Robert L. Bogue
MS MVP, MCSE, MCSA:Security
317-844-5310
Rob.Bogue@ThorProjects.com

Managing the Code When Customizing SharePoint

One of the most common problems that organizations that are customizing SharePoint have is how do they manage the code deployment process. Organizations typically have configuration management guidelines that help them regulate how code makes it into production whether that's internally developed code or are patches that are applied to the operating system and products being leveraged.

The problem is that SharePoint doesn't fit a nice clean model. Because so much of SharePoint is configuration and data driven, code that works in development or a test environment may not work in Prod because things are configured differently or there's different data to operate on.

So how do you manage a situation where you've got configuration information that needs to meet up with code to create a complete solution? In this article I'll tackle this problem, a generic approach, and talk about a few of the sharp edges on the strategy that you'll have to educate everyone on.

A Simple View of Environments

In order to facilitate a discussion, let's assume that we have three formal environments:

- **Development** -- A system integration environment where developers have (or potentially have) access to the servers and to the administrative controls. Some organizations call this environment the "Wild West" because it is only lightly under configuration management. This actually a systems integration environment where all code is forced to live together with test data before moving forward.
- **Quality Assurance (QA)/Test** -- The Quality Assurance (QA) or Test environment is where the code from development and the content or a subset of the content from production comes together. This is the place where final signoffs are received prior to going into production. There might be multiple QA environments in an organization but for our purposes here they can be considered as once since they all live in the same space of configuration management. This environment is generally well controlled and follows a change management process - however, typically with fewer sign offs than the production environment.
- **Production** --The tightly controlled production environment where the tightest restrictions on access are and where availability, reliability, and scalability are of paramount importance. This is the final resting place for code and in at least some environments is the origin for all content.

Certainly it's possible to have more environments or a more complex scenario; however, this simple model will make it easy to talk about the concepts. The one assumption that's not listed as a formal environment in the above is that developers will have their own environments locally which aren't formalized. Their initial development work will take place on their local machines or dedicated development machines - not in the development environment.

Content and Configuration Never Moves To Production, Only Away From It

One of the challenging, and powerful aspects of SharePoint is that the configuration of the system and the content for the system are comingled into a single content database. Because of that it's difficult to extract either the content or the configuration without bringing along the other. As a result, it's too easy to accidentally overwrite production content if you try to pull up content or configuration from another environment. As a result, the hard and fast rule is that content and configuration always move from Prod backwards through the environments to QA and if necessary to Development.

This isn't all that different conceptually from how organizations approach QA environments for other systems. It's common place to migrate production data, or parts of the production data, back into a QA environment to test changes to an application and run validation suites. The key difference is that with SharePoint the code and the content can be more tightly coupled than is common for most traditional systems. This always move from production approach isn't without its limitations not the least of which is what content to migrate backwards and what to do about required configuration information which is covered in the following sections.

Partial Migration

Depending upon the environment size, the amount of testing resources, the type of content, and the sensitivity of the content, it may be impractical or impossible to migrate all of the production data back to QA. However, in other environments QA must precisely mirror the production environment all the way down to the data that's in use. Whether you choose to do a partial migration or a full migration doesn't impact the approach of migrating from production to QA - and potentially even back to Development if the information isn't sensitive. The rule is primarily that nothing moves to production. If you want to allow your QA environment to get out of sync with production data you can do that.

Required Configuration

Once of the most vocal arguments against this strategy is the problem of code and configuration aligning. Say for instance that I have code that requires that there be a new field in a list in SharePoint. The argument becomes how you get that new field in the SharePoint list and how does that meet up with the code. In most cases, the configuration can be made in production in advance of the code getting there. In other words, the new field can be added into production, but perhaps not have it visible to most users. A new list can be created with permissions so only the site administrators can see it. A whole new site can be created and hidden from the users if necessary. There are a variety of options that are all workable ways to get the configuration into production without impacting the look and feel of production for the users.

What's more tools can be developed to make these configuration changes. Whether they take the form of an additional feature with a feature receiver or are a command line tool that the infrastructure team runs, you don't have to configure things via the user interface just because you can. These tools can then be run in the development environment, again in QA, and when the time comes in production as well.

Initial Load

There's one possible exception to the rule of always copying content back from production. That is, the initial load of the system. While it's still not recommended that you roll forward the content database from development, a roll forward of the QA environment to production for the initial load can make sense if you've been treating QA as a pseudo production while the production environment has been being built or converted.

