GeoServer Web Based Configuration Implementation Report

Submitted To: Program Manager
GeoConnections
Victoria, BC, Canada

Submitted By: Jody Garnett
Refractions Research Inc.
Suite 400 – 1207 Douglas Street
Victoria, BC, V8W-2E7
jgarnett@refractions.net
Phone: (250) 383-3022
Fax: (250) 383-2140

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 INTRODUCTION

This document outlines our experience in implementing the GeoServer Web Based Configuration system. Several changes have been made with respect to the GeoServer Web Based Configuration Design Document.

The focus of these User Interface design changes has been:

- To allow specification of attribute type order during FeatureType editing
- The addition of a User Interface for Validation Configuration
- Revised Page layout featuring the separation of the Form Feedback from the status of the GeoServer application

The inclusion of the Validation Processor User Interface as part of the Web Based Configuration System has resulted in a seamless user experience.

This work has been highly anticipated by the GeoServer user community, and we have been pleased to provide a 1.2 beta release.

2 WEB BASED CONFIGURATION DESIGN

In subsequent sections the following diagram (Figure 1) from the GeoServer Web Based Configuration Design Document will be referred to.

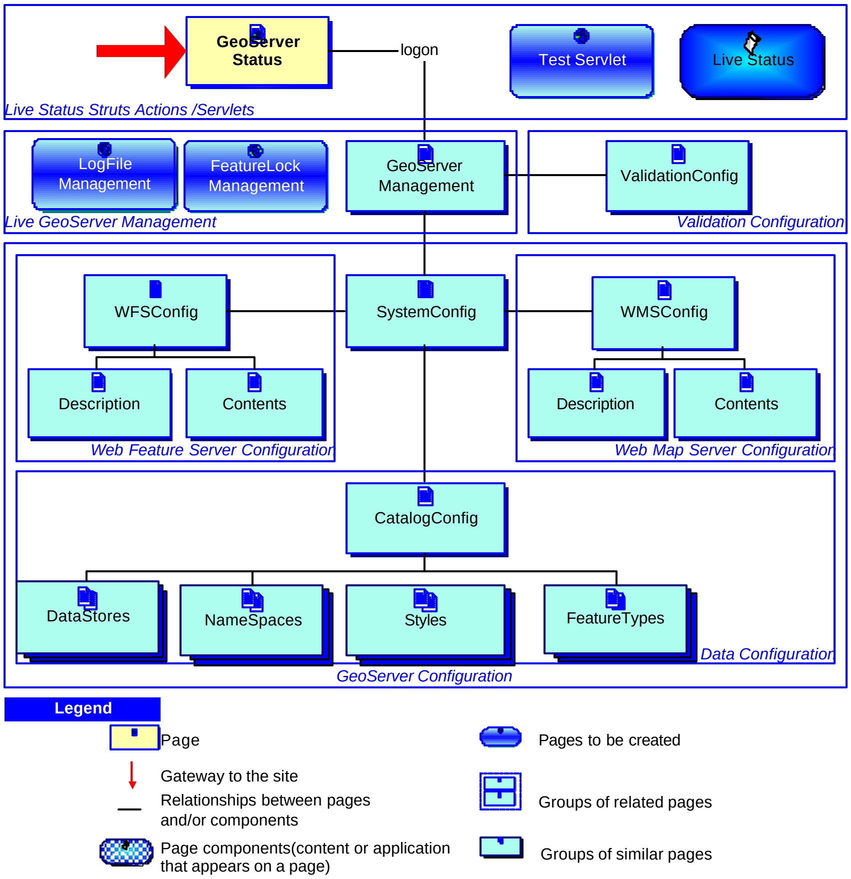


Figure 1 – Initial User Interface Page Diagram

2.1 Design Successes

The design for the GeoServer Configuration System has succeeded in:

- Providing GeoServer Web Based Configuration
- Providing dynamic feedback from the GeoServer application
- Providing a consistent user experience

2.2 Design Extensions

In several instances we have expanded upon the design outlined in the Web Based Configuration Design Document.

These changes have resulted from:

- An increased familiarity with the Struts Application Framework
- The complexity of FeatureType Schema Specification
- The addition of the Validation Processor to the User Interface

2.2.1 Revised Page Layout

The page layout has been revised to provide a consistent area to report back Form validation errors and Status information.

