

The Geotools Users' Manual, version 2.3

1. [Welcome!](#)
2. [Quick Start](#)
3. [Conceptual Foundations](#)
4. [Geospatial Science](#)
5. [SDK Configuration](#)
6. [The Geometry Model](#)
7. [The Geo-Referencing Model](#)
8. [The Feature Model](#)
9. [The Query System \(Part I\)](#)
10. [The Data Access and Storage Model](#)
11. [The Query System \(Part II\)](#)
12. [The Display and Rendering System](#)
13. [Network Graphs for Advanced Analysis](#)

Welcome!

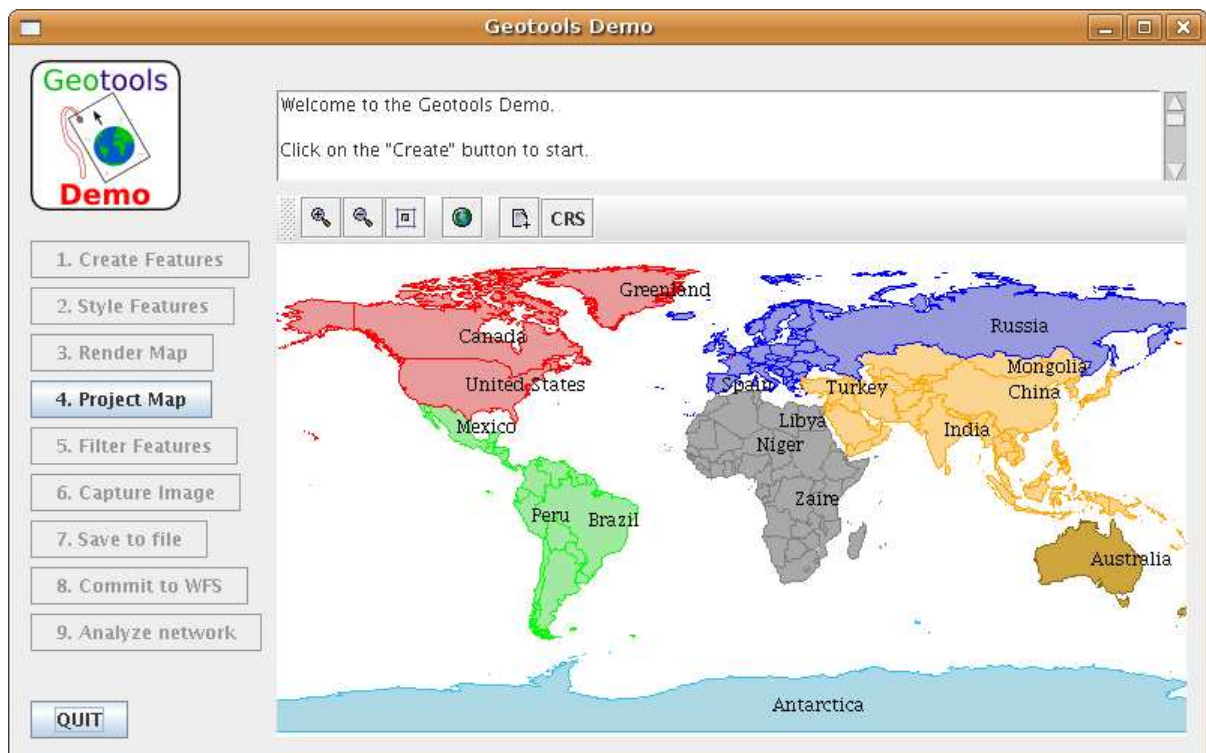
Geotools is a code library for Geospatial applications. The library is intended to provide foundational object classes for creating geographic information systems. The library is written in the Java computer language, currently using version 1.4. For more details, visit the Geotools project [webpage](#).

This manual intends to help people develop software programs which use part or all of the Geotools library to create their own geospatial applications. These programmers are considered *users* of the Geotools library. This manual is intended to help programmers learn how to use both the GeoTools library itself and closely related libraries such as the GeoAPI and JTS libraries.

Geotools can be used to build applications of many different types including network server applications and desktop clients. The library consists of various modules which can be combined according to programmer needs. An example client application is included as the Geotools demo shown in [Figure 1-1](#).

