GeoServer Web Based Configuration Design

Submitted To: Program Manager
GeoConnections
Victoria, BC, Canada

Submitted By: Jody Garnett
David Zwiers
Richard Gould
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

INTRODUCTION

This document represents our proposal for extending the GeoServer application with a Web Based Configuration system.

GeoServer makes use of a series of XML files to store configuration information. To configure the GeoServer application, these files are edited and the server restarted. This is often a time-consuming and frustrating process.

The GeoServer configuration process will be improved on two fronts:

- Web Based User Interface - - STRUTS based web interface, using Tiles for a consistent layout.
- Server State - - Configuration will be separated from the Server State.

An exciting new capability of this system will be the dynamic configuration of the GeoServer application. By allowing the configuration to be modified as the GeoServer application is running, users will get immediate feedback and the initial frustrating of installation will be eased.

A strong separation is maintained between the Configuration system and the GeoServer application even though they are maintained as a single entity. This separation has been requested to help with future improvements to the Configuration System and Application portions of GeoServer. The strong separation will also allow the Configuration System to be broken out as a separate web application (using JMX for GeoServer management) should the need arise.

1 CURRENT GEOSERVER CONFIGURATION

GeoServer has recently changed its configuration as part of a Web Map Server integration effort. As such the current design is of recent vintage and sparsely documented.

1.1 Current Configuration Workflow

The current configuration loads a series of configuration XML files into:

- ServerConfig
- WMSConfig: Web Map Server information
- WFSConfig: Web Feature Server information (used in the GetCapabilites)
- CatalogConfig: FeatureType and Namespace information

The existing system has the data stored in multiple classes, all found in the `org.vfny.geoserver.config` package.

Each class has three main purposes:

- to store data required by the application
- to provide data to the application
- to populate configuration data from XML files

The configuration data is imported into the system exactly once at startup. The data is then served to the application in various forms.

In most cases, it is not possible to modify application configuration dynamically as GeoServer is running. The exception occurs in the CatalogConfig class, where references to XMLSchema files are passed to the DescribeFeatureType response.

2 PROPOSED GEOSERVER CONFIGURATION DESIGN

After reviewing the current GeoServer configuration design we developed the following recommendations. We found limitations of the existing GeoServer Configuration System, and plan to address these limitations by including the following requirements:

- Separate out the Configuration Model from the GeoServer Application
- Build a Web Interface against the Configuration Model
- Allow persistence of the GeoServer Application state to XML

We have been provided with the following guidelines:

- Leave the existing configuration file format intact
- Use Struts Web Framework for web application development

2.1 Layer Description

GeoServer has an existing package structure that does not mesh smoothly with a Struts based application. We have develop a parallel package structure for Struts based work.

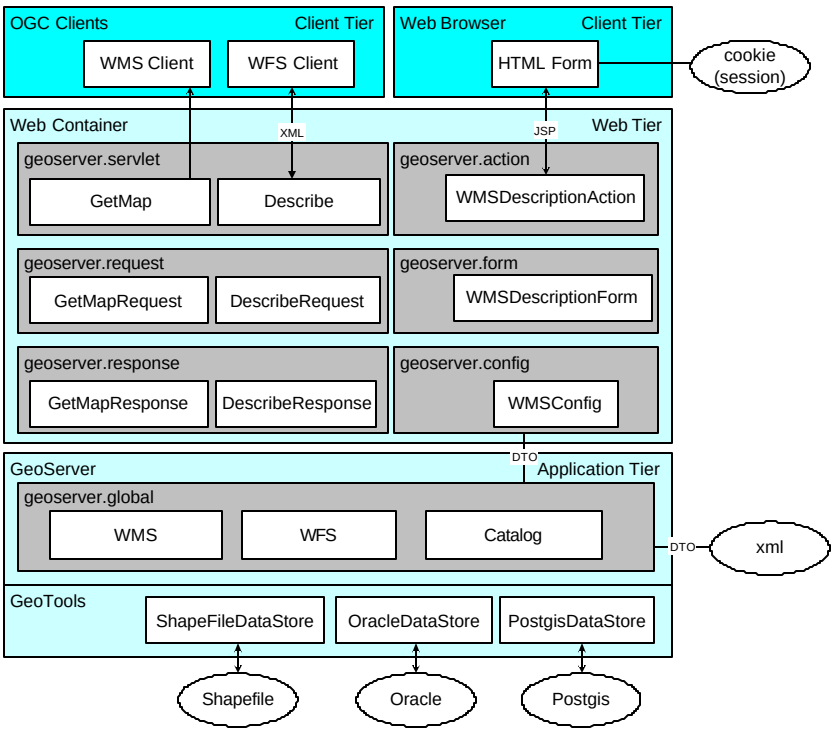

Figure 1 — GeoServer Layer Diagram

The classes in the config, form and action packages conform to the Struts modified Model-View-Control Architecture. These classes are isolated from the existing GeoServer classes.

