# Focus FeatureSource around FeatureReader and FeatureWriter

| | |
|---|---|
| **Motivation:** | *The current api is a mess* |
| **Contact:** | Justin Deoliveira |
| **Tracker:** | http://jira.codehaus.org/browse/GEOT-1922 for GeoTools 2.5.x tasks |
| **Tagline:** | Cleaning up data store / feature access api |

This page represents the **current** plan; for discussion please check the tracker link above.

Children:

## Description

### The Problem

Currently our data access api presents two ways to access data:

1. query-based reader/writer access
2. operation-based collection access

While providing two modes of data access does have some advantages in terms of flexibility it has some major problems:

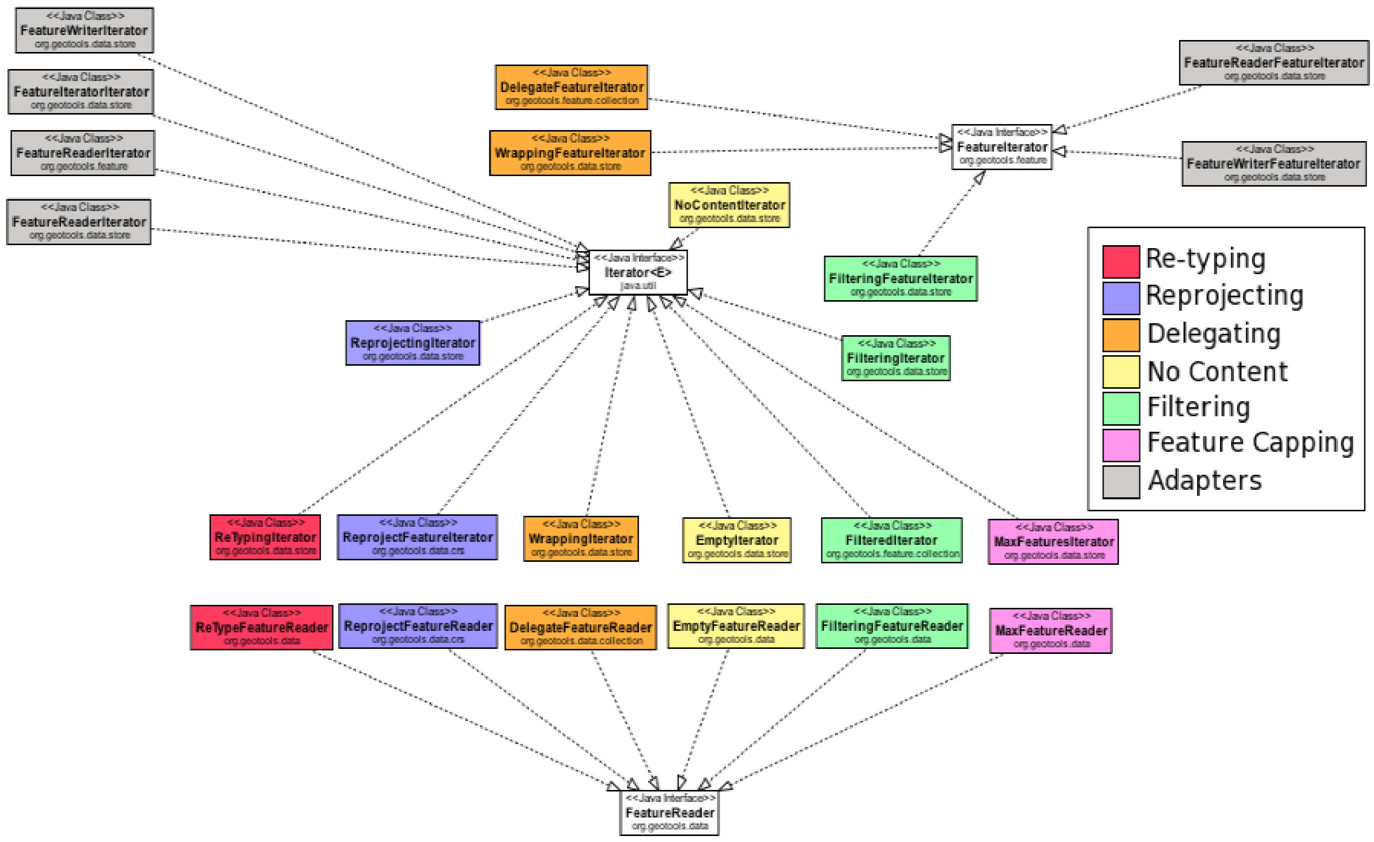
#### API Confusion

Having two apis for accessing data makes it harder to approach for users. Either method does not present any decisive advantage over the other. Its just "more than one way to do it" (to coin a phrase from perl), which has led to more headache than benefit.

