Article
Bill McCormick · Dec 7, 2015 4m read

Source Control Hooks and Atelier

So another topic that has been of interest to a number of people since the beta was put up last week is in regard to Studio Hooks and Atelier. This requires a bit of background and then some discussion of how the architecture of Atelier necessitates some changes and then what our current thinking on the subject is.

Studio hooks were added to Cache over 10 years ago. They have two primary usages. The first is that on an appropriate action such as creating a new document, or compiling a document a callback is executed whose job is to interact with external source control systems. This is done by writing COS scripts that " do the right thing " for a given source control system based on the callback invoked. All of this code ran on the server and used all the facilities available on the server such as locks and transactions etc.

All ISC provided for this was the presence of the callbacks. We did not implement, recommend or support any specific source control system. Over the years a few things changed in this area. Because we were a Perforce shop ourselves a perforce sample of what a Source Control implementation would look like was added to Samples as guidance on how a customer could roll their own. And over time SEs contributed two other kits due to their popularity in the field - Subversion and Team Foundation from MS.

This lack of a proper integration with Source Control has been one of the major deficiencies of Studio and a source of constant pain and frustration for customers in the field for the last 5-8 years. Telling customers they had to roll their own source hooks was met with derision.

However a second use case for Source Control Hooks came to light over this time as well. This was the idea that because we were giving you a callback at the point of compiling code that people had a handy point for implementing coding " rules ". You could evaluate the code and reject it if it doe not meet code quality constraints. You could even do " find and replace " work at the point of saving and compiling. We have a surprisingly large number of customers who are using Studio Hooks in this manner.

One of the major impetuses for moving to Eclipse was changing the Source Control story. We needed to give people a working out of the box solution that did not require customers to write a line of code to get their source to and from the source control systems that the market is using today. Eclipse has built in support for Git and has 27 freely available plugins for 27 different source control systems including all of the major platforms that our customers have inquired to us about in the past.

Of course this is not a straightforward issue. Eclipse does all its source control interactions on the client. We can support this easily out of the box and we do. The problem is what can we do for customers that invested in Studio Hooks. Atelier has no notion of locking. It uses the server merely as a compiler. In fact it is REST based and decoupled entirely from the server except at compile time or when debugging. We also do not have the ability to let users add UI to Atelier that is integrated with the server as we could in Studio. All source integration in Atelier happens through the " Team " framework on the client. So let ' s talk about what we did do, and what might be possible down the road.

First and most importantly we do support Studio Hooks. If you are running Atelier against a namespace where Studio Hooks have been activated it will do the right thing and your callbacks will be triggered at the appropriate times. However this behavior will be atomic as opposed to in Studio. If I open a class in an Atelier project, nothing will happen on the server as we have not connected to the server yet. When I go to compile that class though three actions will happen concurrently - an Open, a Compile/Save, and a Close all in the process of one operation. So the class or routine would get checked out, would get updated and would get committed back to source control, all within one call to the server. If your hooks included logic to " beautify " or validate code that would still happen as well. In this regard we " support " Studio Hooks. However there are no menus in Atelier for " adding " a document to source or adding other UI elements you may have bound to Studio Hooks in Cache Studio. We have not and will

not be implementing those elements at this time.

Why you might ask? Because to interact in that manner in Atelier would mean that essentially we have to write a complete Team Framework plugin for Eclipse ourselves that was a plugin for the Studio Hooks Source Control system. Except of course there is no Studio Hooks Source Control system. All it does is move the job back to the server, where we leave the actual task of implementing source control…. right back in the lap of our customers where we already agreed it never belonged. Essentially we have to invest resources that could be used to make the Atelier experience and the development experience with Cache better in building a plugin whose only purpose is to get you back in to the dead end from which we are trying to liberate you.

Needless to say we don ' t like losing any feature or adding any compatibility issue to Cache. And we know a lot of people invested in Studio Hooks to even construct applications. This is simply to explain what the story is today as we go to beta with Atelier. I look forward to your comments and criticisms and perhaps even proposals for solutions that I may not have considered here. What I can tell you is to try a plug in for your source system of choice. I think like me you will find going back awfully hard to do once you have seen how seamlessly they integrate and enhance your experience with Eclipse. I have played with Git, Perforce and TFS and can vouch that all of them work perfectly with Atelier. Happy Coding!

#Compiler #Development Environment #Atelier