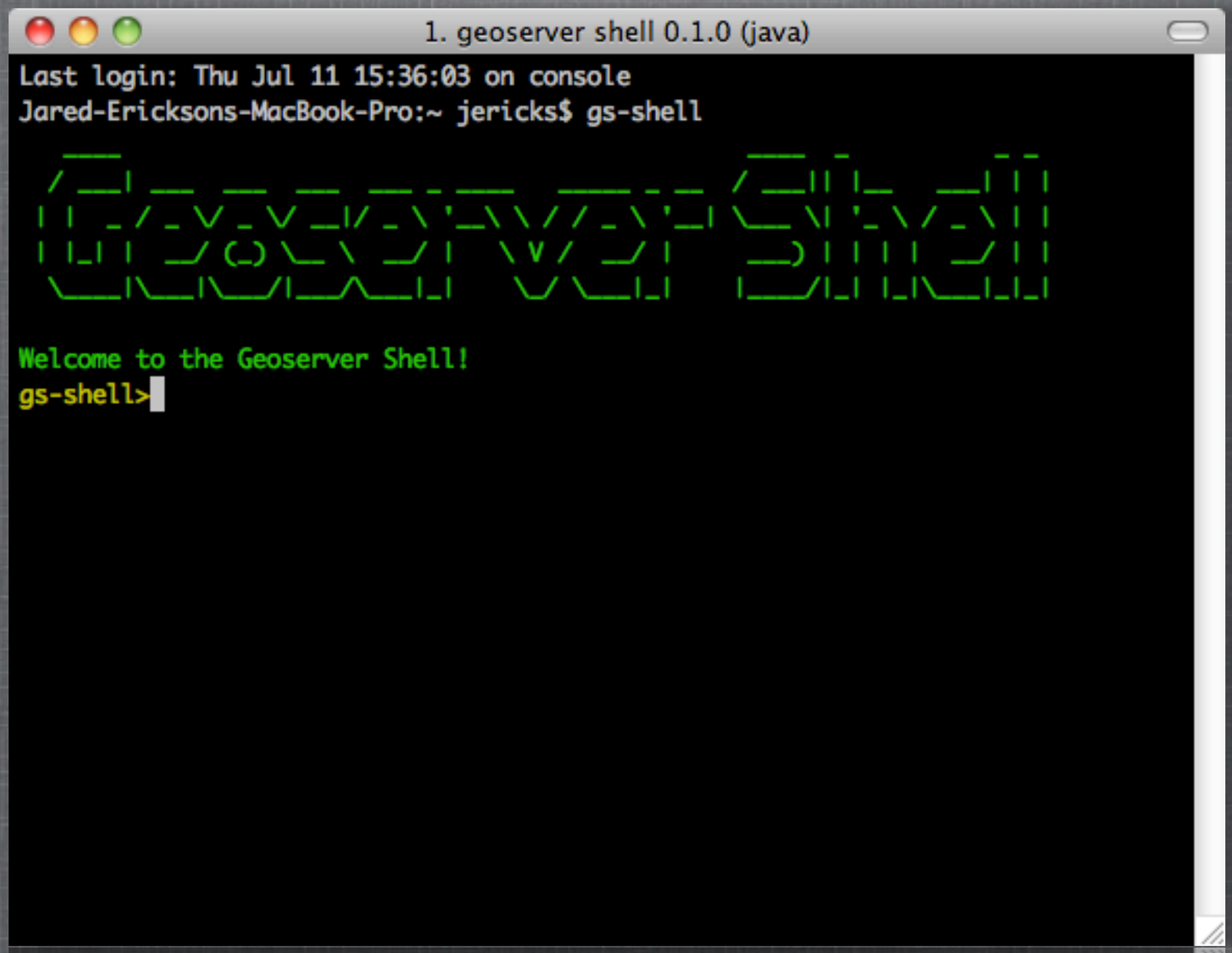
GEOSERVER SHELL

Administer Geoserver with a Command Line Interface (CLI)



