Why Use Templates?

The reasons for using templates to hold the HTML for your website may not be clear at first, or may even seem overly cumbersome. So why should you use templating, then, if it is so much of a burden? Because *most programmers are not designers, and most designers are not programmers.*

Since templating affects different departments different ways, let's take a look at it from a few different views:

Owner/CEO/Head Geek

So your one of your managers has come to you with the idea of incorporating a templating system. The project is close to completion, and you just *know* that saying "yes" to the proposal will push their already overstretched timeline further out.

So why should you allow them to spend a bunch of time converting code with no visible benefit? When your website swells to 500 pages, this will become clear: without templating, the number of developers required to implement minor changes—or your time to market—becomes absurdly large; with templating, large changes get pushed out fast, your designers and programmers are very productive, and maintenance is relatively simple.

But why use CS-Content? Because your designers can build a presentation quickly that has a standardized look and feel. New pages are built quickly. Approved demo pages quickly become dynamic, functional pages as your programmers interface with the templates. When you want to implement a small change to the look and feel of the site, or of one particular portion of the site, the changes are implemented quickly.

Management

Breaking development into design and programming makes everything faster: your programmers concentrate on programming, and your designers concentrate on designing. CS-Content makes this process even faster by allowing the designers to create templates that are immediately viewable simply by naming them properly and placing them in the correct folder. The programmers can then seamlessly integrate code that makes those pages and templates dynamic.

Programmers

Working on how the interface looks sucks, especially when you have to spend so much time on something that you're simply not familiar or good with. By using CS-Content, you can concentrate on what you're best at, without worrying about how the interface will look—except how your code will work with the templates.

Connections to the interface (templates) consist of setting template vars (usually defined by column names from the database or whatever the designer decides) and utilizing block rows. You may have to go into the templates and change some static content into block rows and template vars if your designer doesn't do that for you, but it's much better than having to care about what kind of CSS style to add to the row, or how to make that input look fancy.

Designers

Trying to edit HTML files that contain PHP code is usually hard to grasp: you don't know the code, and you don't care to ever learn it. With CS-Content, you don't have to look at *any* code when designing the pages. What's more, you can quickly and easily build pages for demonstration purposes that are static, and not worry about the code behind it.

When you've finally built a page that management has approved, you can then go back and work with the developers to determine what parts need to be dynamic. Creating block row definitions, determining where template vars should be, and what items in a row need to be dynamic come into play now: talk to the programmers to determine what information goes where, and how to name it for fastest & simplest integration with the code.