

Container Design

This page contains some of my random thoughts on what using a container could look like in geotools.

Preamble

First I think its good to seperate out two main concerns / requirements.

1. Dynamically load instances of specific classes (ie. plugins)
2. Managing dependencies among classes in Geotools.

The first requirement is fulfilled with the FactorFinder system. Which I must say for most applications works relativley well. However it fails when you get into environments / applications which do fancy class loading tricks like breaking out a seperate classloader for each plugin / module. Examples are Eclipse and Geronimo.

The second concern has really not been addressed yet. However luckily the rest of the world has already solved this problem with the notion of a `Container`. Something that hosts objects providing services such as life cycle and dependency management. For those of you not convinced that a container is actually needed, here are some concrete requirements that people have and are failing to be addressed.

1. Supplying an alternate implementation of a JDBC connection pool to PostGIS data store (Geoserver, J2EE)
2. Creating optimized instances of of Geometry or Feature to be used simply for rendering purposes (UDIG)

What these really boil down to is the ability to inject objects in our toolkit with alternate implementations of classes or interfaces that they depend on. Why is this hard? Well in a layered system like geotools, funtionality and api is separated into layers.

⚠ Jody: need an architecture diagram here

Any client of the Geotools toolkit really only gets access to the API in the top layer, making it hard to configure behaviour at lower layers. This is really an issue with any layered system.

So you might be saying, how can a magic container fix this for me. And you are right, a container can't do it alone. Really the first step in handling this problem is designing classes and packages and subsystems to be container friendly. Now different containers offer dependnecy management in different ways but a fairly common theme is the notion of [Inversion of Control](#). This simple concept really is just saying that you the designer of a class are not going to make any assumptions about which instance of a class to create, let the client do that. The client can then use a container to make this process easy.

Geotools Container

What would a container system look like in geotools?

🔍 Do we come up with our own interface, or use an existing one?

Do we come up with our own container interface? Something that might look like the following:

```
interface Container {  
  
    /**  
     * Register a particular implementatin of  
     a class or interface.  
     */  
    void register(Class clazz);  
  
    /**  
     * Load a particular interface of a class  
     or interface.  
     */  
    Object load(Class clazz);  
  
}
```

An implementation of this interface could be written for a particular container implementation, PicoContainer, Spring, etc... The problem with this is we have to roll our own api for registering and loading class implementations.

The alternative would be to just stick with an implementation that is already out there. A popular choice is PicoContainer which is extremley lightweight and easy to use. The problem with this is that it ties us to a particular container.