UI, so it belongs in the UI Model layer. That is, it belongs in a form. The UI Model should never import a Swing class, so how can forms express their intentions about selection models without getting tangled up in Swing code?

We decided that there was no reason to restrict a GUI component to just one binding. In other words, we could make bindings that were specific to an aspect of the component, and those bindings could attach to different form properties.

For instance, we often had separate bindings to represent the content of a widget versus its selection state. The selection bindings would configure the widget for single- or multiselect, depending on the cardinality of its bound property.

Although it takes a long time to explain the property-binding architecture, I still regard it as one of the most elegant parts of Creation Center. By its nature, Creation Center is a highly visual application with rich user interaction. It's all about creating and manipulating photographs, so this is no gray, forms-based business application! Yet, from a small set of straightforward objects, each defined by a single behavior, we composed a very dynamic interface.

The client application eventually supported drag-and-drop, subselections inside an image, on-the-fly resizing, master-detail lists, tables, and double-click activation. And we never had to break out of the property-binding architecture.