#### Duplication

`FeatureCollection` duplicates much of what was originally created in `FeatureReader`/`FeatureWriter`. The biggest violation being iterators. Not only does `FeatureCollection` provide a duplication of `FeatureReader` in the form of `Iterator` from java collections, it goes beyond and defines the `features()` method which returns a `FeatureIterator`, which is literally a one-to-one duplication of `FeatureReader`
.

One of the biggest problems of this duplication has been with resect to *decorators*. Each time a decorator is implemented, it is implemented up to 3 times. To add to the mess a set of *adapters* have been created to go back and forth between the 3 types of iterator. The following diagram illustrates:

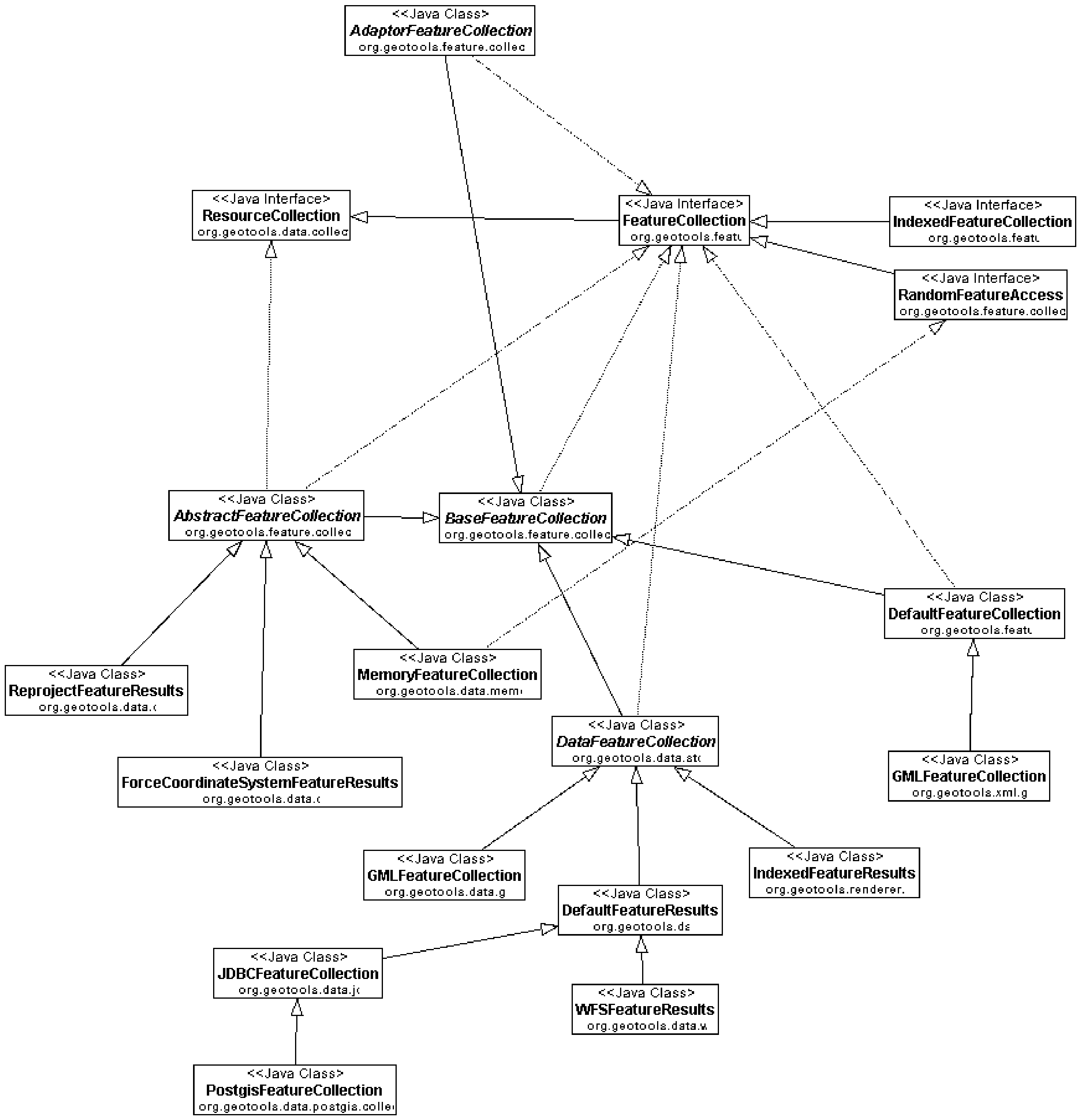


<<Java Class>> FeatureWriterIterator org.geotools.data.store
<<Java Class>> FeatureIteratorIterator org.geotools.data.store
<<Java Class>> FeatureReaderIterator org.geotools.feature
<<Java Class>> FeatureReaderIterator org.geotools.data.store
<<Java Class>> DelegateFeatureIterator org.geotools.feature.collection
<<Java Class>> WrappingFeatureIterator org.geotools.data.store
<<Java Class>> NoContentIterator org.geotools.data.store
<<Java Interface>> FeatureIterator org.geotools.feature
<<Java Class>> FeatureReaderFeatureIterator org.geotools.data.store
<<Java Class>> FeatureWriterFeatureIterator org.geotools.data.store
<<Java Interface>> Iterator<E> java.util
<<Java Class>> FilteringFeatureIterator org.geotools.data.store
<<Java Class>> ReprojectingIterator org.geotools.data.store
<<Java Class>> FilteringIterator org.geotools.data.store
<<Java Class>> ReTypingIterator org.geotools.data.store
<<Java Class>> ReprojectFeatureIterator org.geotools.data.crs
<<Java Class>> WrappingIterator org.geotools.data.store
<<Java Class>> EmptyIterator org.geotools.data.store
<<Java Class>> FilteredIterator org.geotools.feature.collection
<<Java Class>> MaxFeaturesIterator org.geotools.data.store
<<Java Class>> ReTypeFeatureReader org.geotools.data
<<Java Class>> ReprojectFeatureReader org.geotools.data.crs
<<Java Class>> DelegateFeatureReader org.geotools.data.collection
<<Java Class>> EmptyFeatureReader org.geotools.data
<<Java Class>> FilteringFeatureReader org.geotools.data
<<Java Class>> MaxFeatureReader org.geotools.data
<<Java Interface>> FeatureReader org.geotools.data
Re-typing
Reprojecting
Delegating
No Content
Filtering
Feature Capping
Adapters

## Implementation Complexity

Having two data access apis to implement makes it very hard to create a clean datastore, which ultimately leads to a high cost of maintainence. This had led to a mass of unsupported format drivers in Geotools. With a better organization and internal api for implementing datastores, many of these drivers could be saved.