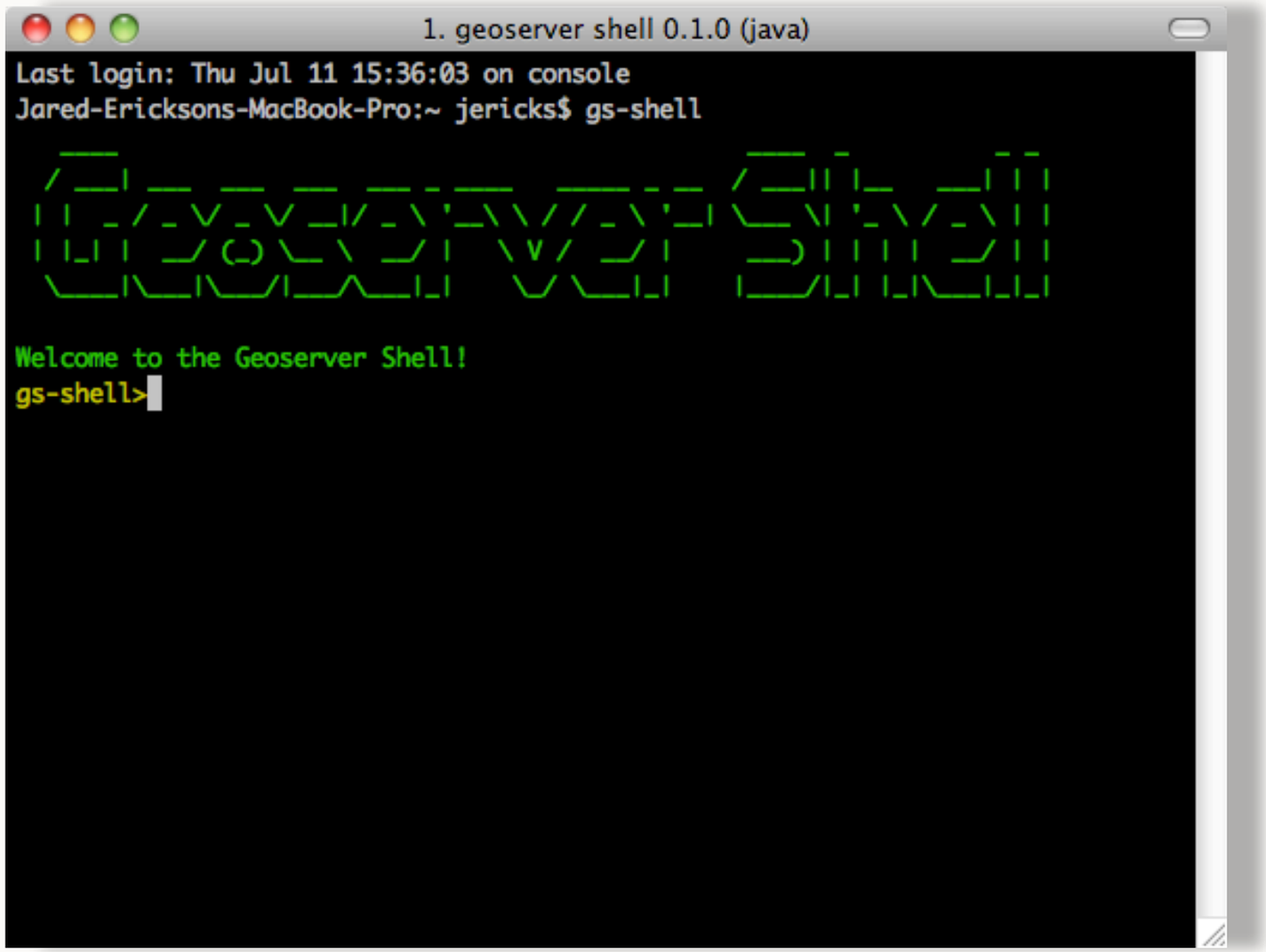
CUGOS UW Fall Fling 2013

OUTLINE

- What is Geoserver Shell?
- What is Geoserver?
- Why a command line interface?
- What does Geoserver shell do?
- How does it work?

