

Next Generation JDBC DataStore

Motivation:	Improve quality and maintainability of jdbc data stores.
Contact:	Justin Deoliveira
Tracker:	http://jira.codehaus.org/browse/GEOT-2056
Tagline:	Next generation architecture for JDBC datastores.

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

- [Description](#)
- [Status](#)
 - [Supported Checklist](#)
 - [Module Rating](#)
 - [Test Coverage](#)
 - [Tasks](#)
 - [API Changes](#)
 - [BEFORE](#)
 - [AFTER](#)
 - [Documentation Changes](#)
 - [Module Changes](#)

Description

This proposal presents the plan to use a new set of support classes for jdbc based data stores. The motivation/benefits being:

- Reduce the work to implement new datastores
- Cut down code duplication due to copying from PostGIS
- Increase quality among all database backends including security, performance, and testing

In the current jdbc data store architecture a set of base classes (`JDBCDataStore` is subclassed for each implementation. Various methods are overridden to customize for each specific backend database.

In the new architecture, there is a single `JDBCDataStore` class, and in fact it is marked final to prevent subclassing. To customize for each data base implementation the notion of a "dialect" (taken from the hibernate architecture) is introduced.

The dialect interface encapsulates all the operations specific to a particular database implementation.

Status

Elements of this proposal were committed to 2.6.x (and the former implementation should be slated for retirement in 2.7.x or 2.8.x):

- [Andrea Aime](#)
- [Ian Turton](#)
- [Justin Deoliveira](#) +1

- [Christian Mueller](#)
- [Jody Garnett](#) +1
- [Michael Bedward](#) +1
- [Simone Giannecchini](#)
- [Ben Caradoc-Davies](#) +1

Supported Checklist

Module Rating

Using the 5 star module rating system:

- ★ IP Check - Good to go. No missing headers.
- ★ Releasable - No blocking issues in JIRA.
- ★ Used - GeoServer trunk has moved to using jdbc=ng modules
- ★ Optimized - On par with old modules.
- ★ Supported - Well maintained, still lacks some user docs.

Rating = 4/5

Test Coverage

Package /	# Classes	Line Coverage	Branch Coverage
All Packages	31	64% 1998/3121	54% 589/1094
org.geotools.data.h2	6	68% 231/340	59% 46/78
org.geotools.jdbc	29	64% 1767/2781	53% 543/1016

Report generated by [Cobertura](#) 1.9 on 7/2/09 4:24 PM.

Note: This is somewhat inaccurate since it only shows the test results of the non prepared statement path. So in actuality test coverage is higher. But regardless it meets the requirements.

Tasks

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

1. API changed based on BEFORE / AFTER
2. Module restructuring
3. Update wiki (both module matrix and upgrade to to 2.5 pages)
4. Update the user guide
5. Update or provided sample code in demo
6. review user documentation

API Changes

There are no api changes from the client point of view. The dialect api is a private api for developers, not for users.

The regular datastore api remains unchanged.

BEFORE

```
public class OracleDataStore extends
JDBCDataStore {

    ...

    //override getFeatureReader
    public FeatureReader
getFeatureReader(...) {
        ...
    }
}
```

AFTER

```
public final class JDBCDataStore {

    //dialect
    SQLDialect dialect;

    public FeatureReader getFeatureReader( ...
) {
    //boiler plate stuff

    //bulid up sql, delegating to dialect
for specifics

    //run query and return a feature reader
}
}
```

Documentation Changes

Documentation for JDBC datstores will need to be updated for the new modules. Basically everything under:

<http://docs.codehaus.org/display/GEOTDOC/11+JDBC>

Module Changes

Currently all the new code lives in the unsupported module. And all the old code remains in library and plugin:

```
library/
```

```
  jdbc/
```

```
plugin/
```

```
  postgres/
```

```
  db2/
```

```
unsupported/
```

```
  jdbc-ng/
```

```
    jdbc-core
```

```
    jdbc-h2/
```

```
    jdbc-oracle/
```

```
    jdbc-mysql/
```

```
    jdbc-postgis/
```

The following module structure is proposed:

library/

**jdbc/ (merged with what was
unsupported/jdbc-ng/jdbc-core)**

plugin/

jdbc/

jdbc-db2/

jdbc-h2/

jdbc-postgis/

jdbc-oracle/

unsupported/

postgis/

oracle/