2.2 GeoServer Data Structures

Information in GeoServer is split across several layers:

- Information used by the running GeoServer application
- The Configuration information being modified by the user
- The XML Configuration files used for Persistence
- Form Beans used to represent the information displayed in a Web Browser
- Data Transfer Objects used to communicate between GeoServer and the Configuration and Persistence subsystems

The following table lists the classes used by GeoServer by both package and subsystem.

Global	DTO	Config	Form
WMS	WMSDTO	WMSConfig	WMS*Form
WFS	WFSDTO	WFSConfig	WFS*Form
GeoServer	GeoServerDTO	GeoServerConfig	GeoServer*Form
Contact	ContactDTO	ContactDTO	
Data *(Catalog)*	DataDTO	DataConfig	Data*Form
Namespace	NamespaceDTO	Namespace	*Form
Styles	StyleDTO	Style	
DataStoreInfo	DataStoreInfoDTO	DataStoreInfo	
FeatureTypeInfo	FeatureTypeInfoDTO	FeatureTypeInfo	
Validation	--	--	--

Table 1 — Data Matrix

2.3 Data Packages

Although all GeoServer Data classes hold equivalent information, they different in intent, abstraction and dependency.

2.3.1 Global Package

The Global package represents the state of the GeoServer Web Application.

- Initially implemented as singletons, we intend to migrate to storage in the Web Container
- The Global package manages the state of the application including GeoTools2 Database connections
- The Global package is not allowed to communicate (or depend) on the Configuration System, XML persistence, or any other subsystem
- Other subsystems may communicate with Global via the DTO objects

As GeoTools2 develops, many of these classes will implement well-known GeoTools2 interfaces. The only current example of this is the Data class that implements the GeoTools2 Catalog Interface

2.3.2 DTO Package (Data Transfer Objects)

Data Transfer Objects are used to communicate with Global.

- Implemented as a package in global
- The XML Persistence and Config subsystem used Data Transfer Objects to modify the state of Global.
- The Config is setup using Data Transfer Objects requested from Global

2.3.3 Config Package

Represents the Model of the STRUTS based Model-View-Controller configuration system.

- Will support additional state used to drive the user interface
- Makes use of GeoTools2 Catalog/DataStore for FeatureType type names and FeatureType schema information/generation.

2.3.4 Form Package

Standard STRUTS Form Beans are used to represent User Form Input.

- Supply a validate and reset method
- May be able to replace Form Beans with STRUTS DynaBeans (that use Maps)

Form Beans are often fine-grained in order to directly match the fields displayed on the user's browser. Often a model from the Config package is split across several form beans.

2.4 GeoServer Configuration Information

To illustrate the information maintained by the GeoServer application we have included a quick outline of the Data Transfer Objects used to communicate with the Global package.

2.4.1 WMSDTO

```
public class WMSModel {
      private Date updateTime = new Date();
      private Service service;
}
```

2.4.2 WFSDTO

```
public class WFSDTO extends ServiceDTO {
      private String describeUrl;
      private Service service;
}
```

2.4.3 ServiceDTO

```
public class ServiceDTO {
      private boolean enabled = true;
      private String serviceType;
      private String onlineResource;
      private URL url;
      private String name;
      private String title;
      private String _abstract;
      private List keywords;
      private String fees;
      private String accessConstraints = "NONE";
      private String maintainer;
}
```

2.4.4 Global DTO

```
public class GlobalModel {
     private int maxFeatures = 20000;
     private boolean verbose = true;
     private int numDecimals = 8;
     private Charset charSet;
     private String baseUrl;
     private String schemaBaseUrl;
     private Contact contact = null;
}
```

2.4.5 Contact DTO

```
public class ContactModel {
    private String contactPerson;
    private String contactOrganization;
    private String contactPosition;
    private String addressType;
    private String address;
    private String addressCity;
    private String addressState;
    private String addressPostalCode;
    private String addressCountry;
    private String contactVoice;
    private String contactFacsimile;
    private String contactEmail;
}
```

2.5 Data DTO Objects

2.5.1 DataDTO

```
public class DataDTO {
      private Map dataStores;
      private Map nameSpaces;
      private Map features;
      private Map styles;
      private NamespaceSupport defaultNameSpace;
}
```

2.5.2 DataStoreInfoDTO

```
public class DataStoreInfoDTO {
      private String id;
      private NamespaceSupport nameSpace;
      private boolean enabled;
      private String title;
      private String _abstract;
      private Map connectionParams;
}
```

2.5.3 FeatureTypeInfoDTO

```
public class FeatureTypeDTO {
      private DataStore dataStore;
      private Envelope latLongBBox;
      private int SRS;
      private FeatureSchema schema;
      private Map styles;
      private String name;
      private String title;
      private String _abstract;
      private List keywords;
}
```

3 GeoServer Web Configuration User Interface

This section contains information about the GeoServer Web Configuration User Interface. The intended workflow and user interface design are presented.

