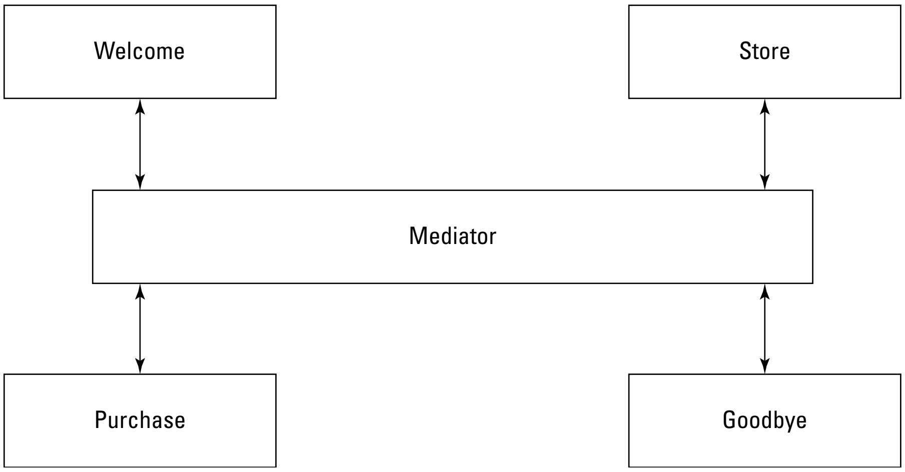

Figure 1-2: The mediator directs Web site traffic.

You can build the mediator to deal with the internals of each page so the various pages don't have to know the intimate details of the other pages (such as which methods to call). And when it's time to modify the navigation code that takes users from page to page, that code is all collected in one place, so it's easier to modify.

Adapting to the Adapter Pattern

Here's another design pattern, the Adapter pattern. Say that for a long time you've been supplied with a stream of objects and fit them into code that can handle those objects, as shown in Figure 1-3.

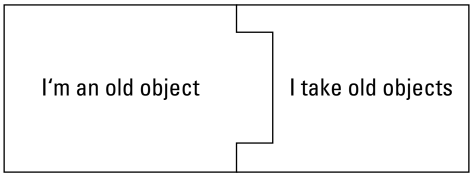

Figure 1-3: Everything seems to be working here.

But now say there's been an upgrade. The code isn't expecting those old objects anymore, only new objects, and the old objects aren't going to fit into the new code, as shown in Figure 1-4.

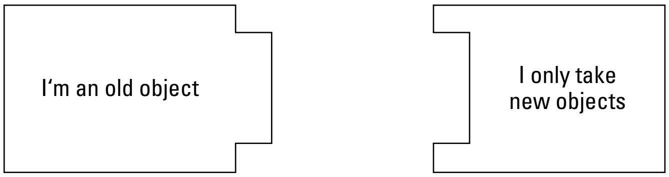

Figure 1-4: This isn't going to work.

If you can't change how the old objects are generated in this case, the Adapter pattern has a solution — create an adapter object that exposes the interface expected by the old object and the new code, and use the adapter to let the old object fit into the new code, as shown in Figure 1-5.

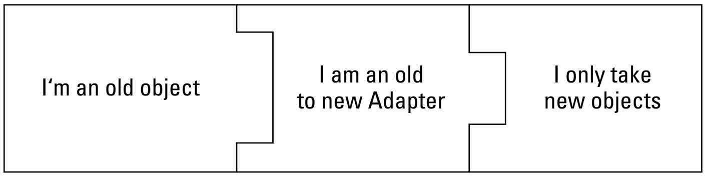

Figure 1-5: Old objects work with new objects via an adapter.

Problem solved. Who says design patterns are hard?

Standing In for Other Objects with the Proxy Pattern

Here's another pattern, the Proxy design pattern. Say that you've got some local code that's used to dealing with a local object as shown in Figure 1-6: