Geometryless Configuration

Introduction

Geometryless Data Store allows features to be created from data stores with no special spatial geometry data types. This includes the very common cases of separate latitude/longitude numerical columns and also the ability to serve data objects where no geometry is required.

Features with no geometry properties

This is configured with the following parameters.

<datastore namespace = "biz" enabled = "true" id = "mysql" >
<connectionParams>
<parameter name = "namespace" value = "biz" />
<parameter name = "user" value = "me" />
<parameter name = "passwd" value = "foo" />
<parameter name = "dbtype" value = "**jdbc**" />
<parameter name = "**driver**" value = "com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource" />
<parameter name = "**urlprefix**" value = "jdbc:mysql://localhost:3306/mydb" />
</connectionParams>
</datastore>

note change - _urlprefix_ is now the entire JDBC url except for username password

e.g. for oracle: jdbc:oracle:thin:@localhost:1521:beryl

the only really tricky thing is the *driver* parameter: this must match a class of type:
java.sql.Driver

[to check - will andreas code support DataSource too? which is preferred]
e.g.

- org.postgresql.Driver
- net.sourceforge.**jtds**.jdbc.**Driver**

JTDS URL syntax:
  jdbc:jtds:sqlserver://localhost:1433;INSTANCE=TEST;DatabaseName=Test_DB

this class should be sourced from a JDBC2 driver package installed in server/geoserver/WEB-INF/lib

Mapping numerical x,y, columns to point features

A special subclass of this datastore called LocationsXYDataStore performs the mapping from designated x,y, columns to geometry attributes.

note the **dbtype** parameter is different:

<datastore namespace = "biz" enabled = "true" id = "mysql" >
<connectionParams>
<parameter name = "namespace" value = "biz" />
<parameter name = "user" value = "me" />
<parameter name = "passwd" value = "foo" />

```
<parameter  name = "dbtype" value = "locationsxy" />
<parameter name = "driver" value = "com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource"  />
<parameter name = "urlprefix"  value = "jdbc:mysql://localhost:3306/mydb" />
<parameter name="xcolumn"  value = "a.x" />
<parameter name="ycolumn"  value = "a.y" />
<parameter name="geom_name" value = "location"  />
</connectionParams>
</datastore>
</verbatim>
```

Critical notes:

1. this mapping is global to the datastore, so you may need to register more than one data store against your database if you have different column names or geometry names (anything else will require a morte significant change to the underlying geotools - this is in train but will take some time)
2. the x,y columns must be the *last* columns in the result set (underlying table or bypassSQL statement)
3. if you are using a join in bypassSQL you must take care to make sure column names are not ambiguous

Bounding Box Data Store

This variant allows a bounding box represented in 4 columns to be used as a geometry

```
<datastore namespace = "biz" enabled = "true" id = "bboxtest" >
<connectionParams>
<datastore namespace = "biz" enabled = "true" id = "mysql" >
<connectionParams>
<datastore namespace = "biz" enabled = "true" id = "mysql" >
<connectionParams>
<parameter  name = "namespace" value = "biz" />
<parameter  name = "user" value = "me" />
<parameter  name = "passwd" value = "foo" />
<parameter  name = "dbtype" value = "jdbc" />
<parameter name = "driver" value = "com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource"  />
<parameter name = "urlprefix"  value = "jdbc:mysql://localhost:3306/mydb" />
<parameter name="minxcolumn"  value = "x1" />
<parameter name="minycolumn"  value = "y1" />
<parameter name="maxxcolumn" value = "x2" />
<parameter name="maxycolumn" value = "y2" />
<parameter name="geom_name" value = "bbox" />
</connectionParams>
</datastore>
```

The same caveats regarding columns being the last in the list apply to this store.

Additionally, currently the resulting geometry is always mapped to a Polygon object for output. Somoe more thinking is required to support a Box object due to limitations in the Java Topology Suite