The proposed new GUI will allow a user to create the GeoServer XML configuration files using a friendly web-based interface. The GeoServer Web Configuration User Interface will provide a configuration for WFS, WMS and Catalog systems.

The GeoServer development team has provided guidance in the selection of the STRUTS application framework. We will explore the design of the STRUTS framework and the implications for the GeoServer Web Configuration User Interface.

3.1.1 Model-View-Controller

Model-View-Controller (MVC) is a traditional design pattern used in Object-Oriented design since the early days of Smalltalk.

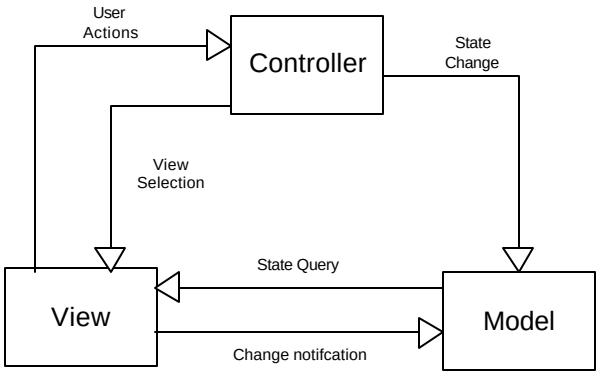

Figure 2 — Model-View-Controller Design Pattern

Model-View-Controller has several advantages:

- Separation of concerns between the Model, View, and Controller
- Uses notify/subscribe protocol and the Observer pattern between Model and View
- Allows multiple Dynamic Presentations
- Consolidate Control

For web based development strict MVC cannot be used due to limitations of the HTTP protocol. The notify/subscribe notification cannot be used and server applications cannot push change notification to the web client.

3.1.2 Modified Model-View-Controller

The STRUTS Application Framework makes use of a modified Model-View-Controller design.

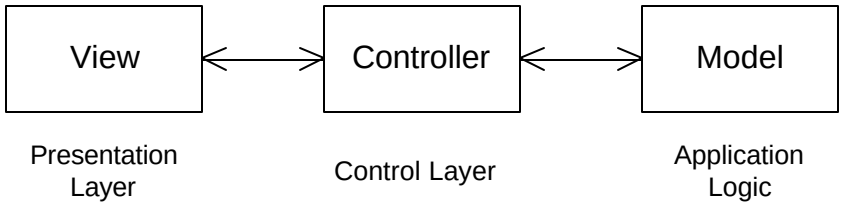

Figure 3 — Modified Model-View-Controller Design Pattern

To address the limitations of the MVC design STRUTS, and indeed many web applications, make use of a flattened Model-View-Control design; in which all model-view communication is funneled through the controller.

4 GeoServer Web Configuration Design

Each configuration area of the interface will be processed through the ActionForm. The ActionForm class acts as the Controller in STRUTS based applications. The configuration information will be stored in memory using a series of Java Beans representing our Configuration Model.

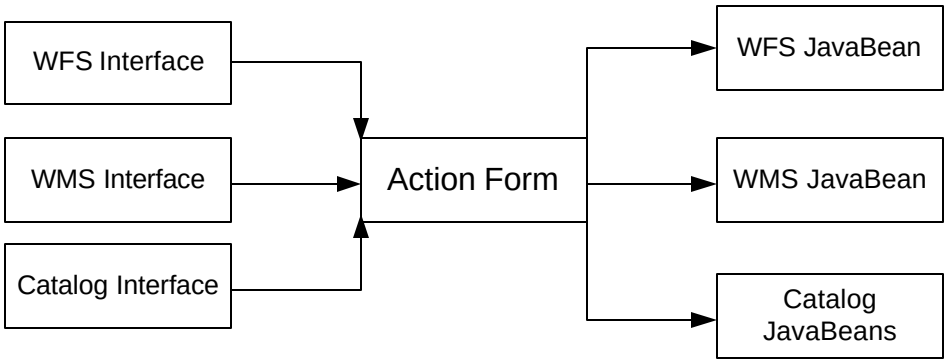

Figure 4 — Interface Flow Diagram

4.1 User Interface Design

STRUTS makes use of a framework called Tiles, which allows for the separation of web page layout from content. As a starting place we will be making use of the following layout.