GEOSERVER SHELL

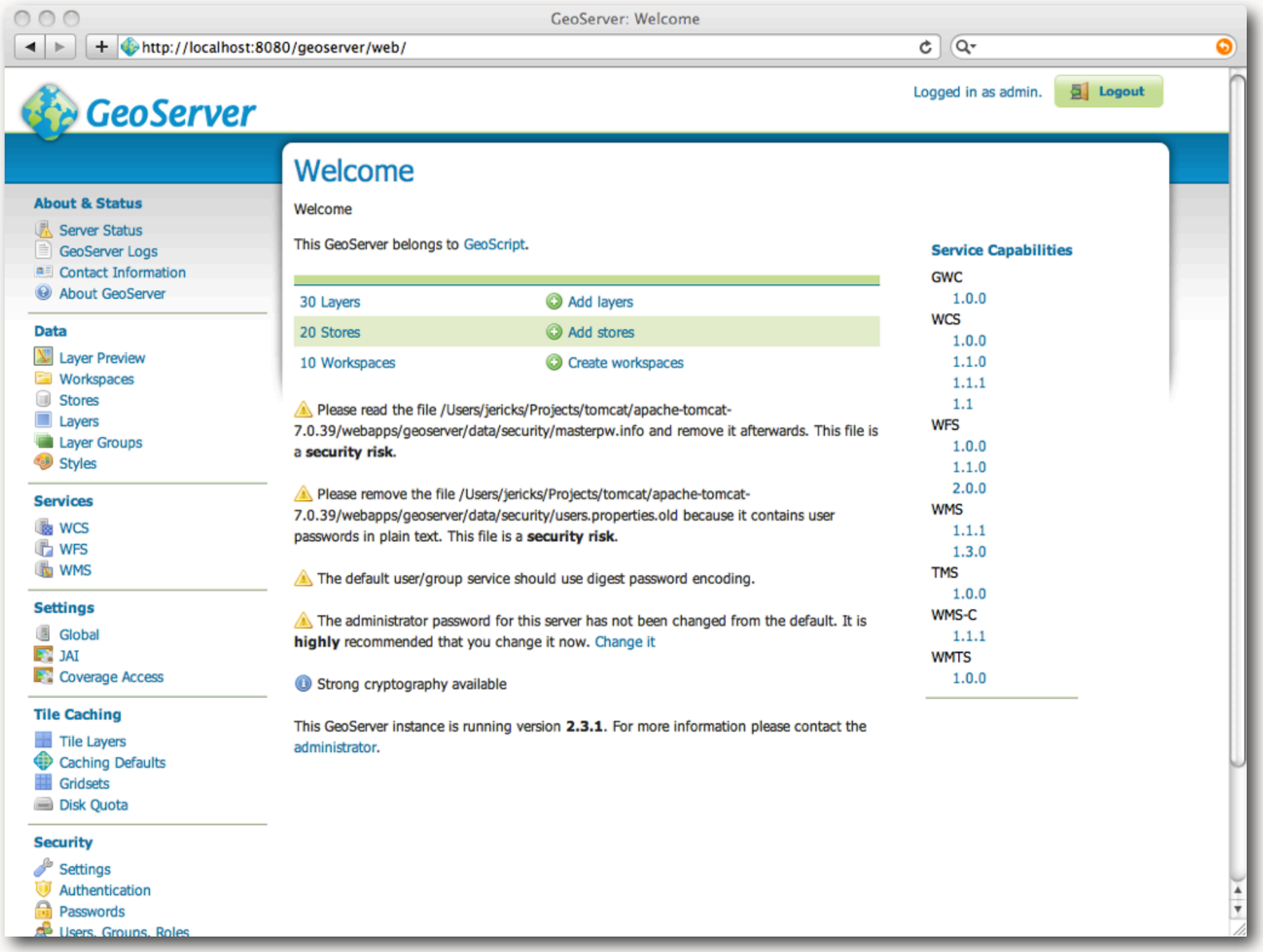
- Command line interface for administering Geoserver