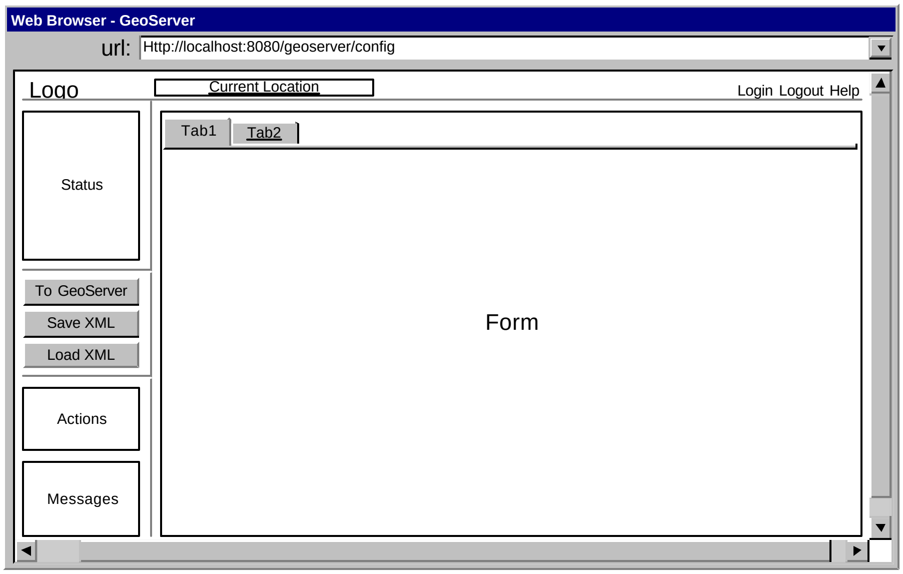
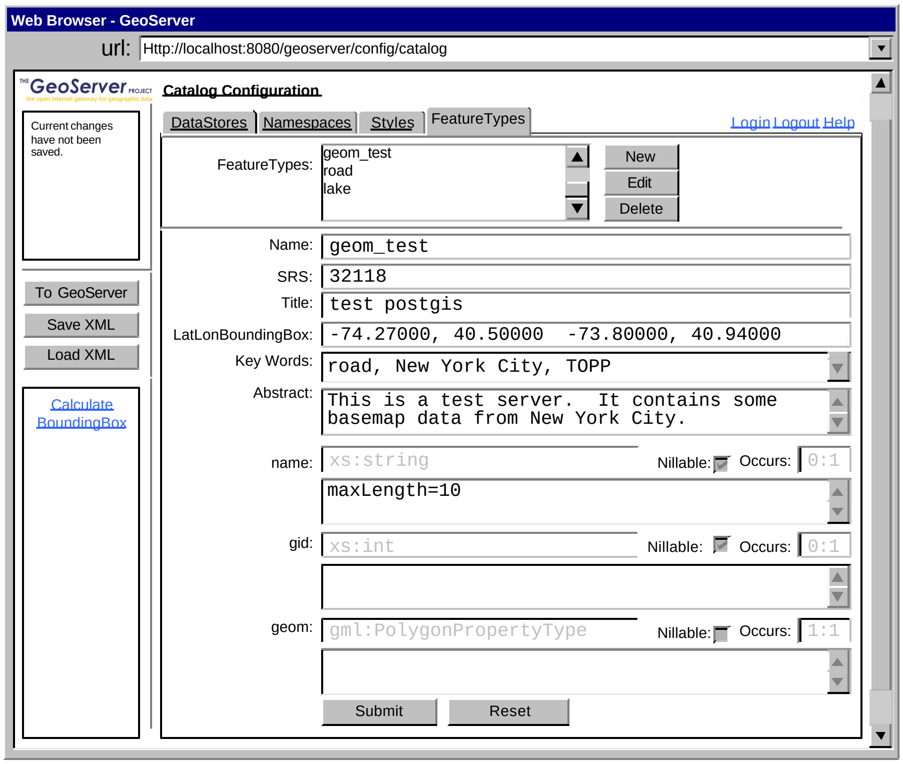


Figure 2 – Revised User Interface Design

This design is expected to undergo a few minor revisions in response to feedback from a graphic design team. In order to respond to these suggestions we may make use of additional HTML rendering technologies such as Cascading Style Sheets.