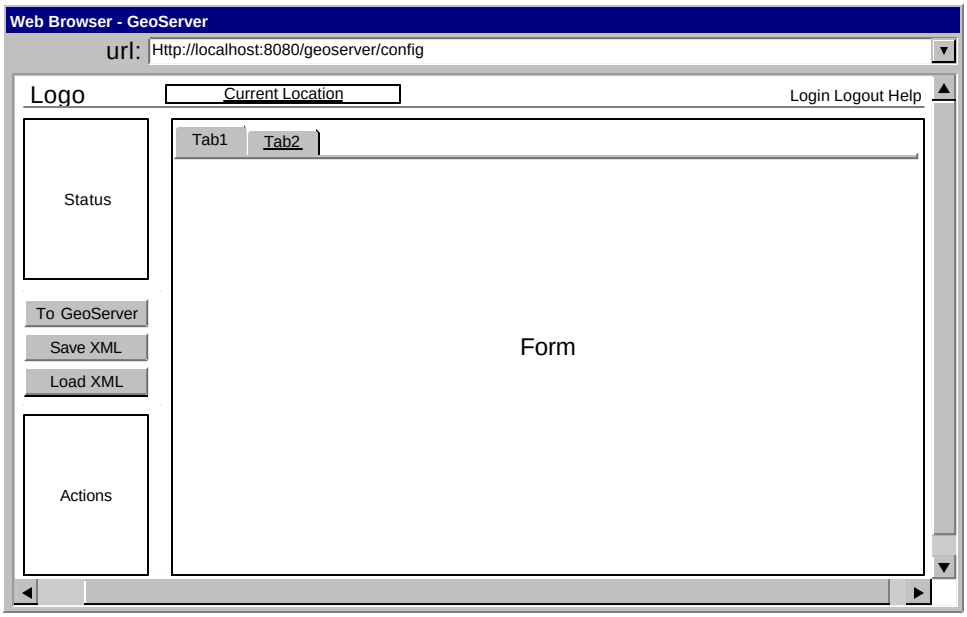

Figure 5 — User Interface Template

Features of this layout:

- Divided into relevant pages accessible through tabs at the top of the screen
- GeoServer status information updates are contained in the top — left
- Global Operations are displayed at the left-middle
- Local “actions” are displayed in the bottom-middle
- The current form is also displayed

4.2 GeoServer Web Configuration Workflow

The basic layout of the configuration interface is presented in Figure 6.

