Fix the Docs

This page documents steps taken to improve our GeoTools documentation for **users** (aka those programmers who use geotools as a library). This page was started in April 2006 around the 2.2.RC2 version of Geotools.

The following work has been started:

- User Manual generated from docbook and stored in our svn repository
- User Guide simple reference pages gathering up code examples from geotools-user email list

The proposal is presented at the top. Below is the stream of discussion during development of the proposal.

- Home

The Proposal

A "Programmer's Manual" will be created, similar to the "Developer's Manual," but focused on users of the Geotools library rather than those improving the library itself.

Outline:

1. Welcome
2. Getting Started (From nothing to a map display in three, hot beverages)
    a. Starting from a command line with Maven
    b. Starting with the eclipse IDE
3. Geotools Concepts
    a. The conceptual model
    b. Terminology and links to standards
4. Tutorials: (Each is compact, a page of text and a crude code tutorial---a single main() method)
    a. Geometry Tutorial (using JTS)
    b. Geometric Transformation Tutorial (Projection / CRS)
    c. Feature Tutorial
    d. ... more as the API stabilizes / we have time.
5. User's Reference
    a. The API's: GeoAPI, Geotools
        i. GeoAPI Javadoc
        ii. Geotools Javadoc
        iii. Javadocs for projects on which Geotools depends(and how to find the right version)
    b. Plug-in Reference - design and interfaces
    c. Data storage Reference
        i. Shapefile - connection parameters, tips on use
        ii. Postgis -
        iii. ...
    d. Glossary
6. Geographic Information Systems and Science
    a. What is GIS?
    b. ...
    c. Sources for more information
7. Migration Guide to Geotools 2.3 from 2.0 and 2.1 (If anyone has the heart).

Focus:

The idea of this documentation project is to make a minimum amount of work for maintenance while giving users a solid start into the codebase.

This proposal does demand of the project that it make a commitment to keeping the tutorials in functioning order by updating the tutorials when the Geotools library changes.

Discussion

Okay so this page gathers together some ideas towards fixing the docs, it will not cover why they are broken. This is part of the geotools "growing up" process, and something we want to nail before graduation from the OSGEO inccubation process.

I would just like to thank the community for the strong show of support for this topic, we all know it is important, and this page outlines what is going to be done.

What is off the Table

I also must appologize for my focus here, I am **not** looking for *more*, I am looking for *less*. The goal here is to identify what documentation we can maintain and keep up to date as a community, we can add additional things later.

Technology

Yes our toys are fun, once they use it they will know.

Names

They will see us on the email list, the wiki signs our work, svn blame knows all and we thank our sponsors on seperate page.

ISO

The user cannot be asked to keep TC211, 19119, 19115 and so on straight, just don't go there girl friend!

Tip: use the language as if the user knew what you were talking about. Say Feature and Geometry and Metadata.

Background Knolwedge

Any thing covered by a specification or a book, we are not an intro. (I feel bad about this but we need to get our own house in order before we can try and help people learn GIS, I know you are all excited about the field, and many of you teach, but this has got to be a firm requirement).

We are here to document a library, not explain what is going on. Document as if the reader
understands GML3.

Computer Science

Patterns are fun, tradeoffs are fun. Users do not care. I will need to restrain my self.

We are not here to explain why transacction is such a cool use of patterns, or why the graph api is so good. It is obvious that WMS uses the stratagy pattern when you read the class names, the reader can understand from WMS1_1_0Stratagy class what forces were balanced (and why),

Justification

No time spent on justification, much like the decision to leave ISO 19119 naming this is better handled in javadocs. We are focused on use, not defence. This is code, if it runs who cares.

UML Diagrams

No Omondo, No Viso nothing we cannot keep up to date. I know it is next to impossible to explain the big picture without a picture. Talk to Andrea about getting diagrams as part of our javadocs.

The Big Picture

I am not intereted in explaining the big picture right now, we can write some articles later. The small picture will do as long as we explain enough small pictures. We can organize our docs bottom up however so that by the time we are looking at rendering the SLD will not scare them.

What is on the Table

So after all that what is left? Here is a set of guidelines that have produced usable docs, and have been explicit enought to make writing easy.

Getting Started

Yes it looks like we could write a tutorial here, and you can (just not now).

> This section typically contains broad overview information. The Workbench User Guide provides two tutorials in this section, namely, Basic Tutorial and Team Tutorial. They mainly present high-level information to introduce the information in the rest of the guide.

- Quickstart with Maven 2 Project
- Quickstart with Eclipse

Concepts

Just the facts, these serve to keep us for explaining what a Feature is badly on many different pages. This gives us a single page to explain what a Feature is badly This is also where we would link to external resources like OGC and ISO specifications.

> Conceptual topics present an in-depth discussion of ideas, architecture, technologies, solutions, or processes. Concepts are high-level descriptions of the schema and functions of the plug-in tool. This section helps provide a general understanding of how the plug-in functions. For example, in the Workbench User Guide, the Concepts section includes a discussion of what perspectives and views are and how they relate to one another.
>
> Additionally, conceptual topics should not contain procedures, which are documented in task topics (Tasks section). Therefore, use conceptual topics to provide a comprehensive discussion of the processes or options behind a procedure, and include links to related task topics._

2.3: No specific goals, just write a page here when you are tempted to explain something
2.next: it would be nice to walk through each specification and make sure we have it covered.