GEOSERVER

- Publish geospatial data
  - Maps and Features
- Focused on implementing standards
  - WMS - Web Mapping Service
  - WFS - Web Feature Service
  - WCS - Web Coverage Service
  - WPS - Web Processing Service

GEOSERVER GUI



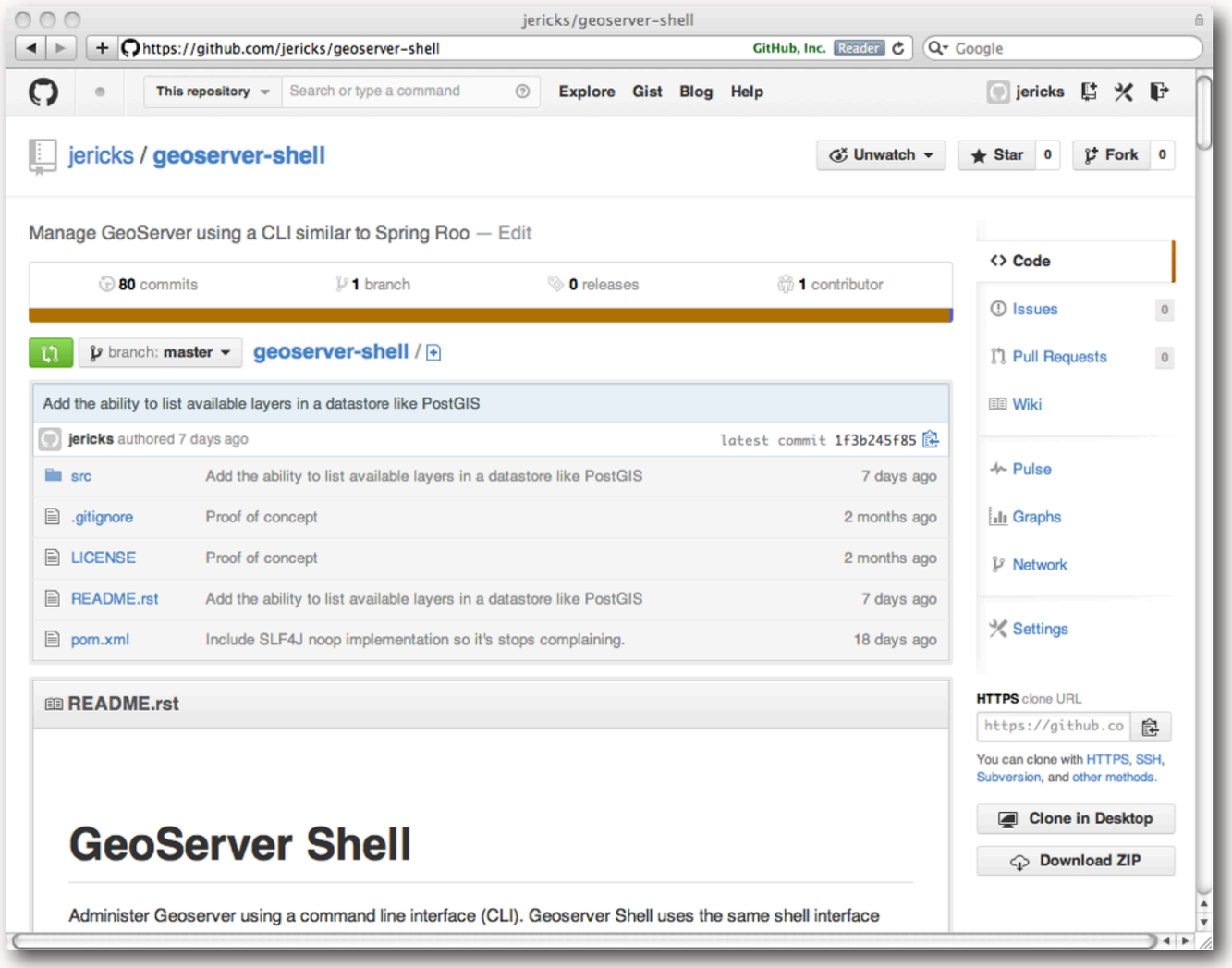
WHY COMMAND LINE?