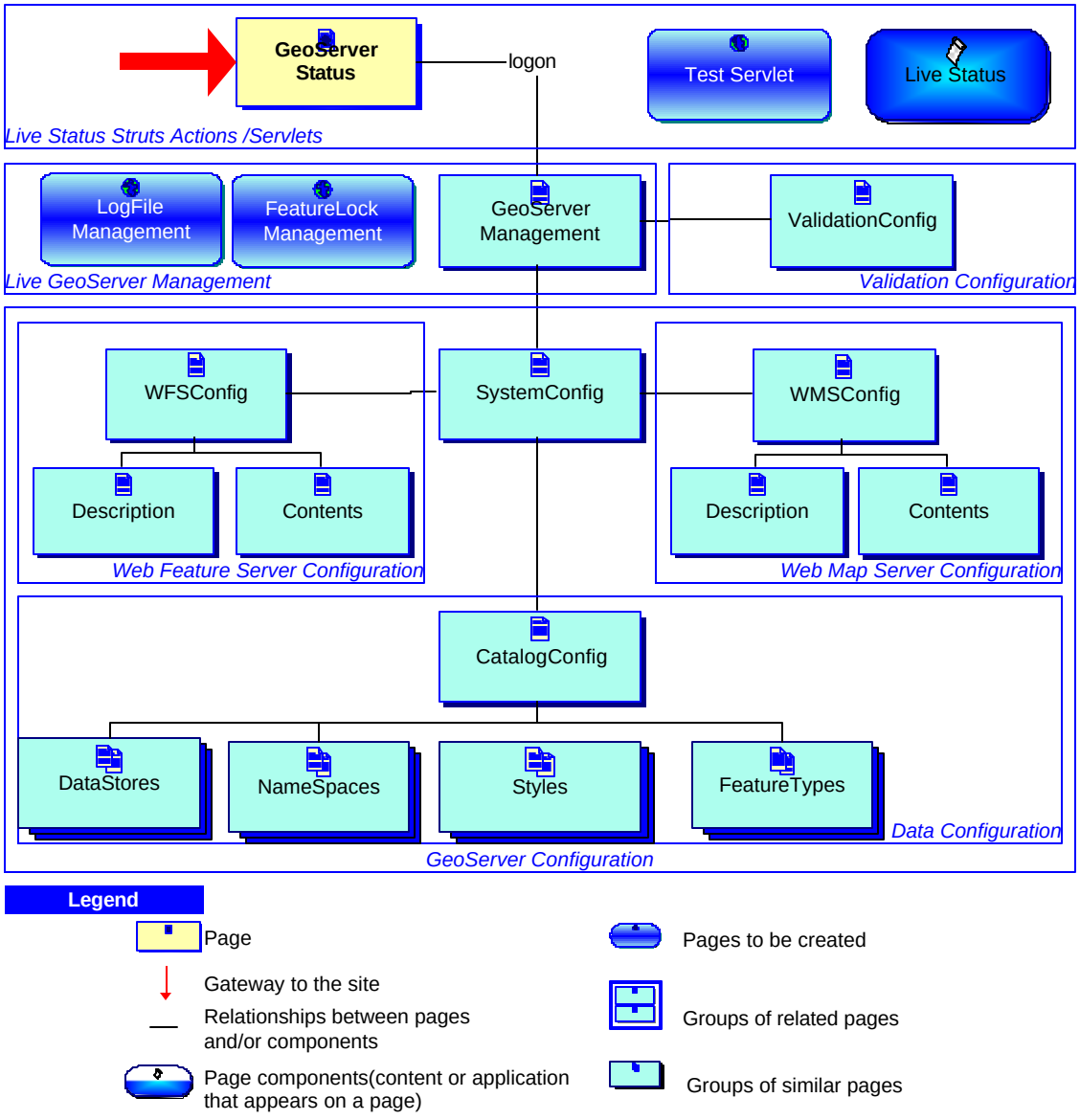

Figure 6 — Interface Page Layout

Once the user is logged in, they have the option of accessing the WFS configuration, the WMS configuration or the Catalog configuration.

5 SAMPLE GEOSERVER CONFIGURATION WEB PAGES

5.1 WFSConfig

The WFS configuration is divided into two pages, Description (Figure 7) and Content (Figure 8).

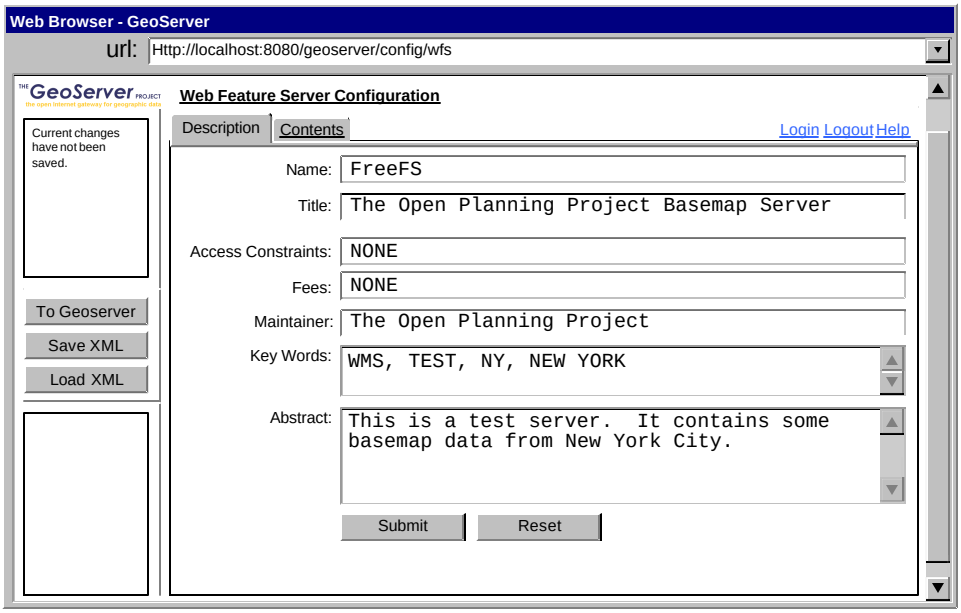

Figure 7 — WFSConfig Page 1: Description

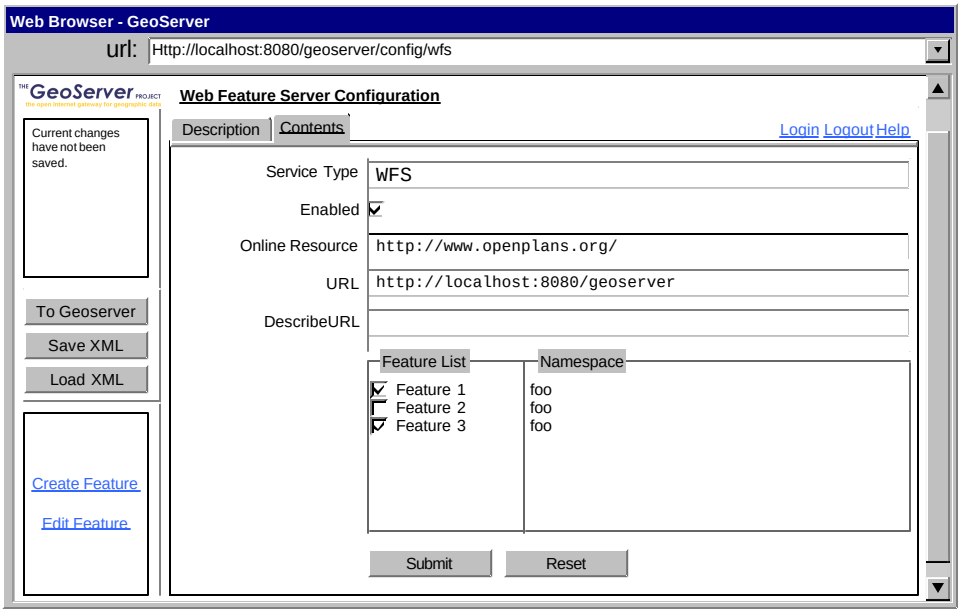

Figure 8 — WFSConfig Page 2: Contents

5.2 WMSConfig

The WMS configuration is almost identical to the WFS configuration. The Contents page features an 'updateTime' field instead of a 'describeURL' field.

