it easier to approach the elephant from several perspectives, but paper doesn't yet support hyperlinks very well.

As we look at each of these facets, keep in mind that they are different ways of looking at the overall system. For instance, we used a modular architecture to support different deployment scenarios. At the same time, each module is built in a layered architecture. These are orthogonal but intersecting concerns. Each set of modules follows the same layering, and each layer is found across all the modules.

Indeed, we all felt deeply gratified that we were able to keep these concerns separated while still making them mutually supportive.

Modules and Launcher

All along, we were thinking "product family" rather than "application" because we had to support several different deployment scenarios with the same underlying code. In particular, we knew from the beginning that we would have the following configurations:

Studio Client
: A studio has between two and four of these workstations. The photographers use them for the entire workflow, from loading images through to creating the orders.

Studio Server
: The central server inside each studio runs MySQL for structured data such as customers and orders. The server also has much more robust storage than the workstations, using RAID for resiliency. The studio server also burns the day's orders to DVD.

Render Engine
: Once in production, we decided to build our own render engine. By using the same code for rendering to the screen in the studio and to the print-ready images in production, we could be absolutely certain that the customer would get what they expected.

At first, we thought these different deployment configurations would just be different collections of *.jar* files. We created a handful of top-level directories to hold the code for each deployment, plus one "Common" folder. Each top-level folder has its own *source*, *test*, and *bin* directories.

It didn't take long for us to become frustrated with this structure. For one thing, we had one giant */lib* directory that started to accumulate a mixture of build-time and runtime libraries. We also struggled with where to put noncode assets, such as images, color profiles, Hibernate configurations, test images, and so on. Several of us also felt a nagging itch over the fact that we had to manage *.jar* file dependencies by hand. In those early days, it was common to find entire packages in the wrong directory. At runtime, though, some class would fail to load because it depended on classes packaged into a different *.jar* file.

Download at WoweBook.Com