2.2.2 FeatureType Schema Specification

We had overlooked the need to specify attribute order (Figure 3).



url: Http://localhost:8080/geoserver/config/catalog

Current changes have not been saved.

FeatureTypes: geom_test, road, lake

Name: geom_test

SRS: 32118

Title: test postgis

LatLonBoundingBox: -74.27000, 40.50000 -73.80000, 40.94000

Key Words: road, New York City, TOPP

Abstract: This is a test server. It contains some basemap data from New York City.

name: xs:string Nillable: ☑ Occurs: 0:1

maxLength=10

gid: xs:int Nillable: ☑ Occurs: 0:1

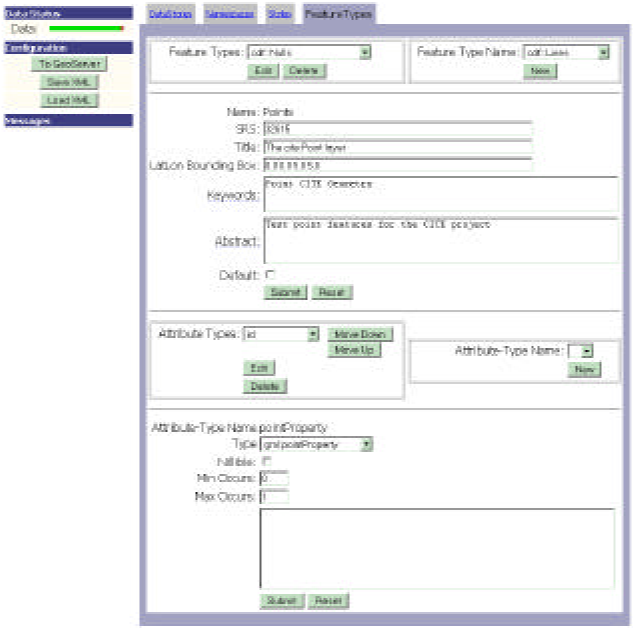
geom: gml:PolygonPropertyType Nillable: ☐ Occurs: 1:1

Figure 3 – Original Catalog Feature Type Design

To accommodate the ordering of Attributes we have made several additions:

- A “default” setting used to control schema generation
- An attribute Select tile with Delete, Move Up and Move Down actions
- New Attribute and Edit Attribute tiles

Modifications for Attribute Type ordering are illustrated by the following screen capture (Figure 4).



Feature Types: cite:Nulls
Feature Type Name: cite:Lakes
Name: Points
SRS: 32615
Title: The cite Point layer
LatLon Bounding Box:
Keywords: Point CITE Geometry
Abstract: Test point features for the CITE project
Default: □
Attribute Types: id
Attribute-Type Name:
Attribute-Type Name pointProperty
Type gml:pointPropertyType
Nillable: □
Min Occurs: 0
Max Occurs: 1

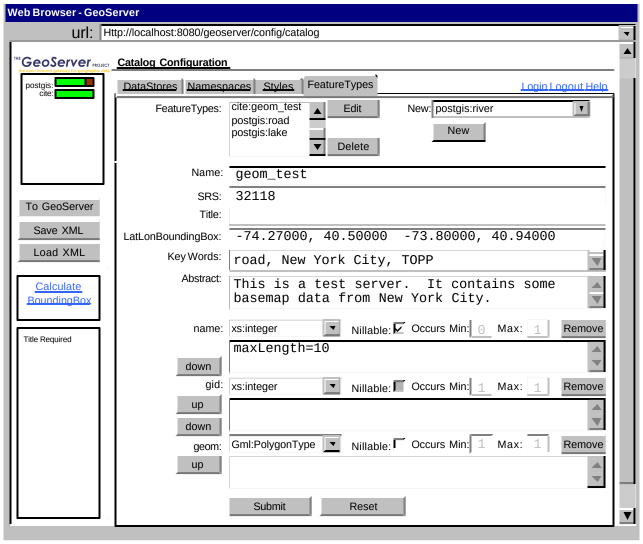
Figure 4 – Current Catalog Feature Type Screen

This extension allows for the required attribute ordering capability.