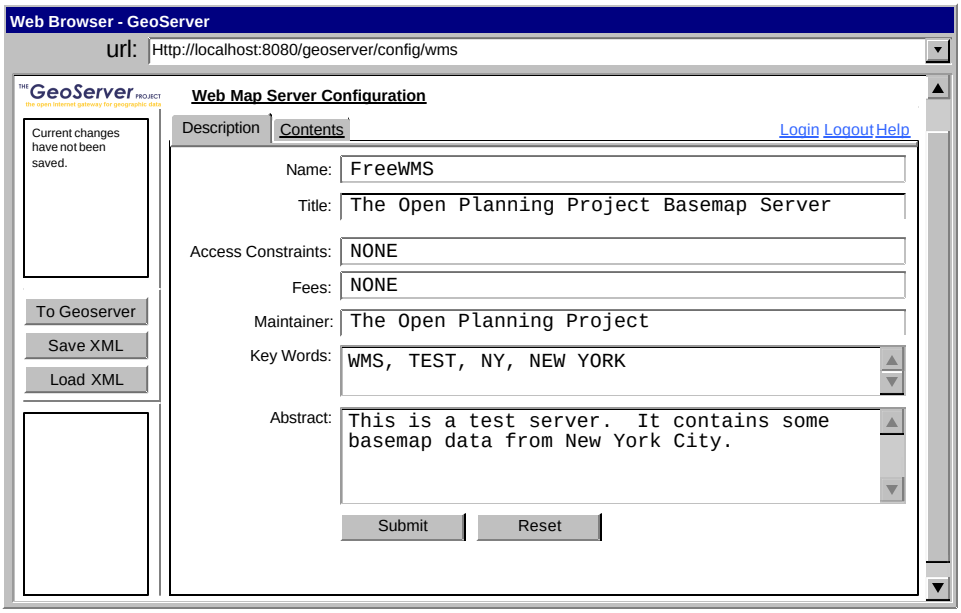

Figure 9 — WMSConfig Page 1: Description

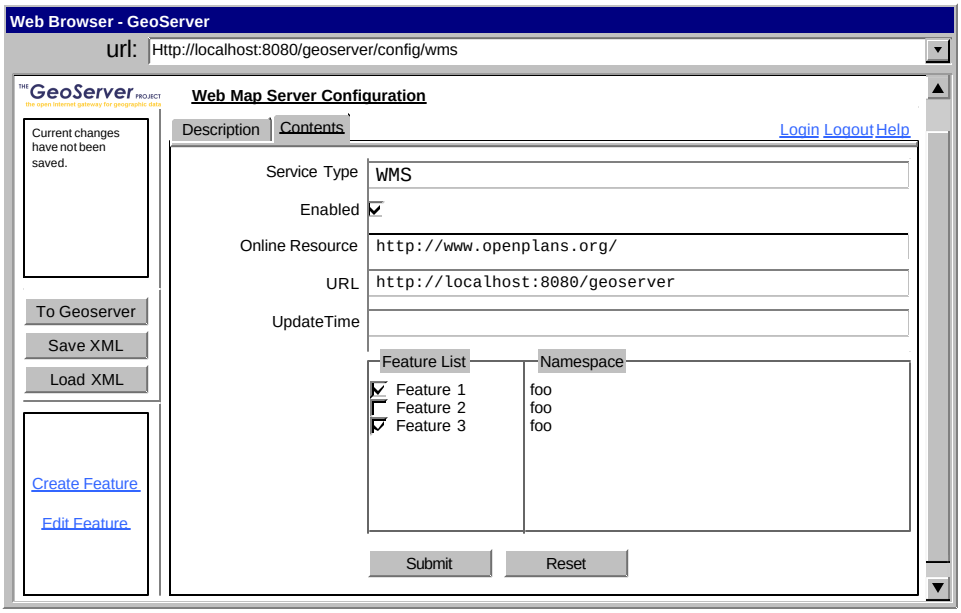

Figure 10 — WMSConfig Page 2: Contents

5.3 CatalogConfig

Common Elements located at the top of each page are:

- A List of available Objects
- An Edit button
- New and Delete buttons for list management
- A Form providing Object definition

Many of the form elements are dynamically generated from DataStore or FeatureType metadata.

5.3.1 DataStores

DataStore definition makes use of the GeoTools DataStoreFactorySPI to provide a list of available DataStores in a select control next to the New Button.

