

Events for Styles

Ideas

Right now it is very hard to track changes that are made to objects in a style tree. There are a number of solutions to this:

- 1) Modify all the existing styling objects to have an event mechanism (probably one that notifies parent objects)
- 2) Redesign the styling architecture to use more generic nodes, cutting down on the number of unique objects and simplifying the addition of a style system
- 3) Leave everything as it is but use crazy dynamic proxy code to add events into the existing structure.

First Attempt!

James - My current approach is No 3). It may not be the best (2 probably is) but is easy to implement and will result in a backwards compatible solution. Questions remain about how easy it would be to use and, critically, what the performance cost would be.

A branch to do this has been created (At revision: 15806) to evaluate its practicality and performance. Details to follow...

Jody - your branch was amusing (he made a DynamicProxy that added PropertyChanged events to every set method)!

Second Attempt! Success

Problems

Before we could even start we ran into a number of problems:

- Style Interface/Factory/FactoryFinder breakdown not complete (or consistent)
- GeoAPI uses a unpublished schema 1.0.20 and thus has different names (what foul manner of magic is this?)
- Requests for Custom Hacks (like TextSymbolizer.getGraphic()) needed to be out of the way of SLD conformant interfaces

Style Interface/Factory/FactoryFinder

This was a simple mistake of the undone kind, everything is set up as expected now.

What is expected? Here is an article:

- [Objects, Interfaces and Factories](#)

DONE:

- Created StyleFactoryFinder
- Set up interfaces in main/api
- Replaced all uses of StyleFactoryFinder.create
- Confirmed conformance to SLD 1.0 specification (we were at SLD 0.72 or something)
- Set up StyleFactory, StyleBuilder, FilterFactory, FilterFactoryFinder to be ready for constructor and setter injection

TODO:

- FactoryFinder uses direct cut and paste of previous code, should be updated with respect to non deprecated code from referencing

Exploring Support for multiple versions

Wrote an article on supporting multiple versions:

- [Supporting Hacks and Versions](#)

Here is what the ideal looks like:

```

LineSymbol <>--- Stroke (geoapi)
      ^                ^
      |                |
LineSymbolizer <>--- Stroke (geotools)

```

And here is what we would need todo with Java 1.4:

```

LineSymbol <>--- Stroke (geoapi)
      ^                ^
      |                |
LineSymbolizer      Stroke (geotools)

```

That is we cannot indicate from our GeoTools interfaces that subclasses returned are explicitly of a known GeoTools type. **We are stuck with lowest common demoninator** aka a lot of instance of checks.

The best advice was to leave things alone and mark new methods with javadocs.

We can still set up Factory classes that are restricted to constructing things within the bounds of their specifications.

Conclusion their is not much to be done until we have Java 5 with type narrowing...

DONE:

- We have provided method names congruent with GeoAPI where possible, so just changing imports will be enough to fix code when the time comes.

TODO:

- deprecate duplicated names
- enquire about move to lists

- Ask GeoAPI to rollback to SLD 1.0 support?

Requests for Custom Hacks

This is a related problem to that above, but we can have more success because no interaction with GeoAPI is needed.

```
TextSymbolizer
```

```
^
```

```
|
```

```
TextSymbolizer2
```

This is consistent w/ Java use, interested Renderers can use an instanceof check and a cast.

Events!

We grabbed the even system from Catalog and made a copy, changing everything over to Style. StyleLayerDescriptor has the add/remove listener code. It has an event system suitable to changes in a tree structure, in particular it is amiable to "batched changes" such as would be produced by a single transformation performed by a StyleVisitor.

Part of the fun of setting up for the above ease of use is this - the need for a getParent() in order to pass the change notification up the tree. To make this easy we set up a StyleComponent and implemented everything once...

Thinking ahead we decided to rip everything out into org.geotools.event so this system can be reused with Filter and Catalog. We did need to isolate the API for getParent/setParent/notification to an interface for this technique to work. (The driving story here being someone wanting to implement a custom Stroke, and making sure they can still hook into everything w/ out breakage). GTComponent is thus not API for client code, it is API for people extending GeoTools.

Finally we made a GTList (subclass of ArrayList) that knows how to pass along events when the list is changed. A smooth move that lets events have very little impact on the Style implementations.

DONE:

- Events

TODO:

- Advertise lists directly, switching over to GeoAPI methods would accomplish this - for now we simply use rules() (and maintain the array based methods to preserve compatibility).