Several drawbacks are apparent:

- Attribute specification is visually separated from FeatureType (apparent in the location of the submit button above the Attribute Select, New and Edit tiles).
- Only one attribute is visible at a time

It is recommended that the initial design be restored with the inclusion of "Up", "Down" and "Delete" buttons next to each AttributeType (Figure 5).



url: Http://localhost:8080/geoserver/config/catalog

FeatureTypes: cite:geom_test
postgis:road
postgis:lake
New: postgis:river
Name: geom_test
SRS: 32118
Title:
LatLonBoundingBox: -74.27000, 40.50000 -73.80000, 40.94000
Key Words: road, New York City, TOPP
Abstract: This is a test server. It contains some basemap data from New York City.

name: xs:integer Nillable: ☑ Occurs Min: 0 Max: 1
maxLength=10
gid: xs:integer Nillable: ■ Occurs Min: 1 Max: 1
geom: Gml:PolygonType Nillable: □ Occurs Min: 1 Max: 1

Figure 5 – Feature Type Configuration Recommendation

2.2.3 Validation Framework Extension

The extension of our initial design for the validation processor was accomplished without incident.

The following diagram illustrates the extensions to the GeoServer layer diagram for the Validation Processor.

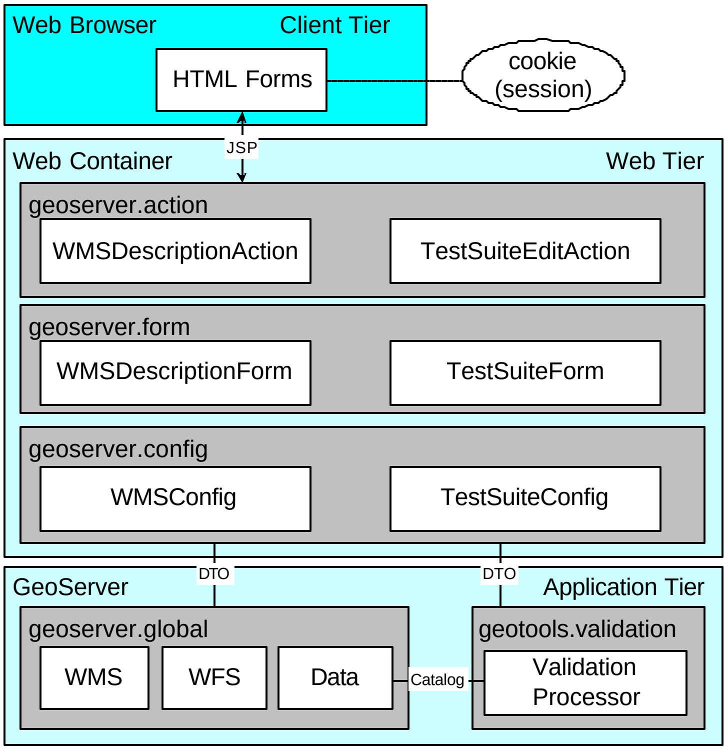


Figure 6 – Validation Extension Layer Diagram

The following table lists the Validation classes and interfaces by package:

Validation	Validation DTO	Config	Form
ValidationProcessor		ValidationConfig	
PlugIn	PlugInDTO	PlugInConfig	
TestSuite	TestSuiteDTO	TestSuiteConfig	ValidationTestSuiteNewForm ValidationTestSuiteSelectForm
Validation	TestDTO	TestConfig	ValidationTestEditorForm ValidationTestNewForm ValidationTestSelectForm

Table 1 – Validation Processor Data Matrix

We have maintained a clear separation between the Validation Processor and the GeoServer application through the use of the Catalog API

2.3 User Interface Design Recommendations

Over the course of development we have arrived at several recommendations for the design of the existing user interface application.

These recommendations will be placed in the GeoServer Jira bug tracking system (http://jira.codehaus.org/).

Several of these design recommendations result in a modified application workflow (Figure 7).

