Discussion
Nikita Savchenko · Dec 12, 2019

Package Manager for InterSystems: the Design Question to All Engineers

Hello, InterSystems community!

Lately, you have probably heard of the new InterSystems Package Manager - ZPM. If you're familiar with it or with such package managers as NPM, Dep, pip/PyPI, etc. or just know what is it all about -- this question is for you! The question I want to arise is actually a system design question, or, in other words, "how should ZPM implement it".

In short, ZPM (the new package manager) allows you to install packages/software to your InterSystems product in a very convenient, manageable way. Just open up the terminal, run ZPM routine, and type install samples-objectscript: you will have a new package/software installed and ready to use! In the same way, you can easily delete and update packages.

From the developer's point of view, quite as same as in other package managers, ZPM requires the package/software to have a package description, fairly represented as module.xml file. Here's an example of it. This file has a description of what to install, which CSP applications to create, which routines to run once installed and so on.

Now, straight to the point. You've also probably heard of InterSystems WebTerminal - one of my projects which is quite widely used (over 500 installs over the last couple of months). We try to bring WebTerminal to ZPM.

So far, anyone could install WebTerminal just by importing an XML file with its code - no more actions were needed. During the class compilation, WebTerminal runs the projection and does all required settings on its own (web application, globals, etc - see here). In addition to this, WebTerminal has its own self-updating mechanism, which allows it to self-update when the new version comes out, made exactly with the use of projections. Apart from that, I have 2 more projects (ClassExplorer, Visual Editor) that use the same import-and-install convenient installation mechanism.

But, it was decided that ZPM won't accept projections as a paradigm and everything should be described in module.xml file. Hence, to publish WebTerminal for ZPM, the team tried to remove Installer.cls class (one of WebTerminal's classes which did all the install-update magic with the use of projections) and manually replaced it with some module.xml metadata. It turned to be quite enough for WebTerminal to work but it potentially leads to unexpected incompatibilities to be 100% compatible with ZPM (see below). Thus, the source code changes are needed.

So the question is, should ZPM really avoid all projection-enabled classes for its packages? The decision of avoiding projections might be changed via the open discussion here. It's not a question of why can't I rewrite WebTerminal's code, but rather why not just accept original software code even if it uses projections?

My opinion was quite strong against avoiding projection-enabled classes in ZPM modules. For multiple reasons. But first of all, because projections are the way how the programming language works, and I see no constructive reasoning against using them for whatever the software/package is designed for. Avoiding them and cutting Installer.cls class from the release is absolutely the same as patching a working module. I agree that the packages which ship specifically for ZPM should try to use all installation features which module.xml provides, however, WebTerminal is also shipped outside of ZPM, and maintaining 2 versions of WebTerminal (at least, because of the self-update feature) makes me think that something is wrong here.

I see the next pros of keeping all projection-enabled classes in ZPM:

- The package/software will still be compatible with both ZPM and a regular installation done for years (via XML/classes import)
- No original package/software source code changes needed to bring it to ZPM
- All designed functions work as expected and don't cause problems (for instance, WebTerminal self-updates - upon the update, it loads the XML file with the new version and imports it, including projection-enabled Installer.cls file anyway)

Cons of keeping all projection-enabled classes in ZPM:

- Side effects made during the installation/uninstallation, made by projection-enabled classes won't be statically described in the module.xml file, hence they are "less auditable". There is an opinion that any side effect must be described in module.xml file.

Please indicate any other pros/cons if this isn't the full list. **What do you think?**

Thank you!

#InterSystems Package Manager (IPM) #ObjectScript #Caché #InterSystems IRIS