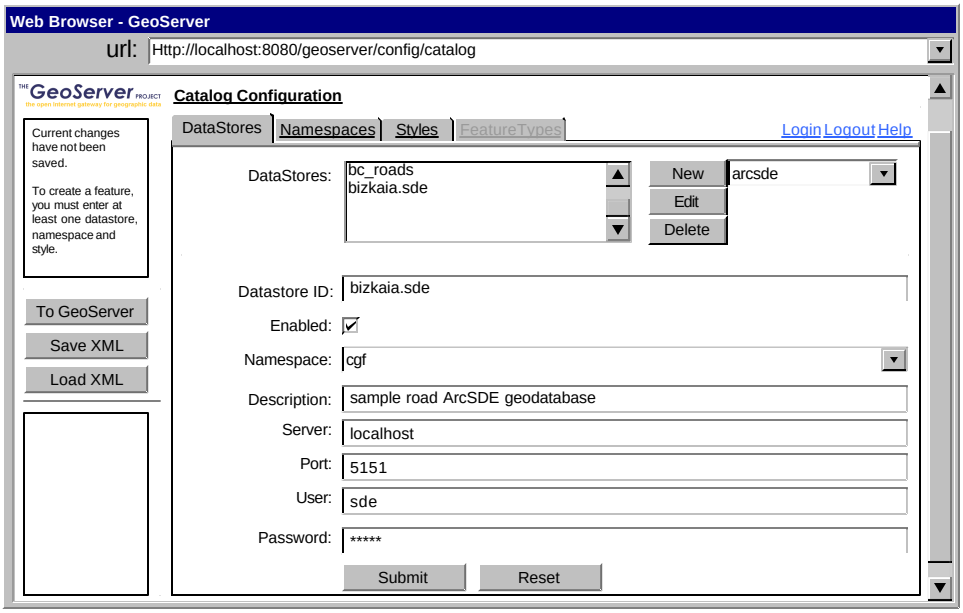

Figure 11 — Catalog Configuration DataStores

The contents of the form are DataStore dependent. The above form illustrates the needs of an ArcSDE DataStore while a Shapefile DataStore will simply need a URI.