Tasks

Numbered list indicating how a given task should be performed in prose.

> Task topics tell users how to do something. Users who access task topics typically are focused on the job at hand, so provide them with a procedural presentation of the information they require. Task descriptions are step-by-step instructions for performing specific actions and tasks in the platform. For example, the Tasks section contains step-by-step instructions for creating a repository location, and for importing a file from the file system into the workbench.
>
> Task topics are comprised of the following sections:
>
> - A brief introductory section, which helps orient the user to the task.
> - A procedure, which is a series of steps that describes how to accomplish a specific goal.
>
> Use links to direct the user to related information; task topics do not provide conceptual information or present reference material.

Include a code example at the end. The
code example must be upto date, even if all it ends up being is a link to a file in the demo directory. If we can inline the code example from svn we will do it.

2.3:

- We should target the top two uses for each "package"
- so Construction and Reprojection for referencing.

2.next:

- inline the svn example
- aim to document the 80% benifit, or just to cut down on email

Reference

This is mostly covered by javadocs - go team! This is where many of the articles will go to die, anything worthwhile should be placed in the codebase in doc-files directories.

> General reference topics document non-conceptual or non-procedural stand-alone information. Examples of general reference topics can include descriptions of various workbench resources such as wizards, dialogs, views and perspectives, as well as, a list of keyboard shortcuts, a glossary, or JavaDocs information. Reference materials are helpful resources that will assist the user while using the platform. Additionally, you can use this section to site relevant external reference material. Information in general reference topics is often used to supplement other conceptual, task, and reference topics.

And Now for the Scope

I am tagging in Adrian here as a champion of the user, his dedication to larning geotools has been great. And I hope that all his scars have not yet healed, so he can help us remember what there is to trip over.

Now we did spend some time offline on scope, and we are finding some common ground.

- Working on real code examples (small apps not something like styped map pain)

Common Ground

If we are going to do this well, we must ensure that we have a minimum

coverage of all the basic topics: Data Model, Rendering Model, (not Control since that's implementation dependent).

Snippets are fine later in the process but are not good intros because they don't give the import statements (which are critical to finding the javadocs/code).

I think we should expect to have a working human readable example+test for each of the datastores.

+ Cover the entire library
+ Have code examples (that run) rather then snippets
- I am scared of code example for each datastore, I would rather have developers use the API

Setup

Much like for uDig, I think you all should put together a full, heavy-weight SDK package which includes all the binary, source and documentation data. The idea is to show people how to read the code and javadocs **as much as** how to code because we are trying to get them to become self-sufficient as quickly as possible. This means that the JTS jar, for example, must include source and ideally the eclipse metadata so we can skip into that code when we hit it. I would do this myself but I have **no** clue how to do this. Actually, I would be interested to know.

As a corrollary, I would like our examples to end with 'further code reading' pointing to sections of the code that do similar things. This would help introduce people to the geotools packages as well.

+ Getting Started: Starting a Maven 2 Project
+ Getting Started: Starting a Eclipse Project (using a binary and source download)

Topics

Referencing (Transformation)

We can start this now (since referencing is stable for 2.3.x). We can use this as an example for review during Monday's meeting.

Transformation is at the core of the geo aspect of the library. I suspect we need to start with an explanation of CRS at least so far as mentioning the different kinds of CRS's and the EPSG authority.

This page is great:
http://svn.geotools.org/geotools/trunk/gt/demo/referencing/src/org/geotools/demo/referencing/CTSTutorial.java
but not so much as a snippet.

This would result in a simple projection converter:
geoProjectionConvert --input simple.shp --output WGS84 reproject.shp

again working as an example.

- We will not have introduced Shapefiles yet, so the converter is not going to work as an example

Features (actually everything up to Data)

We still got to work through this one, Jody wanted this:
+ Introduce FeatureType (aka the Feature Model)
+ Introduce Feature (aka the Data model)
+ Introduce Expression (data acess)
+ Introduce Filter (query language)
+ Introduce Data Reading
+ Introduce Data Writing

And Adrian wants to go the other way:

Building a Feature was critical for me since that was an intro to the Data Model. Actually, I would like a three part:

1. Make a set of features from a simple shapefile
    - demoRead simple.shp
2. Access all the components of the features: geometries, CRS, attrib.
    - demoExpression
3. Build a correct (i.e. with CRS) feature.
    - demoCreate
4. Write a Feature out to GML
    - demoShpToGml simple.shp simple.gml

Each of these would be a seperate application, we had thought of a simple file converter bringing them together (but it made Jody nervous).

- geoFileConvert --input simple.shp --output simple.gml

 on complete examples programs, if possible I would like to use the simple.properties to allow the examples to remain independent of shapefile for as long as possible.

Rendering

This should obviously wait until they have refactored the renderer. If we need to write the MapPane then let's do it---you have seen how painful the current situation is to new users. (let's delete the others) and leave links on the wiki to revisions where they can be found.

 Removing the others (they are already "gone" due to legacy dependency)

Similarly, I would like a

- naiveBrokenViewShp simple.shp
  which would have as a GUI, only a MapPane and an AWT "dismiss" button.

 Greate we can use this to make earlier examples visible,
and then explain how it was done here.

Advanced Topics

These can be more freeform, more compact (snippets) more vague. By the time a user gets to these, she has a solid background in navigating and using the library so we can be more terse.

 Lets just not do this, remember the 80% focus