Strategies For Editing DSL Scripts in Production

By Oren Eini writing as Ayende Rahien, author of *DSLs in Boo*

Once the application is in production, how are users supposed to edit the DSL scripts? It's likely that some important behaviors are defined in the DSL and that users will want to make modifications to them. This is a question that comes up often. Yes, the DSL is easy to change, but how should you deal with changes that affect production? In this article, based on chapter 12 of DSLs in Boo, the author discusses strategies for editing DSL scripts in production.

You may also be interested in…

Let's first assume we're talking about a web application or a backend system, not a Windows application. There are several aspects to this problem. First, there is the practical matter of creating some sort of UI to allow users to make the changes. This is generally not something trivial to produce as part of the admin section of an application.

There are also many other challenges in this scenario that need to be dealt with, such as handling frequent changes, cascading updates, debugging, invasive execution, error handling, and so on. Just making sure that all the scripts are synchronized across all the nodes in a web farm can be a nontrivial task. You also have to deal with issues such as providing auditing information, identifying who did what, why, and when, and you need to be able to safely roll back a change.

In development mode, there is no issue, because you can afford to be unsafe there. The worst thing that can happen is that you'll need to restart the application or revert to a clean checkout. For production, this isn't an option. This is not a simple matter. My approach, usually, is to avoid this requirement as much as possible. I don't allow such changes to be made in production. It's still possible, but it's a manual process that's there for emergency use only. Like the ability to log in to the production database and run queries, it should be avoided if possible.

But disallowing changes isn't always possible. If the client needs the ability to edit DSL scripts in production, you need to provide a way for them to do so. What I have found to be useful is to provide a way to not work directly on production. Instead, we work on scripts stored in a source control server that is considered part of the application itself. You can see how it works in figure 1.

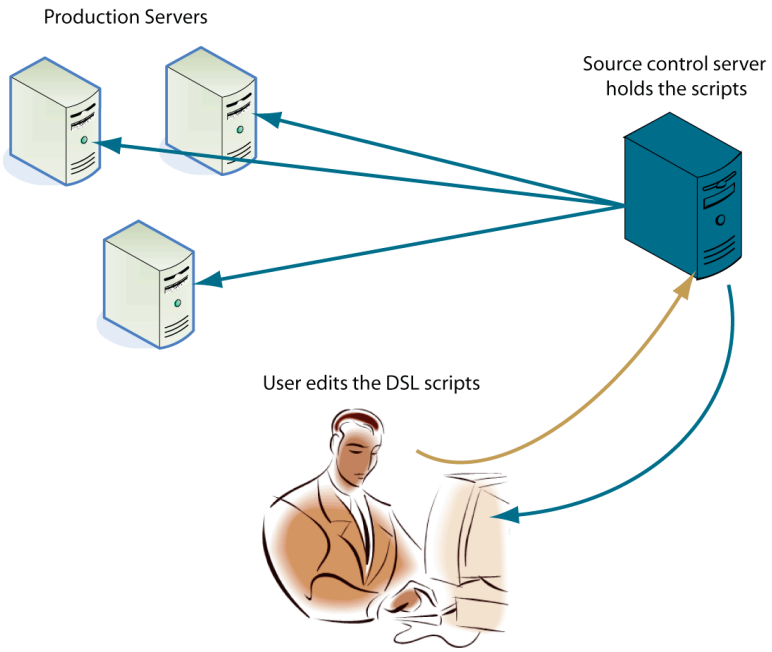


Figure 1 Using a source control server to hold the scripts for the system offers big benefits, such as authorization, auditing, and distribution.

If you want to access the scripts, you check them out of source control. Then you can edit them with any tool you want (often the same tools that you use during development) and finish by committing them back to the repository. The application monitors the repository and will update itself when a commit is done to the production branch.

This approach has several big advantages. First, you don't have the problem of partial updates, you get a pretty good audit trail, and you have built-in reversibility. In addition to that, you avoid the whole problem of having to build a UI for editing the production scripts; you use the same tools that you used during development. As a side benefit, this approach also takes care of pushing script changes to a farm, so you don't have to provide a separate solution for that. And yes, this method of editing scripts in production incorporates continuous integration as part of the application.

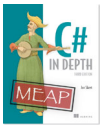
Summary

In this quick tip, we talked about managing deployment and ensuring that we have sufficient control over how we deploy new DSL scripts to production.

Here are some other Manning titles you might be interested in:

Dependency Injection in .NET
Mark Seemann

Windows Phone 7 in Action
Timothy Binkley-Jones, Massimo Perga and Michael Sync



C# in Depth, Third Edition
Jon Skeet

Last updated: 8/2/13