Add hints support to Query object

Motivation:	Allow result objects customization and general query behaviour settings
Contact:	Andrea Aime
Tracker:	http://jira.codehaus.org/browse/GEOT-1291
Tagline:	Query,Hint,Generalization,Fetch

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

Motivation

Allow result objects customization and general behaviour settings

Current Query execution cannot be customized in any significant way, yet there are various situations where this would be useful, for example:

- if a data store can perform efficient generalization on a remote host, it should do so, to decrease the amount of data to be transferred
- if results aren't involved in topologic operations, standard JTS coordinate sequences are overkill
- if the datastore is a jdbc one, fetch size could be fine tuned to achieve better performance

To allow for a variety of variant behaviours we could add a way to specify hints during DataStore requests, that DataStore implementors can try to honor, or simply ignore, as they see fit according to the available capabilities. This would allow for gradual pick up of extra functionality, as well as datastore specific hints.

Status

Proposal passed, implementation pending.

Voting status:

- Andrea Aime +1
- Ian Turton
- Justin Deoliveira +1
- Jody Garnett +1
- Martin Desruisseaux
- Simone Giannecchini +1

Discarded alternatives

Two classes seem to provide an initial support for this kind of idea:

- Transaction has support for generic property objects, which can be used to customize the behaviour of datastores using it (at the moment a code inspection with Eclipse suggests this support is unused)

Transaction does not seem to fit the bill for a few reasons (all of these can be cured thought):

- Transactions are usually associated with write operations, or a mix of read/write ones, but not with pure read only ones (this can be easily overcome with some documentation).
- Read only operations are performed against the AutoCommitTransaction, that does not support property management (will throw an UnsupportedOperationException).
- Transaction it's global to a set of interactions with one datastore (or possibly, multiple datastores), but different requests in the same transaction may need to enjoy different parameters. This could be mitigated if the parameters are modified during execution thought.
- Transactions cannot be used in read only accesses since FeatureSource does not have transaction getters and setters (this is a real issue that should be handled separately, since a user could legitimately want to specify a transaction in order to access the state of features that have been modified during a specific transaction).

Proposal

Hints already has the idea of a set of well known hints, in this case, to be provided to GT2 factories.
Hints looks fine, seems to be easily bent to current needs. That class could host the general, datastore independent hints, whilst the specific hints could be specificied as constants in the DataStore class.

To use Hints, the Query class will have to be extended with two methods:

```
  void setHints(Hints hints);
  Hints getHints();
```

Since hints are optional, there should be a way to tell which hints have been used for real.
The FeatureSource interface shoud also be changed to provide a list of supported hints (which would be independent of the query being made, but may be dependent on the feature type and eventually the current transaction, that is, it may depend on FeatureSource current state). Note that a hint being supported only means that the FeatureSource is able to handle it, but not that it will use it for good. At the moment there is no way to check if the hint was honored or not, the user will have to resort to custom tests to assess the hints was supported for real.

```
  public Set<RenderingHints.Key> getSupportedHints();
```

API Changes

BEFORE

Example.java

```
  DefaultQuery query = new DefaultQuery(DefaultQuery.ALL);
  FeatureCollection coll = featureSource.getFeatures(query);
}
```

AFTER

Example.java

```
public void exampleMethodFeatureSource source, Filter filter){
    assert(source.getSupportedHints().containsKey(Hints.DECIMATION_DISTANCE);

assert(source.getSupportedHints().containsKey(StoreHints.COORDINATE_SEQUENCE_FACTORY);
    DefaultQuery query = new DefaultQuery(DefaultQuery.ALL);
    Hints hints = new Hints(
       Hints.DECIMATION_DISTANCE, new Double(1500.12),
       Hints.COORDINATE_SEQUENCE_FACTORY,
"org.geotools.geometry.jts.PackedCoordinateSequenceFactory"
    );
    query.setHints( hints );
    FeatureCollection coll = featureSource.getFeatures(query);
}
```

Documentation Changes

Warning
Tp be filled in when a final design is approved

Website:

- Update Upgrade to 2.4 instructions

Developer Guide:

- We may want to provide guidelines for places where the hints are to be declared, for example a well known class or interface for common hints, and the DataStore class for data store specific ones (do you have any suggestion on this one?).

User Guide and User Manual:

- Given that this change just deals with changing some interfaces but does not add real meat for the moment, the user guide should probably not be updated. But we should schedule a task to update it as soon as new "well known" hint gets its way into the code.

Issue Tracker:

- check related issues to see of problems are affected

Tasks

The scope of this proposal is just to have the hints infrastructure ready for 2.4.0 release, so that uses of it can developed freely during the 2.4 time frame without requiring further API breaks.
So the tasks cover just the implementation, fixing the broken interface implementors, and documentation.

	no progress		done		impeded		lack mandate/funds/time		volunteer needed

A target release is also provided for each milestone.

Milestone 1	**2.4-RC0**	
	Andrea Aime	Update Query and FeatureSource/FeatureCollection interfaces
	Andrea Aime	Hunt for broken implementations in both GT2 and GeoServer and fix them
	Jody Garnett/Jesse Eichar?	Hunt for broken implementations in uDig and fix them
	Andrea Aime	Create a test implementation. Maybe have PropertyDataStore use a specific GeometryFactory?
		Update documentation