Data store implementors having to implement `FeatureCollection` is something that has been a disaster. The following diagram illustrates:



## The Solution

The approach presented in this proposal is two fold:

- focus the `DataStore`/`FeatureSource` api around `FeatureReader` and `FeatureWriter`
- make `FeatureCollection` a convenience wrapper around `DataStore`/`FeatureSource`

The benefits of this are:

- 100% backwards compatable
- Implementors can ignore `FeatureCollection` and implement the simpler `FeatureReader`/`FeatureWriter` api
- `FeatureCollection` gets implemented **once** and all duplication is removed

The modifications to the current datastore api are strictly additions. Consider the following:

```
interface DataStore2 extends DataStore {
  FeatureSource getFeatureSource( String typeName, Transaction tx );
  FeatureSource getFeatureSource( Name typeName, Transaction tx );
}
```

```
interface FeatureSource2 extends FeatureSource {

  /** get a reader over the entire set of features */
  FeatureReader getReader();

  /** get a reader over a subset of features */
  FeatureReader getReader( Filter filter );
  FeatureReader getReader( Query query );

}
```

```
interface FeatureStore2 extends FeatureStore {

  /** get an inserting + updating writer over the entire set of features */
  FeatureWriter getWriter();

  /** get an inserting + updating writer over a subset set of features */
  FeatureWriter getWriter( Filter filter );
  FeatureWriter getWriter( Query query );

  /** get an updating writer */
  FeatureWriter getWriterUpdate();
  FeatureWriter getWriterUpdate( Filter filter );
  FeatureWriter getWriterUpdate( Query query );

  /** get an inserting writer */
  FeatureWriter getWriterInsert();

}
```

As for `FeatureCollection`, with this additional api it is possible to create a single implementation which delegates to the `DataStore` and `FeatureSource` api. Such an implementation can be found here.

The `ContentDataStore` class is an experimental abstract datastore implementation which is based on the api changes presented in this proposal. It is the base of the new jdbc datastore which has been used in GeoServer and is fully cite compliant (so we know it works).

## Status

This proposal is under construction.

Voting has not started yet:

- Andrea Aime
- Ian Turton
- Justin Deoliveira
- Jody Garnett
- Martin Desruisseaux
- Simone Giannecchini

## Tasks

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | no progress | ✓ | done | ✗ | impeded | ⚠ | lack mandate/funds/time | ? | volunteer needed |

GeoTools 2.5

- ✓ feature collection is no longer a Java Collection
- ✓ http://jira.codehaus.org/browse/GEOT-1922

GeoTools 2.7.x

- Review the FeatureCollection implementation in H2 datastore that completly delegates to FeatureSource
- Copy the FeatureSource implementation from H2 into data store modules one at a time