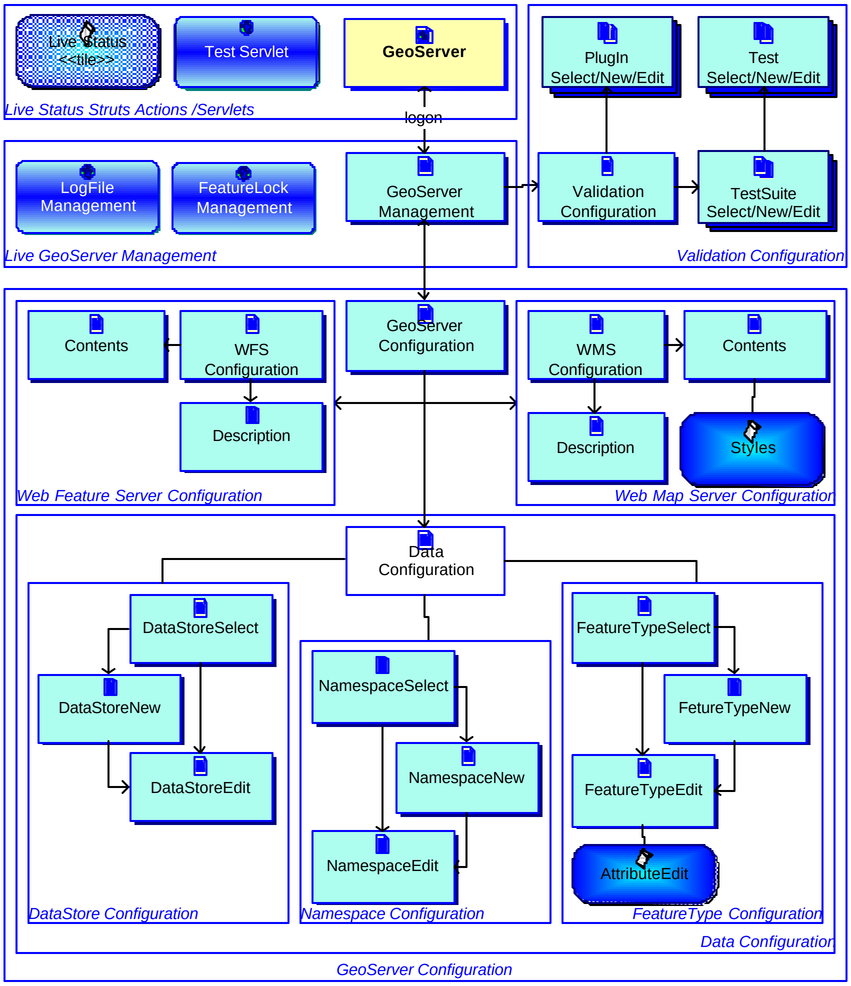


Figure 7 – Revised User Interface Page Diagram

Summary of workflow changes:

- Distinct page for Select, New and Edit pages
- Select Page redirects to New Page when empty
- Catalog pages renamed to Data

2.3.1 Separate Page for New Actions

By separating the New Actions for DataStore, Namespace and FeatureType we can reduce the complexity of these already very crowded screens.

In addition we can provide a redirect to the appropriate New Action in the case where no content exists to be selected.

2.3.2 Extended Status Reporting

The logs provided by GeoServer configuration provide a nice summary of any problems encountered during setup. This information has been included as part of the Configuration pages and is updated during the “to GeoServer” action.

The `dataStoreId:typeName` used to list configuration problems should be linked to the appropriate Edit page where the problem can be fixed.

As part of this transition Struts can include the error message for display.

2.3.3 Visual Appearance

The first cut of the web interface has been focused on functionality, often using the default appearance of native Struts widgets.

In the next phase of development we will take command of this process in order to address any issues raised by the aesthetic review.

During the review process we will be looking for feedback on:

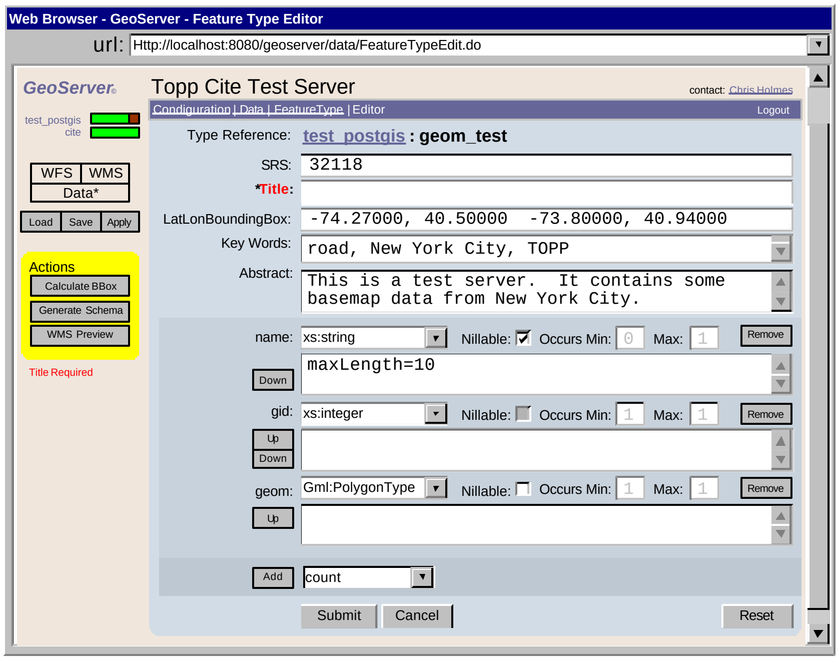
- The efficient use of screen real estate
- The consistent location of Navigation components
- The use of colors, icons and fonts