Figure 1-1 The Geotools Demo showing an unprojected map of the world.

◀ [About This Document](#)



The Geotools demonstration application uses swing components, such as a label, buttons, and a text area, but also uses the JMapPane component to display geospatial data and provide simple tools to manipulate those data.

This manual will get users started using the geotools codebase. We hope the manual can get users to the point that they can continue on their own by reading the GeoAPI and Geotools javadocs along with the source code. Advanced questions should be addressed to the mailing list or raised in the Internet Relay Chat (IRC) room.

This manual is not a guide to developing and improving the library itself; for that see the [developers page](#) of the web site and the Developers' Guide available from that web page.

Geotools is explicitly *not* an extensible application. Geotools provides almost no graphical user interface functionality. Several projects may yield more user interface elements, notably the [GeoWidgets project](#) which will hopefully provide widgets based on the [GeoAPI](#) interfaces.

There are two notable projects which leverage Geotools to provide higher level functionality.

The [Geoserver](#) project is a network server to provide maps using the Web Mapping Standard (WMS) or to provide individual features using the Web Feature Standard (WFS) and its transactional extension, the Web Feature Standard for Transactions (WFS-T).

The [User-friendly Desktop Internet GIS \(uDig\)](#) project provides a powerful platform for the creation of user, desktop-based applications drawing on the [Eclipse](#) project and its [Rich Client Platform](#) sub-project.

There are also many other software projects for spatial data which may be easier to use for simple projects than the Geotools library. For example, the [JUMP](#) project and its [OpenJUMP](#) fork are reputedly easy to use. The

[FreeGIS](#) project attempts to maintain a list of the free software projects related to geospatial data. Those projects, or others like them, may be a better place to start than working directly with the library itself.

Quick Start

This chapter describes, first, how to obtain the library and the tools required to be able to use the Geotools library and, second, how to use these libraries to display a map on screen with an exceedingly primitive application.

The chapter assumes that the programmer understands the following:

- The Java (tm) Programming Language
- The basics of Geographic Information Systems

This chapter tries not to assume any other specialized knowledge; the examples therefore start from the basic desktop of modern GUI operating systems either of the UNIX heritage, namely GNU based or MacOS X, or from the CPM heritage, namely Windows (tm). Since each programmer uses her own tools to actually develop her programs, each will have to adapt these examples to fit her own needs.

2.1. Get Geotools

Geotools is distributed in three separate pieces: as a binary distribution, as a source distribution, and as a javadoc collection. Each of these can be obtained from the project website.

2.2. Configure Java

Geotools requires a Java VM and two imaging libraries, the Java Advanced Imaging library and the Java Image I/O library. These must all be configured correctly for the examples which follow.

2.3. Run the Demo

Geotools contains a handy demonstration application which is useful both to ensure that the libraries are properly installed and to get programmers started exploring each of the different areas of the library.

Once the Java libraries have been configured, a simple demonstration program can be run with the following command:

```
java -jar /path/to/gt2-demo-introduction-VERSION_NUMBER.jar
org.geotools.demo.introduction.QuickStartGUI
```

2.4. Read the Demo

The source code for the Demo application provides a quick introduction to each of the major modules of the code library. The source walks through the process of creating Features, both programmatically and from various sources such as shapefile and web servers. The source then discusses how to create styles and display a map. The source continues by discussing advanced features of the Geotools library. The source is contained in the `org.geotools.demo.introduction.QuickStartGUI.java` source file within the `gt2-demo-introduction-VERSION.jar` JAR file (where VERSION is replaced by the current version number).

Conceptual Foundations

This chapter introduces the Geotools library from a conceptual point of view. The chapter starts by introducing the components of the library and showing how these components could fit into a Model/View/Controller architecture of an actual application. The chapter then details the different sub-components of the Geotools library. Each of these sub-components will be discussed in a tutorial example in the next section as these tutorials become available.

3.1. Terminology and Standards

Geotools uses certain terms in very specific ways. The terms used to refer to geospatial information come from a set of standards developed by the International Organization for Standardization (ISO) and the Open Geospatial Consortium (OGC). These standards, for example, use the term 'Feature' to describe the fundamental unit of geospatial information: both real physical entities such as rivers and abstract ideas such as property boundaries.

