

# DataAccess super class for DataStore

<b>Motivation:</b>	DataStore allows access to SimpleFeature, we need to access Feature as well
<b>Contact:</b>	<a href="#">Gabriel Roldán</a> , <a href="#">Jody Garnett</a>
<b>Tracker:</b>	<a href="http://jira.codehaus.org/browse/GEOT-1701">http://jira.codehaus.org/browse/GEOT-1701</a>
<b>Tagline:</b>	★ introduce <code>org.opengis.geoapi.Feature</code>

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

## Description

This proposal:

- introduces DataAccess as a super class of DataStore
- traditional DataStore methods are maintained; often type narrowing a method in DataAccess
- client code can be written against DataAccess for the general case; DataStore offers more specific that can make use of the SimpleFeature assumption
- this proposal covers a naming convention / design strategy that can be used for GridAccess as well

Additional information:

- [Data Access Design Goals](#)
- [Dry Run at DataAccess Story](#)

## Status

Voting took place at [today's IRC meeting](#) over the approach #1 (Generics + DataStore superclass), see [Dry Run at DataAccess+Story](#) for a summary.

- [Andrea Aime](#) 0
- [Ian Turton](#)
- [Justin Deoliveira](#) +1
- [Jody Garnett](#) +1
- [Martin Desruisseaux](#) +1
- [Simone Giannecchini](#) +1

## Tasks

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

1. ✓ Introduce DataAccess level classes
2. ✓ Allow DataStore level classes to extend; patching up implementations as needed
3. ✓ Update and test GeoServer
4. ✓ Update and test uDig (and axios community edit tools)

5.  Update the user guide

## API Changes

The API changes needed are minimal and respect the current interfaces and behaviour. The general strategy is to pull up the common methods from `DataStore` to a superclass and parametrize as per the `Feature` and `FeatureType` flavor they use.

### BEFORE

```
/** @since 2.0 */
interface DataStore{
    void createSchema(SimpleFeatureType
featureType);
    SimpleFeatureType getSchema(String
typeName) throws IOException;
    FeatureSource getFeatureSource(String
typeName);
    ...
}
```

### AFTER

```

/** @since 2.5 */
interface DataAccess<T extends FeatureType,
F extends Feature>{
    List<Name> getNames();
    void createSchema(T featureType);
    T getSchema(Name name);
    FeatureSource<T,F> getFeatureSource(Name
typeName);
    ....
}
/** @since 2.0 */
interface DataStore extends
DataAccess<SimpleFeatureType,
SimpleFeature>{
    void createSchema(SimpleFeatureType
featureType);
    /** @since 2.0 */
    SimpleFeatureType getSchema(String
typeName) throws IOException;
    /** @since 2.5 */
    SimpleFeatureType getSchema(Name
typeName) throws IOException;

FeatureSource<SimpleFeatureType,SimpleFeatur
e> getFeatureSource(String typeName);
    ...
}

```

**BEFORE**

```
/** @since 2.0 */
public interface FeatureSource {
    FeatureCollection getFeatures(Query
query);
    SimpleFeatureType getSchema();
    DataStore getDataStore();
}
```

## AFTER

```
/** @since 2.0 */
public interface FeatureSource<T extends
FeatureType, F extends Feature> {
    FeatureCollection<T,F> getFeatures(Query
query);
    T getSchema();
    DataAccess<T, F> getDataStore();
}
```

## Documentation Changes

- Developers Guide will need a section on adding a DataAccess api that has the right feel
- [Data Module](#) from the [Module matrix](#) page
- [Upgrade to 2.5](#)
- <http://svn.geotools.org/geotools/trunk/gt/demo/example/>
- [Home](#)
- [Home](#)