- GUI are good for occasionally adding or updating 1-10 layers
- Command line interface is good for 10-100 layers
- Command line interface is **repeatable** and **scriptable**

WHY COMMAND LINE?

- Why Use Make? by Mike Bostock: http://bost.ocks.org/mike/make/
- Had to repeat a geospatial analysis
- “how much easier it could have been had I simply recorded the process the first time”
- “Makefiles are machine readable documentation that make your workflow reproducible”

GEOSERVER SHELL



https://github.com/jericks/geoserver-shell

INSTALL

- Download 0.1 release
- https://github.com/jericks/geoserver-shell/releases
- Put bin directory on path
- run **gs-shell** from the command line

COMMANDS

geoserver	datastore	layers
workspace	shapefile	ows
version	postgis	settings
manifest	featuretype	gwc
namespace	coverage stores	wmsstore
style	coverage	
template	worldimage	
font		

WORKSPACE

```
gs-shell> geoserver set --url http://localhost:8080/geoserver

gs-shell> workspace create --name naturalearth
true

gs-shell> workspace list
topp
naturalearth

gs-shell> workspace get --name naturalearth
naturalearth
```

SHAPEFILE

```
gs-shell> geoserver set --url http://localhost:8080/geoserver

gs-shell> workspace create --name naturalearth
true

gs-shell> shapefile zip --shapefile 110m_admin_0_countries.shp
true

gs-shell> shapefile publish --file 110m_admin_0_countries.zip --
workspace naturalearth
true
```

POSTGIS

```
gs-shell> geoserver set --url http://localhost:8080/geoserver
gs-shell> workspace create --name post

gs-shell> postgis datastore create --workspace post --datastore postgis --host localhost --port 5432 --database postgres --schema public --user postgres --password postgres
true

gs-shell> featuretype list --workspace post --datastore postgis --list available
states
countries
cities

gs-shell> postgis featuretype publish --workspace post --datastore postgis --table 110m-admin-0-countries
true
```

GEOTIFF

```
gs-shell> geoserver set --url http://localhost:8080/geoserver

gs-shell> coverage store upload --workspace naturalearth --file
GRAY_50M_SR_OB.zip --coveragestore world --type worldimage --
configure first
true

gs-shell> coverage list --workspace naturalearth --coveragestore world
GRAY_50M_SR_OB
BLUE_MARBLE

gs-shell> coverage modify --workspace naturalearth --coveragestore world
--coverage GRAY_50M_SR_OB --srs EPSG:4326 --enabled true
true
```

STYLES

```
gs-shell> style create --file 110m_admin_0_countries.sld --name
110m_admin_0_countries
true

gs-shell> layer style add --name 110m_admin_0_countries --style
110m_admin_0_countries
true

gs-shell> layer get --name 110m_admin_0_countries
110m_admin_0_countries
  Title: null
  Type:VECTOR
  Abstract: null
  Default Style: 110m_admin_0_countries
  Namespace:    Type String:VECTOR

gs-shell> layer modify --name 110m_admin_0_countries --defaultStyle
110m_admin_0_countries
true
```

SCRIPTS

- Scripts contain the same commands entered interactively
- History support saves session to geoserver-shell.log

```
$ gs-shell --cmdfile mycommands.gs

gs-shell> script --file naturalearth_countries.gs
```

HOW DOES IT WORK?