The fundamental standards used by Geotools are:

Table 3-1 Geospatial Standards

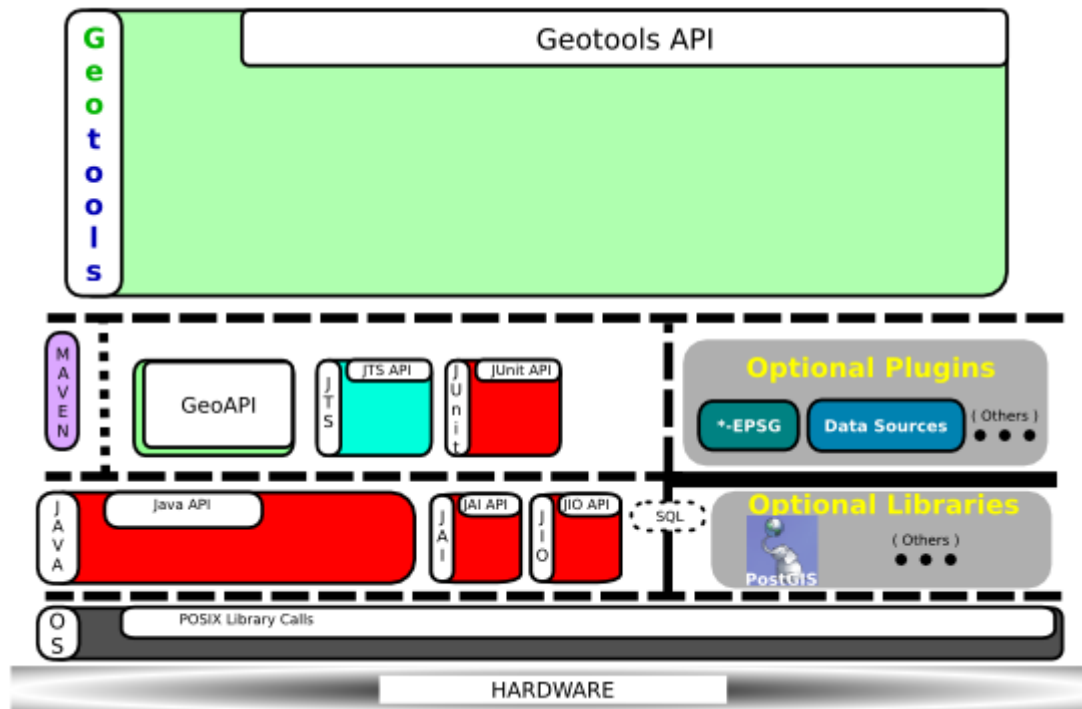
Specification	Title	Base package
ISO 19103	Geographic information - Conceptual schema language	org.geotools.util
ISO 19107	Feature Geometry (Topic 1)	org.geotools.geometry
ISO 19111	Spatial Referencing by Coordinates (Topic 2)	org.geotools.referencing
ISO 19115	Metadata (Topic 11)	org.geotools.metadata

The best introduction to this terminology is the general introduction written by the OGC.

3.2. Runtime Environment

The environment on which the Geotools library runs is shown in [Figure 3-1](#).

Figure 3-1 The Geotools Environment.