The emphasis will remain on functionality and efficiency.

Initial thoughts on visual appearance:

- Use a neutral medium background for the page
- Use a distinct color for the current page focus depending on application area:
  – lighter background for pages dealing with information reporting
  – darker background for configuration pages
- Uses a darker tile to indicate more detailed parameters
- Keep status, and configuration controls "flush" with this background in order not to detract from the current focus
- Modify Actions tile to makes use of a "yellow-sticky" or slight 3D effect

An initial mock up of these changes is show below (Figure 8).



Web Browser - GeoServer - Feature Type Editor

url: Http://localhost:8080/geoserver/data/FeatureTypeEdit.do

Topp Cite Test Server

contact: Chris Holmes

Condiguration | Data | FeatureType | Editor

Type Reference: test_postgis : geom_test

SRS: 32118

*Title:

LatLonBoundingBox: -74.27000, 40.50000 -73.80000, 40.94000

Key Words: road, New York City, TOPP

Abstract: This is a test server. It contains some basemap data from New York City.

name: xs:string Nillable: ☑ Occurs Min: 0 Max: 1

maxLength=10

gid: xs:integer Nillable: □ Occurs Min: 1 Max: 1

geom: Gml:PolygonType Nillable: □ Occurs Min: 1 Max: 1

Title Required

Figure 8 – Recommended Visual Appearance

3 IMPLEMENTATION RECOMMENDATIONS

3.1.1 Enhanced Authentication Support

The existing configuration design depends on a simple access control. Access is restricted based on the presence of a UserContainer object located in Session. Management of the UserContainer is handled through login and logout actions.

We need to ensure that each Configuration Action provides a login check and an appropriate redirect to the Login page.

Example ConfigAction with Login Check:

```
class MyConfigAction extends ConfigAction {
  Redirect execute( HttpServletRequest request,  ){
    if( !isLoggedIn( request )){
      return new Redirect(“Login Page”);
    }
    UserContainer user = getUserContainer( request );
    …
    return new Redirect(“my.jsp”);
  }
}
```

This boilerplate code can be removed through the use of enhancing the ConfigAction base class.

It is recommend that we modify the base ConfigAction class to provide the logged-in check in the execute method. Derived classes can make use of a custom execute method that provides the UserContainer as a parameter.

Enhanced ConfigAction with logged-in check:

```
class ConfigAction extends GeoServerAction {
  Redirect execute( HttpServletRequest request,  ){
    if( !isLoggedIn( request )){
      return new Redirect(“Login Page”);
    }
    return execute(getUserContainer( request ), request );
  }
  Redirect execute( UserContainer user, HttpServletRequest request ){
    return null;
  }
}
class MyConfigAction extends ConfigAction {
  Redirect execute( UserContainer user, HttpServletRequest request ){
    …
    return new Redirect(“my.jsp”);
  }
}
```

Note that the extended execute method has not been made abstract; this allows subclasses of ConfigAction to implement execute in the usual manner if needed.

3.1.2 Move New DataStoreConfig to Session

During the process of setting up a DataStore for the first time a DataStoreConfig object is created and placed under control of DataConfig.

This order of execution always results, at least for a short while, in DataConfig managing a DataStore that is incapable of working. It would be preferable to store the new DataStoreConfig in the UserContainer until such time as it works.

3.1.3 Provide Feature Lock Control

Management of Transaction Web Feature Server locks is recommended by the OGC WFS Specification. Currently GeoServer does not provide a mechanism through which Locks can be released manually.

This capability should be added to the GeoServer management page.