- Spring Shell
- Geoserver Rest API
- GeoServer Manager and GeoTools Java Libraries

SPRING SHELL

- Open source project from Spring Source
- Shell extracted from Spring Roo
- Spring Roo is a Java RAD tool for creating web apps


```
mkdir hello
cd hello
roo
roo> hint
roo> project --topLevelPackage com.foo
roo> jpa setup --provider HIBERNATE --database HYPERSONIC_IN_MEMORY

roo> entity jpa --class ~.Timer --testAutomatically
roo> field string --fieldName message --notNull
roo> hint web mvc
roo> web mvc setup
roo> web mvc all --package ~.web
roo> selenium test --controller ~.web.TimerController
roo> web gwt setup
roo> web gwt all --proxyPackage ~.client.proxy --requestPackage ~.client
.request
roo> perform tests
roo> quit
```

SPRING SHELL

- Interactive shell environment
- Tab completion for commands, arguments, and files
- History support (up and down arrows)
- Knows your commands and when you can use them
- You just need to add commands (Java classes)

GEOSERVER REST API

- Administer GeoServer without GUI
- Multiple representations: HTML, JSON, XML
- HTTP Verbs: GET, POST, PUT, DELETE
- HATEOS - linking
- http://docs.geoserver.org/stable/en/user/rest/index.html
- Integrate with 3rd party applications