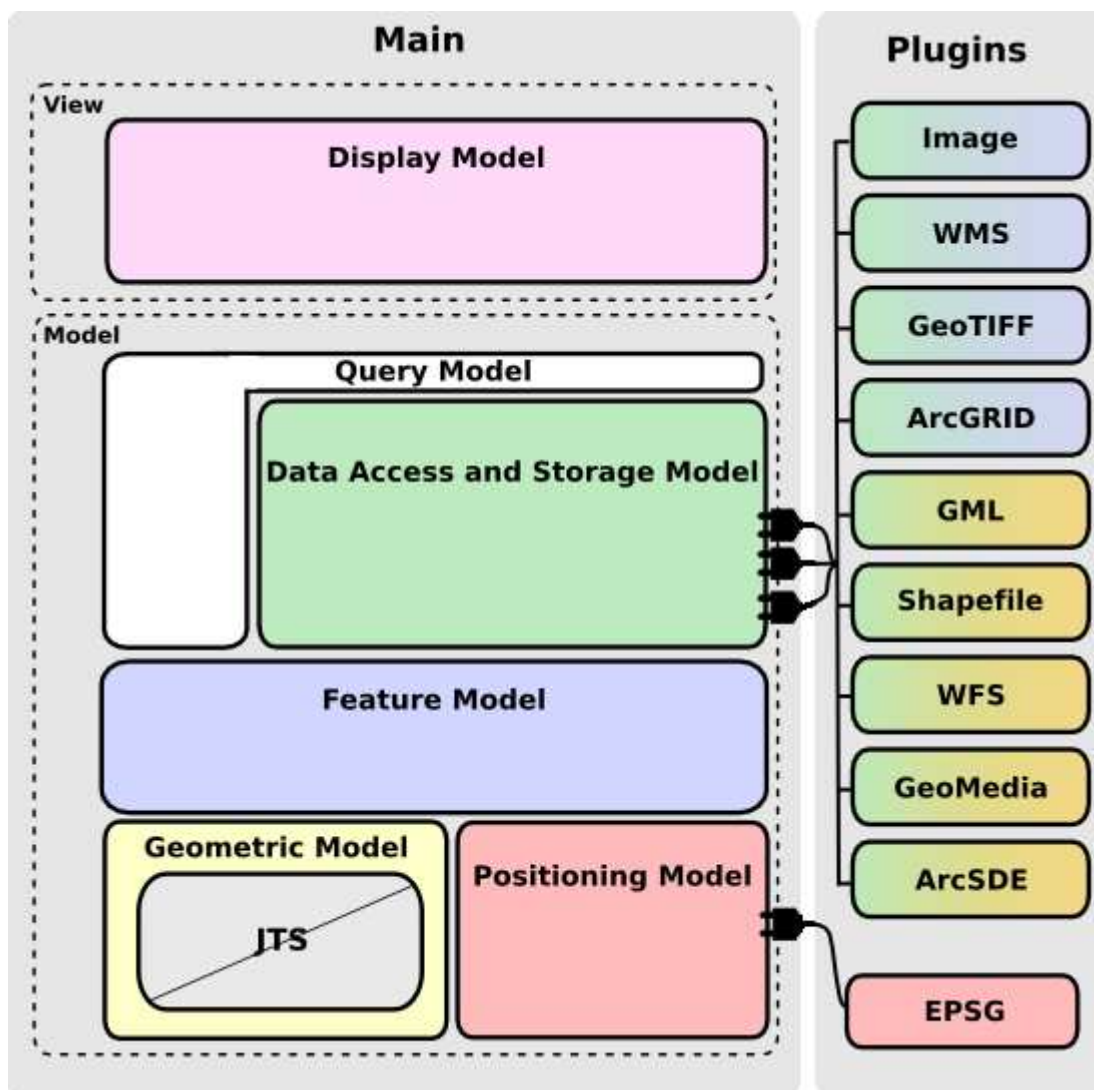
Geotools is written in Java and therefore runs inside a Java Virtual machine (VM). The Java VM generally runs through the POSIX layer on the hardware. The library requires two of the standard Java extension libraries, Java Advanced Imaging and Java Image I/O. The Geotools uses the GeoAPI project to define a standard API, the JTS project for robust geometric convolutions, and the unit definitions provided by JUnit. There are also several external libraries and code plugins which can be connected to Geotools.

3.3. Library Architecture

The Geotools library is split into several, essentially separate components. This component architecture makes it easy to use only part of the library, provides a clean separation to extend Geotools with new functionality, and provides a simple way to replace the default implementation of any particular component with another implementation.

The architecture of the Geotools library runs is sketched in [Figure 3-2](#).

Figure 3-2 The General Geotools Architecture.



The Geotools architecture is comprised of several data models and a display system.

