Generic variance in DI containers

Posted by Jimmy Bogard on January 13, 2015

DI containers, as complex as they might be, still provide quite a lot of value when it comes to defining and realizing the composition of your system. I use the variance features quite a bit, especially in my MediatR project andcomposing a rich pipeline. A side note, one of the design goals of MediatR is not to take any dependency on a 3rd party DI container. I instead take a dependency on Common Service Locator, which all major DI containers already have. As part of this exercise, I still wanted to provide examples of all major containers, and this led me to figure out which containers supported what.

I looked at the major containers out there:

- Autofac
- Ninject
- Simple Injector
- StructureMap
- Unity
- Windsor

And tried to build examples of using MediatR. As part of this, I was able to see what containers supported which scenarios, and how difficult it was to achieve this.

The scenario is this: I have an interface, IMediator, in which I can send a single request/response or a notification to multiple recipients:

I then created a base set of requests/responses/notifications:

I was interested in looking at a few things with regards to container support for generics:

- Setup for open generics (registering IRequestHandler<,> easily)
- Setup for multiple registrations of open generics (two or more INotificationHandlers)
- Setup for generic variance (registering handlers for base INotification/creating request pipelines)

My handlers are pretty straightforward, they just output to console:

I should see a total of seven messages output from the result of the run. Let's see how the different containers stack up!

Autofac

Autofac has been around for quite a bit, and has extensive support for generics and variance. The configuration for Autofac is:

Autofac does require us to explicitly add a registration source for recognizing contravariant interfaces (covariant is a lot rarer, so I'm ignoring that for now). With minimal configuration, Autofac scored perfectly and output all the messages.

Open generics: yes, implicitly

Multiple open generics: yes, implicitly

Generic contravariance: yes, explicitly

Ninject

Ninject has also been around for quite a while, and also has extensive support for generics. The configuration for Ninject looks like:

Ninject was able to display all the messages, and the configuration looks very similar to Autofac. However, that "ContravariantBindingResolver" is*not* built in to Ninject and is something you'll have to spelunk Stack Overflow to figure out. It's somewhat possible when you have one generic parameter, but for multiple it gets a lot harder. I won't embed the gist as it's quite ugly, but you can find the full resolver here.

Open generics: yes, implicitly

Multiple open generics: yes, implicitly

Generic contravariance: yes, with user-built extensions

Simple Injector

Simple Injector is a bit of an upstart ~~from the same folks behind NancyFx~~someone not related to NancyFx at all yet has a very similar Twitter handle, and it focuses really on the simple, straightforward scenarios. This is the first container that requires a bit more to hook up:

~~While multiple open generics is supported, contravariance is not. In fact, to hook up contravariance requires quite a few hoops to jump through to set it up. It's documented, but I wouldn't call it "out of the box" because you have to build your own wrapper around the handlers to manually figure out the handlers to call~~. UPDATE: as of 2.7, contravariance *is* supported out-of-the-box. Configuration is the same as it is above, the variance now "just works".

Open generics: yes, explicitly

Multiple open generics: yes, explicitly

Generic contravariance: ~~no~~ yes, implicitly

StructureMap

This is the most established container in this list, and one I've used the most personally. StructureMap is a little bit different in that it applies conventions during scanning assemblies to determine how to wire requests for types up. Here's the StructureMap configuration:

I do have to manually wire up the open generics in this case.

Open generics: yes, explicitly

Multiple open generics: yes, explicitly

Generic contravariance: yes, implicitly

Unity

And now for the most annoying container I had to deal with. Unity doesn't like one type registered with two implementations, so you have to do extra work to even be able to run the application with multiple handlers for a message. My Unity configuration is:

Yikes. Unity handles the very simple case of open generics, but that's about it.

Open generics: yes, implicitly

Multiple open generics: yes, with user-built extension

Generic contravariance: derp

Windsor

The last container in this completely unnecessarily long list is Windsor. Windsor was a bit funny, it required a lot more configuration than others, but it was configuration that was built in and very wordy. My Windsor configuration is:

Similar to Ninject, the simple scenarios are built-in, but the more complex need a bit of Stack Overflow spelunking. The "ContravariantFilter" is very similar to the Ninject implementation, with the same limitations as well.

Open generics: yes, implicitly

Multiple open generics: yes, implicitly

Generic contravariance: yes, with user-built extension

Final score

Going in, I thought the containers would be closer in ability for a feature like these that are pretty popular these days. Instead, they're miles apart. I originally was going to use this as a post to complain that there are too many DI containers in the .NET space, but honestly, the feature set and underlying models are so completely different it would take quite a bit of effort to try to consolidate and combine projects.

What is pretty clear from my experience here is that Unity as a choice is probably a mistake.