Code Always Moves from Development through QA to Production

To someone who's been doing configuration management the idea of moving code from development to QA and then once approved to production may seem like an obvious approach, it's not necessarily that obvious to someone that hasn't done configuration management - or has been loose in their practices. In addition to a semi-rigid approval process for code which requires a stop in QA and sign off that the code is ready for production, I generally recommend that all code be delivered as a SharePoint Solution package. The SharePoint Solution package, with a .WSP extension, can be atomically deployed to an entire web farm which makes it ideal for the identical delivery of code to QA and to production. There are only two steps for deploying code with a WSP - adding it to the solution store and scheduling the deployment. The ability to avoid a large number of manual steps means that there is less chance for error when moving to production and that's a good thing.

What is Code?

Code is all of the things that you can't do from the user interface or SharePoint designer. Developing web parts is obvious, as are the development of SharePoint Features. However, there are some situations where you'll convert content into code so that it can be universally applied across site collections in the farm. For instance, master pages, content types, and page layouts used for publishing are generally published via a Feature so they're code -- even if they're initially created in the user interface. Here's a list of examples that are always code:

- Workflow Templates (the kind created in Visual Studio 2005)
- Event Handlers
- Navigation Providers

- Web Services
- Site Definitions
- List Definitions
- Web Parts
- Features
- SharePoint Solution Files (WSPs)

What is Content?

Some content is easy to identify. Documents, web pages, and list items are clearly content and should therefore be rolled back from production to other environments. However, what about list instances, lists schemas, web sites, etc? In short, yes, that's content. If you can create it in a web browser or in SharePoint Designer - it's content. It needs to roll backwards from production to QA and possibly to Development. Here's a list of things that are always content:

- Site Collections
- Web Sites (Sites/Webs)
- List and Library Instances including both the schema and the items in it
- Web Content Management Pages
- Workflow Associations (the relationship between a template and a list or content type)
- Workflow Instances (The instance of a workflow template running on an item)
- Workflows created in SharePoint Designer
- User permissions
- Site Templates
- List Templates

Determining the Difference Between Code and Content

You may have noticed that the list of code items and the list of content items don't represent everything that is in SharePoint. There are some major gaps between the two. It's already been mentioned that some items may start out as content and then be converted into code to allow them to be more easily deployed in a consistent manor. Let's take a look at that list:

- Master Pages
- Content Types
- Page Layouts
- Themes
- Web Part Pages
- Navigation

Generally speaking it's a good idea to deploy master pages via Solutions and Features. Content Types -- particularly those used for Web Content Management should also be deployed via Solutions and Features. Page Layouts which are designed for cross site use fall into the category for deployment -- and thus should be treated as code -- as well. Similarly Themes are good candidates for Solution deployment since the point of a theme is a consistent experience between sites.

It does get a bit sticker when we start to talk about Web Part Pages because sometimes you just need another page in the database. In other cases you want the page to appear on a whole set of sites. If it's a one off, the page can certainly stay in the content database as content, however, if it's a part of a solution that already includes code, it might be best to wrap the web part page up into the Solution and Feature and to use that for deployment.

Navigation is the final difficult to classify component of the SharePoint configuration landscape. In many ways, Navigation is built up from the data that's in SharePoint so it's a function of the content. However, it is possible to deploy some navigation via code. Again this would typically happen when you're already deploying code solutions and the objective is to add that code solution to the navigation tree. However, because it's bundled with a new application it's unlikely that anyone would misconstrue one type of navigation for another.

Smashing Code and Content Together

By viewing the problem from the perspective that content rolls backwards from production and code rolls forward from development you have the ability to smash code and content together in a QA environment so you can perform real-world testing before you get into the critical production environment. Being able to find issues in QA will lead to a more stable production environment and fewer emergency deployments of new code.

About The Author

Robert Bogue, MCSE (NT4/W2K), MCSA:Security, A+, Network+, Server+, INet+, IT Project+, E-Biz+, CDIA+, is president of Thor Projects LLC, which provides SharePoint Consulting services to clients around the country. He has contributed to more than 100 book projects and numerous other publishing projects. His latest book is The SharePoint Shepherd's Guide for End Users. (You can find out more about the book at www.SharePointShepherd.com.)

Bogue has been part of the Microsoft Most Valuable Professional (MVP) program for the past 5 years. He was most recently awarded for Microsoft Office SharePoint Server. Before that, Bogue was a Microsoft Commerce Server MVP and Microsoft Windows Servers-Networking MVP.

Bogue runs the SharePoint Users Group of Indiana (SPIN, www.spindiana.com), and he is also a member of the steering committees for the Indiana Windows Users Group and Indianapolis .NET Developer Association. In addition to speaking at local and regional events, Bogue speaks at national conferences. He blogs at www.thorprojects.com/blog , and you can reach him at Rob.Bogue@thorprojects.com .