Geotools is comprised of several models. The main library consists of the geometric model, the geopositioning model, and the feature model, which jointly provide the core data model for geospatial entities. The positioning model uses one or more authorities, most notably the EPSG database. Geotools also uses a Data access and storage model, with plugins for the common data formats. A Query model provides a rich facility for data filtration and the creation of data dependent

The Geotools library is split between the Main module and the Plugin modules. The Main module provides the core libraries and the plugins provide specific functionality. There are several plugins that provide access to different format versions of the European Petroleum Survey Group (EPSG) data base of known coordinate geo-referencing systems. Similarly, there are multiple plugins which provide access to common geospatial data formats such as imagery and vector formats either on the local disk, in databases, or on the internet.

The Main module itself is split into two components, the data model component and the data view component. These two components are two of the three components in a standard Model-View-Controller split used for most modern GUI application. The Model-View-Controller architecture is a standard architectural design pattern designed to cleanly separate the data model from the logic used to create and modify that model. In general, this architecture does not split the view and controller components in quite as distinctive a manner. However, the Geotools library does not provide any controller logic

so the library is cleanly split into a Model-View division.

3.3.1. The Data Component

The Geotools library follows the standard split between the model, view, and controller components. The data component consists of the entities which describe the real world entities within the computational system.

3.3.1.1. The Geometric Model

This model provides the infrastructure for defining vector based geometric figures. The model is based on the idea of coordinates and of aggregates of these coordinates into points, lines, polygons, multipoints, multilines, or multipolygons. The geometric model also includes operators to perform standard computational geometry operations on the aggregates. In versions up to Geotools 2.2, the Geotools library depends on the Java Topology Suite (JTS) library and is therefore limited to working in two dimensions.

3.3.1.2. The Geopositioning Model

This model provides the infrastructure for geopositioning and for common operations on geo-referenced data such as transformations and re-projections. This model provides ways to define Coordinate Reference Systems (CRS) and the EPSG plugins provide a standard source for the commonly defined geodetic datums and CRS. The model further provide access to fundamental units and parameter structures.

3.3.1.3. The Feature Model

Features are the fundamental unit of geospatial data in the terminology used by Geotools. Features may describe a real world entities or theoretical constructs. Conceptually, Features contain an identifier, a bunch of attributes in a java array, and a schema defining the attributes and the layout of these attributes within the attribute array. The attributes will include the geospatial definition of the feature, any temporal definition of the entity, and any other attributes of the feature. The standard method to obtain feature contents is through the creation of 'queries', combinations of filters and expressions; these are described below.

3.3.1.4. The Data Access and Storage Model

This model provides methods for creating, manipulating, and storing data. The model provides an approach to storing data from different sources in a single catalog structure. The model facilitates the interaction with data in vector or raster formats contained on the local file system, in a database or from a network server. The plugin module provides a number of plugins to access common data formats such as GML files, Shapefile format files, GeoTiff images, Geospatial databases, Web Mapping Servers and Web Feature Servers.

3.3.1.5. The Query Model

This model provides a standard way to discover and obtain the contents of geospatial data from the data sources and from known features. The query model involves the creation of Filters to sub-select the data contents and expressions to extract only the relevant attributes from the features of interest.

3.3.2. The View Component

3.3.2.1. The Display Model

The display model provides a standard approach to the creation of visuals through which

to present the contents of a group of features. This model provides a standard approach to styling feature contents according the complex rules which are necessary to create visually appealing maps. The model also provides a rendering infrastructure for the creation of images.

3.3.3. No Controller component

There is *no* controller component in the Geotools library; the component is explicitly left to be implemented by library users.

3.3.4. Plugins

Geotools uses several plugins to provide geopositioning information or to provide access to specific data formats.

Table 3-2 Plugins

Name	Description
arcgrid	A plugin to support the use of files in the ArcGRID raster format.
arcsde	A plugin to support access to the ESRI ArcSDE database. This may require a download of a proprietary library through which to access the database. [TODO: Check]
db2	A plugin to support access to the DB2 database.
dir_ds	TODO
epsg-access	A plugin with a version of the EPSG database of coordinate reference systems and datums in the Microsoft Access (tm) database format. This is the only authoritative version.
epsg-hsql	
epsg-postgresql	A plugin with a version of the EPSG database of coordinate reference systems and datums in the in-memory HSQL database format. This is an alternative, non-authoritative version.
epsg-wkt	A plugin with a version of the EPSG database of coordinate reference systems and datums in a set of Well-Known Text (WKT) file format. This version is almost always functional, but its content is significantly different from the content of the Access database. This is an alternative, non-authoritative version.
geomedia	TODO
geotiff	A plugin which supports access to files in the GeoTIFF image format.
gml	A plugin to enable the reading and writing of files in the Geographic Markup Language (GML) extension to the eXtensible Markup Language (XML) format.
image	TODO
mif	A plugin to enable access to the MIF file format.
mysql	A plugin to enable access to the MYSQL database.
oracle-spatial	A plugin to enable access to the MYSQL database. This may require a download of a proprietary library through which to access the database. [TODO: Check]

