# 2.2.x

GeoTools version 2.2.x is the current stable release of the library.

## Resources

- CatalogAPI — manage your datastores
- Color Brewer — ideas from http://colorbrewer.org/ http://colorbrewer.org/ and producing SLD styles
- Events for Styles — track changes that are made to objects in a style
- FeatureType Survey — take a FeatureType instance and construct a to more meaningful GML schema
- FeatureVisitor — extension for making summary calculations
- Random Data Access — when dealing with large data sets
- Tasks for 2.2.0 release — required before 2.2.0 release
- Upgrade to 2.2

## Downloads

Avaiable 2.2 Downloads:

- 2.2.0
- 2.2.1
- 2.2.2
- 2.2-M0
- 2.2-M1
- 2.2-M2
- 2.2-RC0
- 2.2-RC2
- 2.2-RC3

# GeoTools 2.2 At a Glance

## Who should be using 2.2?

Most developers should be using 2.2.x. The only exception to this are people who either need the functionality of 2.3.x or are making significant changes to Geotools.

- Geoserver 1.4.x is based on 2.2.x
- uDig 1.1 is based on 2.2.x.

If you want to make **significant** changes to Geotools, you are more than welcome to base a your work on 2.2.0 (or on the 2.2.x branch) and later integrate them into the library.

**TIP**: If you are doing any work against in GeoTools 2.2.x, please use **Expression** all the time. Don't access feature attributes directly! If you make use of expression you will have no code breakage when upgrading to 2.3.

This is just good programming practice, but following this guideline will explicitly save you pain when upgrading.

## Where is 2.2.x?

You can access 2.2.x in two ways:

- With SVN access to the latest version (2.2.x branch)
- By downloading a numbered release (ie. 2.2RC1)
- Using a Maven 2 build script (sample in developers guide)

See the developers guide for more information on SVN access, especially this page.
You can download the numbered releases from sourceforge.

## Transitioning from 2.1.x to 2.2.x

**transitioning**
Here are my notes for transitioning geoserver 2.1.x to 2.2.x.
1. Add the referencing.jar, api.jar, and render.jar to your project
2. change all the "StyleFactory.createStyleFactory()" to "StyleFactoryFinder.createStyleFactory()"
Dont forget to add "import org.geotools.styling.StyleFactoryFinder;"
3. change all the "FilterFactory.createFilterFactory();" to "FilterFactoryFinder.createFilterFactory();
Dont forget to add "import org.geotools.filter.FilterFactoryFinder;"

4. NamedLayer.getFeatureTypeConstraint() is now spelt correctly
5. <Attribute>.isGeometry() should be replaced with:
Geometry.class.isAssignableFrom(<Attribute>.getType())
6. Anything that implements FeatureStore will also have to have addFeature(FeatureCollection) added to the method. Its very similiar to addFeatures(FeatureReader) so probably not much work.

### The Good (New Stuff!)

The following are new and shiney for the 2.2.x release:

- FeatureCollection
  - Aggregate Functions: sum, min, max etc.. are now available (and optimized!) on feature collections
  - FeatureVisitor is how we get those aggregate functions optimized into sql
  - Sorting into a FeatureList
  - Consistent use of iterator and close( iterator )
- ColorBrewer: able to generate beautiful styles using academic research from http://colorbrewer.org
- Style
  - has been brought in line with SLD 1.0
  - has events (for those making user interfaces)
- JTS: utility class has moved, the old one is still there and deprecated
- Shapefile: performance improvements, supports FeatureList api directly, woot!
- Oracle: makes use of metadata to quickly report the bounds of a table
- EPSG: hsql and property file implementations available. hsql implemention recommended
- Catalog API: finally some help in managing DataStores! An implementation of resource handles.
- Much More ...

### The Bad (needs improvement)

There are some limitations with 2.2.x:

- Documentation has not yet been updated (please help!)
- We still don't have a seperation between FeatureType and FeatureFactory.
- GridCoverage support is still poor (and GCE seems to be on its way out, try and use the formats directly)
- Tests are incomplete (many test cases are disabled, due to Maven 2 being more strict than Maven 1)

### The Ugly (aka will break your code)

We found a couple of Q&A mistakes from 2.1 that we fixed for 2.2:

- StyleFactory is no longer a class, please use StyleFactoryFinder to aquire a StyleFactory implemenation for use

We are sorry about these mistakes, they will require you to do a search and replace in your code 

### Removal of Deprecated classes

The following classes have been around for one public release cycle (they were deprecated in 2.1.0 and will now be removed:

- FeatureResults
- ...we need a complete list

## Working on 2.2.x

Found a scary bug? Did we mess up a specification? Did our documentation confuse you? Here is how to help make 2.2.x a happy place to work and play....

### What changes should be made to 2.2.x?

Anything that does not change the api. The following are fair game:

- Bug fixes
- minor improvements
- new plugins
- functionality fixes

### What changes are not allowed for 2.2.x?

Do not make any API changes! For example, the 2.2 branch implements GeoAPI 2.0. This dependency should not be changed. Transition to any

other GeoAPI release should be done on the trunk. If you want to make "larger" than minor changes (say we got a specification wrong), we may be able to integrate them in on a numbered 2.2.x release (ie. 2.2.1).

If in doubt, just ask.

## Don't lose your work!

Please make sure that you do the following when changing 2.2.x:

- create a JIRA task (and remember the number)
- mention the JIRA task in your commit message on 2.2.x
- mention the JIRA task in your commit message on 2.3.x

If you commit to 2.2.x, you should try to ensure your changes will be in the next stable version of Geotools. To do this, also move your changes to trunk (2.3).
- --# Tell the geotools developer's list what you are working on (or report in the JIRA bug tracker)
--# Make your changes in 2.2.x
--# Test Test Test and run the automatic test cases
--# Commit your changes to the 2.2.x SVN respository
--# 'svn update' your 2.3 archive
--# Move your changes over to 2.3
--# run the 2.3 automatic test case (hand testing would be great too!)
--# commit your code to the 2.3 (trunk) SVN respository
For more information please see the Developers Guide.

**Automatic Code Formating Tools**
Do NOT run any code formatting tools on 2.2.x or 2.3.x. You'll find that, when you (or others) try to move changes between 2.2.x and 2.3.x, you'll be spending a huge amount of frustrating time rectifying changes.
If we can set it up, we will restore Jalopy code formatting, so if you run the formatter before a commit pain will be minimal.

## Changes to Expected on the 2.2.x branch

Here's some things I've heard developers are talking about doing on the 2.2.x branch:

- fix FeatureType.create (done)
- move to JTS 1.7
- move to JTS 1.7's WKB parser (remove WKB4J)
- renderer improvements - adding highway shields
- renderer improvements - labeling improvements
- renderer improvements - moving SLD Filters to Datastore
- PostGIS datastore improvements
- NEW datastore - PostGIS Versioning datastore
- lots of bug fixes

All of these changes should not affect anyone else working on the 2.2.x branch.