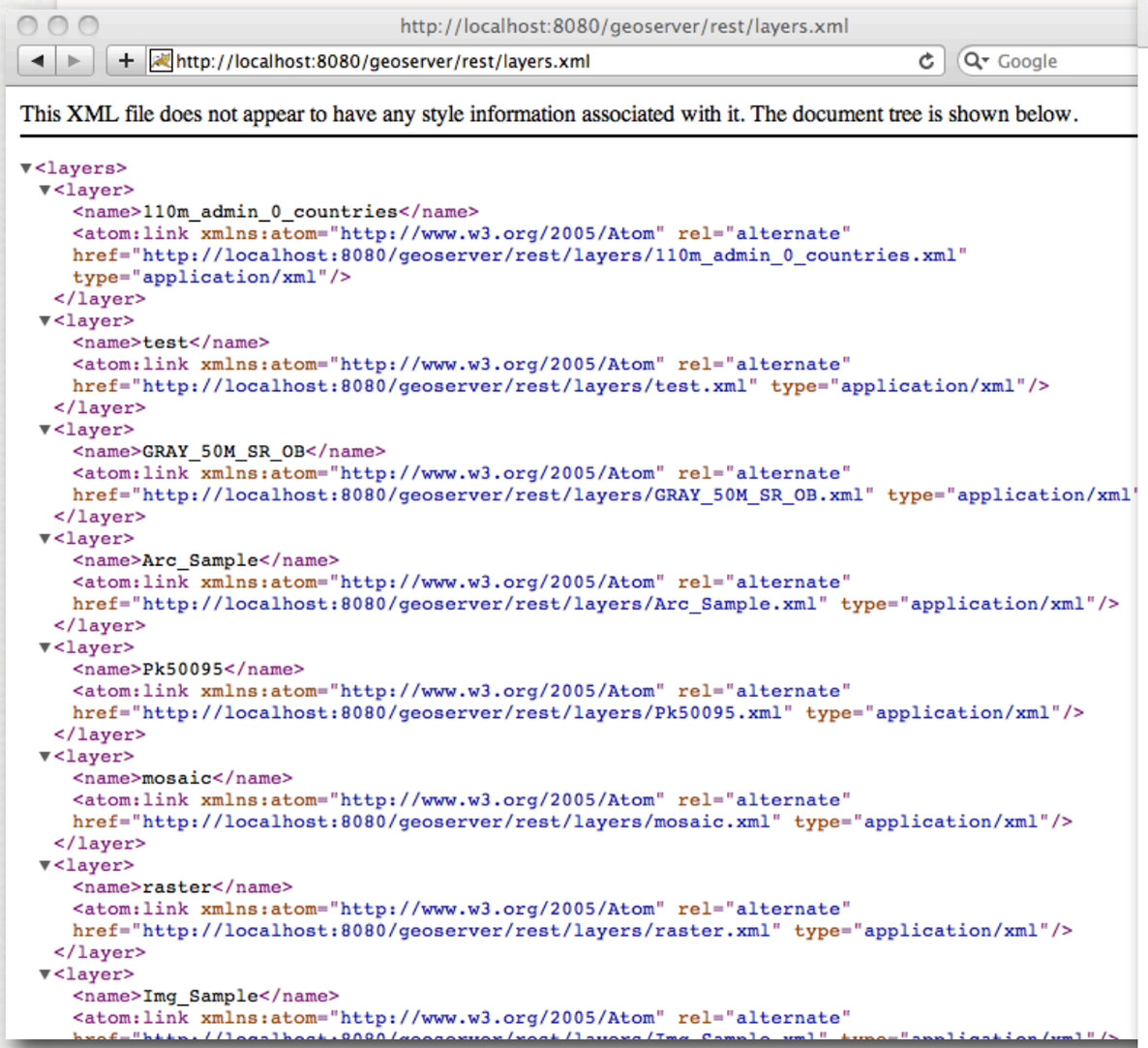
GEOSERVER REST

/geoserver/rest/layers

http://localhost:8080/geoserver/rest/layers.xml

http://localhost:8080/geoserver/rest/layers.xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.



localhost:8080/geoserver/

localhost:8080/geoserver/rest/layers.json


```
{
  - layers: {
      - layer: [
          - {
              name: "110m_admin_0_countries",
              href: "http://localhost:8080/geoserver/rest/layers/110m_admin_0_countries.json"
            },
          - {
              name: "test",
              href: "http://localhost:8080/geoserver/rest/layers/test.json"
            },
          - {
              name: "GRAY_50M_SR_OB",
              href: "http://localhost:8080/geoserver/rest/layers/GRAY_50M_SR_OB.json"
            },
          - {
              name: "Arc_Sample",
              href: "http://localhost:8080/geoserver/rest/layers/Arc_Sample.json"
            },
          - {
              name: "Pk50095",
              href: "http://localhost:8080/geoserver/rest/layers/Pk50095.json"
            },
          - {
              name: "mosaic",
              href: "http://localhost:8080/geoserver/rest/layers/mosaic.json"
            },
          - {
              name: "raster",
              href: "http://localhost:8080/geoserver/rest/layers/raster.json"
            },
          - {
              name: "Img_Sample",
              href: "http://localhost:8080/geoserver/rest/layers/Img_Sample.json"
            },
          - {
              name: "110m-admin-0-countries",
              href: "http://localhost:8080/geoserver/rest/layers/110m-admin-0-countries.json"
            },
          - {
              name: "archsites",
              href: "http://localhost:8080/geoserver/rest/layers/archsites.json"
            },
          - {
```

REST CLIENTS

```
curl -v -u admin:geoserver -XPUT -H "Content-type: application/zip"  --data-binary @roads.zip
http://localhost:8080/geoserver/rest/workspaces/acme/datastores/roads/file.shp
```

```
>>> from geoserver.catalog import Catalog
>>> cat = Catalog("http://localhost:8080/geoserver/rest", "admin",
"geoserver")
>>> cat.get_layers()
[Layer[Arc_Sample], Layer[Pk50095], Layer[Img_Sample], ...

>>> that_layer = cat.get_layer("roads")
>>> that_layer.enabled = False
>>> cat.save(that_layer)
```

```
GeoServerRESTReader reader = new GeoServerRESTReader(url, user, password);
RESTStyleList styleList = reader.getStyles();
List<String> names = styleList.getNames();

GeoServerRESTPublisher publisher = new GeoServerRESTPublisher(url, user, password);
publisher.createWorkspace(name);
```

HOW IT WORKS

- Geoserver Shell is a collection Spring Shell commands
- Each command constructs a URL from arguments pass in by the user
- The URL is passed to the GeoServer REST API
- The response is parsed and presented to the user

WHY?

- I wanted to learn the Geoserver REST API
- I wanted to build something with Spring Shell
- Open source is about scratching an itch
- Open Source is about sharing and collaborating and giving back.
- Open Source is about standing on the shoulders of giants. You almost never have to start from scratch.

THANK YOU!

Jared Erickson

jared.erickson@gmail.com

https://github.com/jericks

CUGOS UW Fall Fling 2013