5.3.2 Namespaces

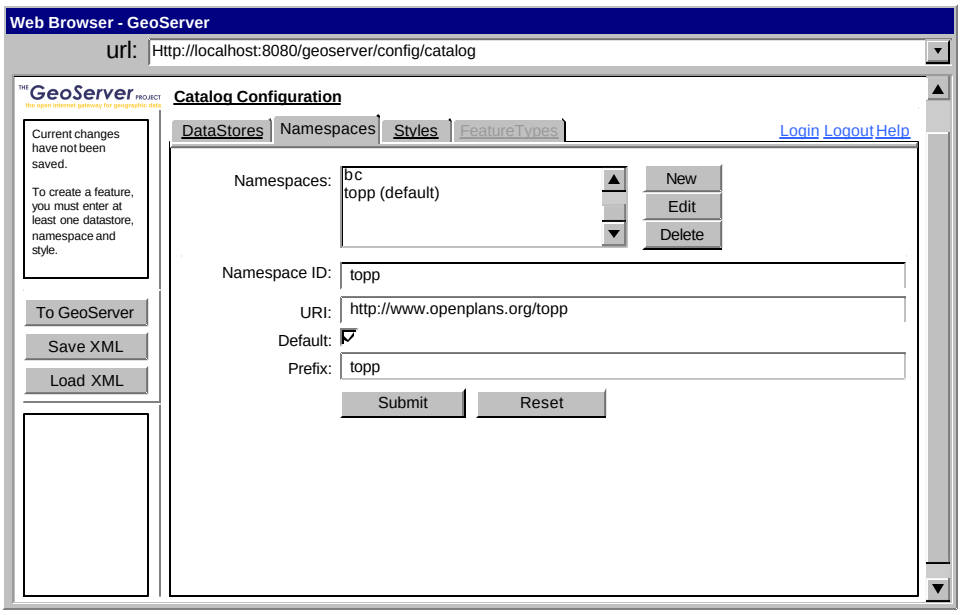

Figure 12 — Catalog Configuration Namespaces

5.3.3 Styles

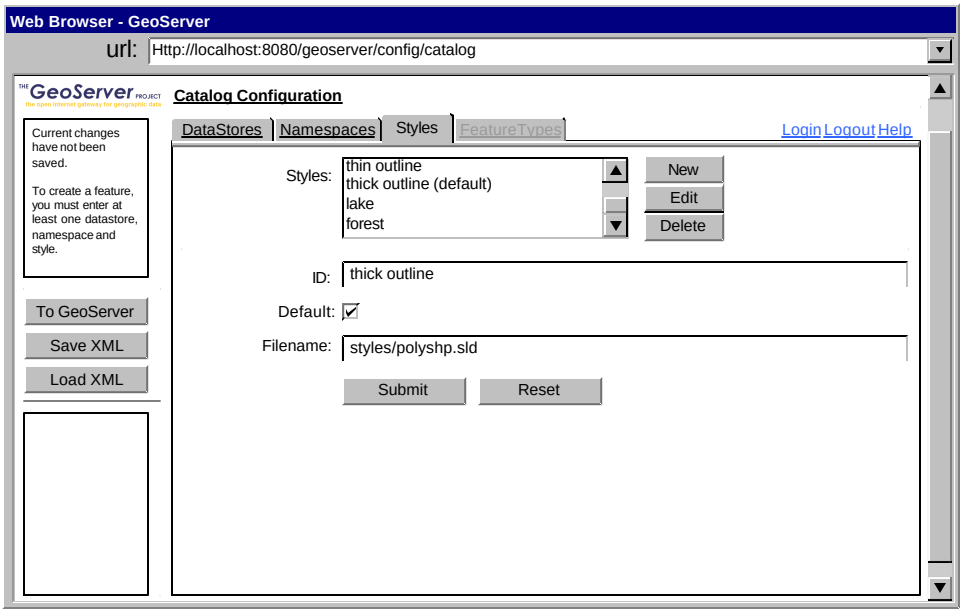

Figure 13 — Catalog Configuration Styles

5.3.4 FeatureType

The FeatureType form is dynamically generated from the schema information provided by GeoTools.

A single action, calculate bounding box, has been provided.

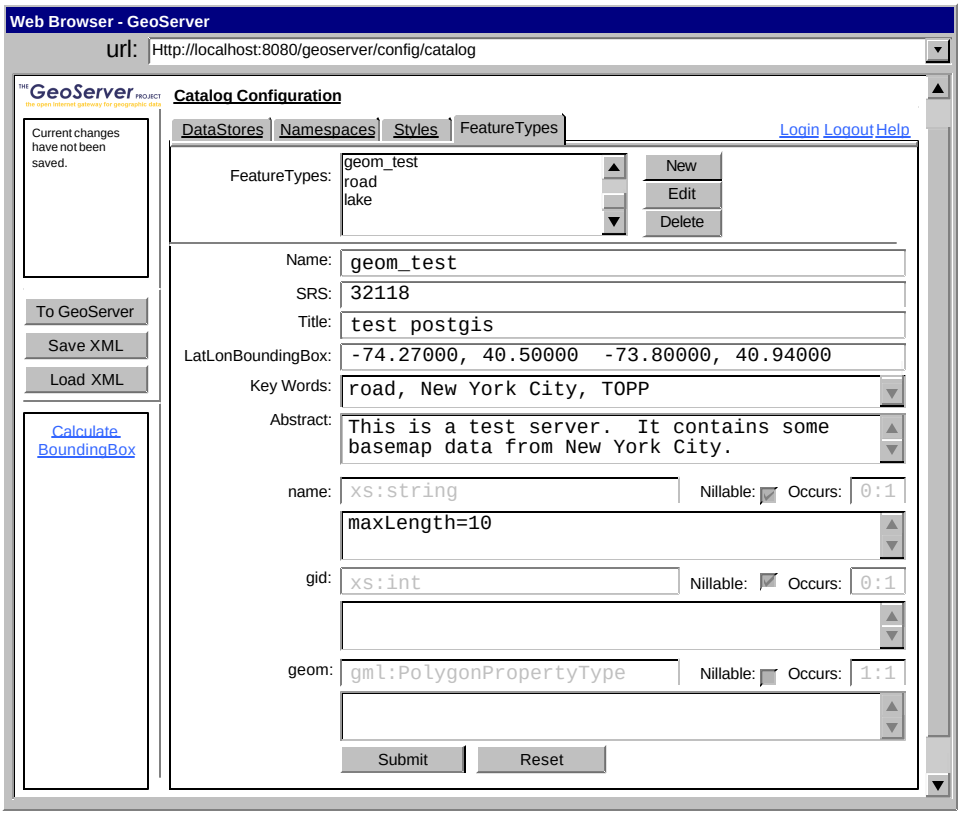

Figure 14 — Catalog Configuration Feature Types

FeatureType Schema configuration is not completely defined using widgets in our initial user interface. This is due to the complexity and open-ended nature of the XMLSchema specification used to describe FeatureTypes.

A text area has been provided to allow advanced users the opportunity to make full use of the XMLSchema. This approach does not limit the power of advanced users, and may be safely ignored by those new to GeoServer.

If any text is provided in the text area, it is assumed to be an extension of the named type generated from the GeoTools2 schema information.

From the example above:

```
<xs:element name="geom_test.name" nillable="true" minOccurs="0" maxOccurs="1">
   <xs:simpleType>
      <xs:restriction base="xs:string">
         <xs:maxLength value="10"/>
      </xs:restriction>
   </xs:simpleType>
</xs:element>
```