Name	Description
postgis	A plugin to enable access to a PostGIS database, a spatial extension to the PostgreSQL database.
shapefile	A plugin to enable the use of files in the Shapefile format.
tiger	A plugin to enable the use of files in the TIGER format used by the Census Bureau of the United States Government.
vpf	A plugin to enable the use of files in the Vector Product Format (VPF) which has become the standard format for the distribution of geo-spatial data by the United States Geological Survey (USGS).
wfs	A plugin to support the use, over a network, of a Web Feature Server (WFS) host.
wms	A plugin to support obtaining raster images, over a network, from a Web Mapping Server (WMS).

3.4. Code Layout

The Geotools library consists of several modules which can be assembled according to user needs. For example, a simple application could use only the api, geometric, referencing and feature modules. Each module contributes several java packages to the overall library. A few java packages are special and are built piecewise from code in several modules.

3.4.1. The Modules

3.4.2. The Packages

Should this be combined with the section above?

3.5. GeoAPI

The GeoAPI project defines an Application Programming Interface (API) consisting of java interfaces which outline the conceptual model developed by the ISO and OGC standards. Geotools intends eventually to be simply one implementation backing the GeoAPI interfaces. It will hopefully eventually be possible to use Geotools simply by obtaining a single reference to a Geotools object and then develop all code against the GeoAPI interfaces.

Neither Geotools nor GeoAPI have matured sufficiently for the ideal use pattern to be effective. Currently, the user need to mix methods from the GeoAPI project, from the Geotools internal API, and from actual Geotools classes themselves.

The GeoAPI project grew of the needs of Geotools and several other java language geospatial projects which wanted to implement standards conformant code.

3.5.1. The GeoAPI Design

3.6. The Factory System

Geotools uses a factory system extensively to enable users to create objects while

developing most of their code against the standard APIs.

3.6.1. The Geo-Referencing Factory System

The Geo-Referencing module has developed its own factory system.

Geospatial Science

This chapter provides a brief introduction to geospatial science including the core ideas needed to use the Geotools effectively.

4.1. Resources

There are many resources from which to learn geospatial science, including books and web sites.

4.1.1. Books

Geographic Information Systems and Science by Paul A. Longley, Michael F. Goodchild, David J. Maguire, and David W. Rhind published by John Wiley & Sons, Chichester, UK 2001.

4.2. Geodesy for GIS

Geodesy is the study of the shape of the earth. Its core contribution to Geographic Information Systems relates to the definition of georeferenced coordinate locations.

SDK Configuration

This chapter describes the configuration of a more complete development environment for Geotools than was provided in the quick introduction in [Chapter 2 — Quick Start](#). This chapter assumes that the user has setup a Java environment according to the instructions provided in [Section 2.2 — Configure Java](#). The chapter describes the configuration of a Geotools source distribution, the use of the Maven build tool and the

5.1. The Geotools SDK

The full Geotools distribution consists of three components: a binary archive, the javadoc archive, and the source archive. The latter archive provides all the elements required to build the first two.

To be written...

5.2. Maven setup

Maven is a java language tool designed to help build complex java programs.

To be written...

5.3. Command-line development

To be written...

5.4. Eclipse IDE development

To be written...

5.5. Netbeans IDE development

To be written...

The Geometry Model

6.1.

The Geo-Referencing Model

7.1.

The Feature Model

8.1.

The Query System (Part I)

9.1.

The Data Access and Storage Model

10.1.

The Query System (Part II)

11.1.

The Display and Rendering System

12.1.

Network Graphs for Advanced Analysis

13.1.