Caching Module

Overview

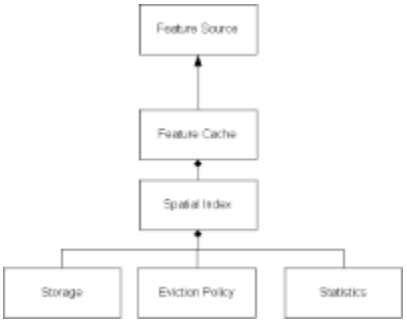
This module provides an implementation of a feature cache.

It started out as a Google Summer of Code project:

- http://docs.codehaus.org/display/GEOTOOLS/SoC+2007+Caching+data
- SoC 2007 Caching data for more info

It has since been revived to be used in a WFS caching datastore for uDig.

Class Diagram



Feature Cache Implementations

There are currently two implementations of feature cache:

GridFeatureCache

- This was the initial implementation for the FeatureCache.
- Based on using a GridSpatialIndex (see below).
- It blocks the entire cache while collecting features from the original feature source and saving them to cache.
- When features are requested the features from the cache and the source are combined into a single in-memory feature collection and returned.
- Feature Cache has a maximum number of features which can be stored in it. If this limit is reached then items are evicted from the cache. (If this limit is set to Integer.MAX_VALUE then all features are cached).

StreamingGridFeatureCache

- This implementation improved on the initial GridFeatureCache.
- Only the nodes which are currently being read or written to are locked. This allows multiple threads to be reading/writing from the cache at the same time.
- When features are requested the request returns a FeatureCollection that only reads the features as they are requested. (No in-memory feature collections are used). This FeatureCollection is also responsible for caching the features as they are read from the feature source.

Spatial Index Implementations

GridSpatialIndex

- Currently this is the only spatial index implementation.

- This index works by building a grid the encompasses the bounds of the feature source dataset (or the bounds provided by the user). The size of the index (number of grid cells) is user customizable.
- Each grid cell (Node) contains a collection of data as well as a state (valid or not valid). Valid nodes are nodes which have all there data (or are in the process of writing data to them). Invalid nodes are nodes whose data has not yet been cached.
- Features are added to all nodes they intersect which one exception - if a feature intersects more than 4 nodes the feature is added to the "root" node.
- Statistics are kept about the number of items in the cache; the access and evictions.

Eviction Policy

LRUEvictionPolicy

- Evicts nodes based on the Least-Recently-Used policy. Note that nodes that are currently locked for reading/writing are not evictable.

Storage Implementations

There are currently three storage implementations:

MemoryStore

- Stores all data in memory.

DiskStore

- Stores index to disk.
- Two files are used - one file to store the nodes in pages; another file to store the page index.
- Each time a node is accessed it is written/read from disk.

BufferedDiskStore

- Extension to the DiskStore that "caches" the nodes so the are not constantly accessing the disk.

Usage

So you have a feature source that you want to cache . . .

A FeatureCache can be made using:

```
FeatureCache cachingFS = new StreamingGridFeatureCache(featureSource, indexcapacity, featurecapacity, storage);
```

- featureSource - the original feature source
- indexcapacity - the desired size of the index (in the case of the grid spatial index this is the total number of grid cells)
- featurecapacity - the maximum number of features that can be stored in the cache. Integer.MAX_VALUE will cache all features
- storage - the store to use for the cache

The above constructor makes use of the bounds of the featureSource for determing the bounds of the cache. If you wish to provide your own bounds for the cache you can do so using the following:

```
FeatureCache cachingFS = new StreamingGridFeatureCache(featureSource, bounds, indexcapacity, featurecapacity, storage);
```

Re-using the Cache

A cache store can be re-used and providing the feature types are the same the features from the store will be used. For example if you are using this within uDig; cache a bunch of WFS features; restart uDig; the feature cache will still be used to serve up features.

Clearing the Cache

Currently there is no policy for checking if features in the cache are valid. Features in the cache do not expire. However the feature cache does listen to feature events and if a feature is added or removed from the cache the nodes that correspond to that feature event are invalidated and cleared. (Thus next time features from those nodes are requested it will pass the request along to the original feature source).

The cache can be cleared manually using:

```
FeatureCache.clear()
```