Process API

The process api is used to capture GeoTools processes so they can be run in a nice threadsafe fashion
Table of Contents:

Sub pages;

Reference:

- Process Module Matrix Page
- GEOS-4706 Cut over to GeoTools port of GSProcess implementations

Overview

The GeoTools Process API has been defined as:

- a data structure used to describe the process parameters; expected results
- a set of support classes used to run a process in a threadsafe manner and report on progress for long running processes
- a set of java annotations used to mark up methods so they are available to the process module

Supported

Here is the plan for taking the gt-process concepts to supported status.

1. setting the stage
   - gt-process split out into different modules (gt-process-feature, gt-process-geometry, gt-process-raster)
2. backport annotations from geoserver along with supporting infrastructure
   - Review AnnotationDrivenProcessFactory, AnnotationBeanProcessFactory and StaticMethodsProcessFactory
   - Rename the internals of AnnotationDrivenProcessFactory for InvokeMethodProcess and InvokeMethodRenderingProcess
   - Refactor recognising of RenderingProcess stuff to allow for static methods to support RenderingProcess
   - Add lots of javadocs with examples
   - Q: What to do about the marking interface GeoServerProcess? GSProcess (for "geospatial") giving Spring something fun to pick up and have a party with
   - Update annotations; and marker interface
3. gt-process
   - redistribute existing geotools "examples"
   - Made ProcessException extend RuntimeException (so that transition from WPSException would be easier)
   - Remove Examples of API that do not accomplish anything (dem, buffer single feature, addition, union, intersection)
4. gt-process-feature
   - populated with a lot of GSProcess implementations
   - dependency on gt-grid
   - back ported as many test cases as I could

    - grabbed a bugsites.property file for testing; along with a test dependency on gt-process and gt-epsh-gsql
    - Add enough tests to allow the module graduation into supported land
5. gt-process-geometry
    - populated with the GeometryProcessFactory from geoserver
    - test case for GeometryProcessFactory
    - Add enough tests to allow the module to become supported (wondering if some reflection against Geometry could allow for faster test buildup)
6. gt-process-raster
    - populated with a lot of GSProcess implementations (very nice clear readable code)
    - pick up a dependency on jai-tools
    - need to sort out how to take on raster test cases and get enough test coverage to move the module to supported land
7. docs
    - advanced tutorial showing how to implement a simple process

Testing ideas

Most of the processes are backported from GeoServer, where they were tested by exercising the WPS api and using the testing facilities provided by GeoServer.

Need to replicate some of that at least partially:

- have a number of datasets that can be used for testing (most of the GeoServer ones are in the main module)
- have base class methods to perform quick checks against feature collections (feature existance, attribute existance, and so on)
- have base class methods to perform checks against coverages, like pixel and band checks

gt-api

Contains the Process interface, ProcessFactory along with DescribeProcess annotations etc...

docs:

- need to be updated to reflect api change
- migrate docs from unsupported/process to library/api
- tutorial: Process simple tutorial showing use of annotation
- tutorial: WPS (Matthias Lendholt) may need to extend the tutorial to at least mention GeoServer PPIO objects (used to handle complex parameters). JG PPIO was a bit of a mistake used to cut corners; easier at the time to push into geoserver as GML only worked there. I have back ported most of the good bits to the GML utility class so perhaps PPIO die?

gt-coverage

RasterToVectory code may wish to be wrapped up as an opperation; and the internals replaced a JAITools operator.

- http://jira.codehaus.org/browse/GEOT-3336
- If this happens gt-process can keep a IProcess wrapper available for udig code

gt-main

Contains the base factory implementations; ThreadPoolProcessExecutor and so on.

gt-process

Contains simple things often working on literals:

- (possible) bridge to geotools functions.
- (possible) bridge to coverage operations

gt-process-geometry

From GeoServer:

- Common JTS methods wrapped up a IProcess / static methods

gt-process-feature

From GeoServer:

- gs:RectangularClip
- cookie cutter
- buffer
- intersection
- overlap
- and so on...
- Add enough tests to

gt-process-raster

- raster to vector
- anything else ...

gt-swing (unsupported)

- Add an "provided" dependency on gt-process (as a temporary measure until process is moved to gt-api)
- Grab the process wizard stuff from 2.5.x
  - http://svn.osgeo.org/geotools/branches/2.5.x/modules/unsupported/widgets-swing-pending/src/main/java/org/geotools/gui/swing/process/

gt-wps (unsupported)

Contains a client allowing web processing service to be used in the same manner as a normal local process

Out of the above list the wps module is the odd one out; there is no way to make that one go to supported status without backing; especially with WPS 2.0 specification on the horizon. If anyone has an interested customer (or employer) they are more than welcome to develop against it.

gt-sextante (unsupported - requires volunteer)

My understanding is that the sextante license has changed; perhaps we could now have a gt-process-sextante plugin in geotools.

> Yep, if someone is interesting in maintaining it there is a set of rough bindings available in GeoServer community land.
> I basically gave up on integrating it because it uses a push model, it's too hard to make it scale to large quantities of data, raster processes do not use tiling for input/output, vector can do streaming only if not chained... I mean, making clever use of thread (one per process), blocking queues, JAI wrappers that call the process tile by tile it might be possible to get it going but it's a lot of work.
> A plain wrapping is easier but would be bound to memory... meh... not exciting at all.

Q&A

Q: What about jgrasstools?

It already supports the process api; it should just need to change its dependencies a bit (to just depend on gt-api and/or gt-main). Although there is probably a license incompatibility.

Q: What about geoserver?

The plugins are optional; my hope is that geoserver could transition to their use (for geometry etc...) without changing anything from the users point of view. While users are warned that in the user guide that these things may change there is no reason to go out of our way to rock the boat.

As such processes that have been prototyped in the geoserver code base should keep their prefix:

- "gs:RectangularClip"
- cookie cutter

When updating GeoServer:

#. change imports to:

```
import org.geotools.process.factory.DescribeParameter;
import org.geotools.process.factory.DescribeProcess;
import org.geotools.process.factory.DescribeResult;
import org.geotools.process.gs.GSProcess;
import org.geotools.process.gs.WrappingIterator;
```

1. Change "WPSException" to "ProcessException"
2. This transition has gone as expected: GEOS-4706 Cut over to GeoTools port of GSProcess implementations

Discussion:

> JG" Is that the ESRI definition of Cookie Cutter?
> AA: We have a clip process optimized for rectangles and a ESRI style polygon overlay that keeps attributes of both intersected featuers in the result. I'm also working to make a more generic clip process that will work with any shape, not just rectangles, but found issues with bbox filters and reprojected feature collections that prevent me from committing it, I guess I need another weekend of work to clean that mess before the process can be committed.

Q: What about uDig?

I am going to push the process user interface into uDig; it already lists any geotools processes in the catalog.

Q: What about gt-swing?

I have missplaced the process wizard and need to find it in an older branch of GeoTools. Michael if the process api is actually added to gt-api could we restore the process wizard to gt-swing? It would not require any additional dependencies

Conversation Points

- Expr Examples - illustrate unified Expr syntax
- Opperations API - IM
- IRC Breakout Number 1
- Cascading Coordinate Systems
- Operations