Help! My GeoServer Doesn't Like My App-Schema!

An in depth look at a GeoServer 2.2 workspace that utilizes the app-schema extension

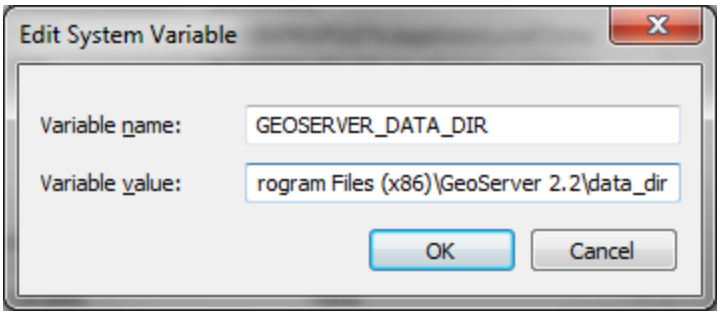
The purpose of this post is to break down the parts of a GeoServer 2.2 workspace and show how an app-schema fits into that workspace. All code in its entirety is available at https://github.com/jpTipton/gserv-app-schema-workspace. This inludes 5 workspaces: aasg1, aasg2, aasg3, aasg4, and gsmlp. The aasg workspaces include the Hypocenter, ThermalConductivity, wellHeader, and WellLog examples respectively. The gsmlp workspace includes the GeologicUnitView and ContactView examples. The code also includes a PostgreSQL database dump that includes all of the data used by these app-schemas. I encourage you to reference the full code examples as you read this over. If you are interested in running the examples, the readme explains how to get everything setup, but it is just about as easy as drag and drop.

Dependencies:
GeoServer
GeoServer app-schema extension
PostgreSQL
PostGIS

My Environment:
Windows 7 (Also tested on Ubuntu 32 bit 12.04 Desktop)
Geoserver 2.2
PostgreSQL 9.1

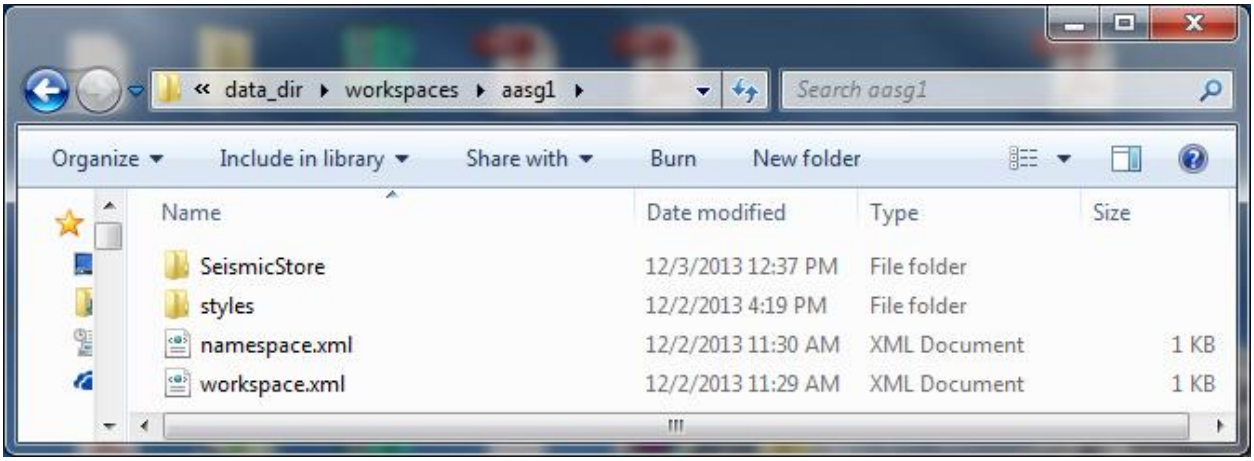
*After writing this, I tried installing it on Ubuntu. I ran into 2 major problems: 1) It appeared to not work with OpenJDK. You need to uninstall and use Oracle Java. 2) This will not work with GeoServer 2.4, I'm not sure what all versions this workspace works with, but if you are following along at home, use GeoServer 2.2! I will try and upgrade the workspace in the near future.

I'm working in Windows, but the same concepts will apply for Linux systems as well. 1st, you need to know where your workspace directory is. For Windows the default will be to "C:\Program Files (x86)\GeoServer 2.x\data_dir\workspaces", but you can change the "GEOSERVER_DATA_DIR" Environment Variable to any path (not necessary, but you can).



Workspaces – (aasg1)

Each Workspace in Geoserver has 2 files: “workspace.xml” and “namespace.xml”. These files along with the workspace folder are created when you create a new workspace. Here is the look at my “aasg1” workspace.



Workspace.xml

The workspace.xml just holds the prefix “aasg1” and the id that I changed to “WorkspaceInfoImpl-aasg1” so that I could keep it straight. If you change this id, make sure you also change it in the “datastore.xml” document as well.

Namespace.xml

The “namespace.xml” document holds the same prefix “aasg1” along with a uri, in this case http://stategeothermaldata.org/uri-gin/aasg/xmlschema/hypocenter/1.7. Note: this namespace does not necessarily have to be a real webpage. Actually, this example happens to be one that does not, as of writing, resolve to a real page.

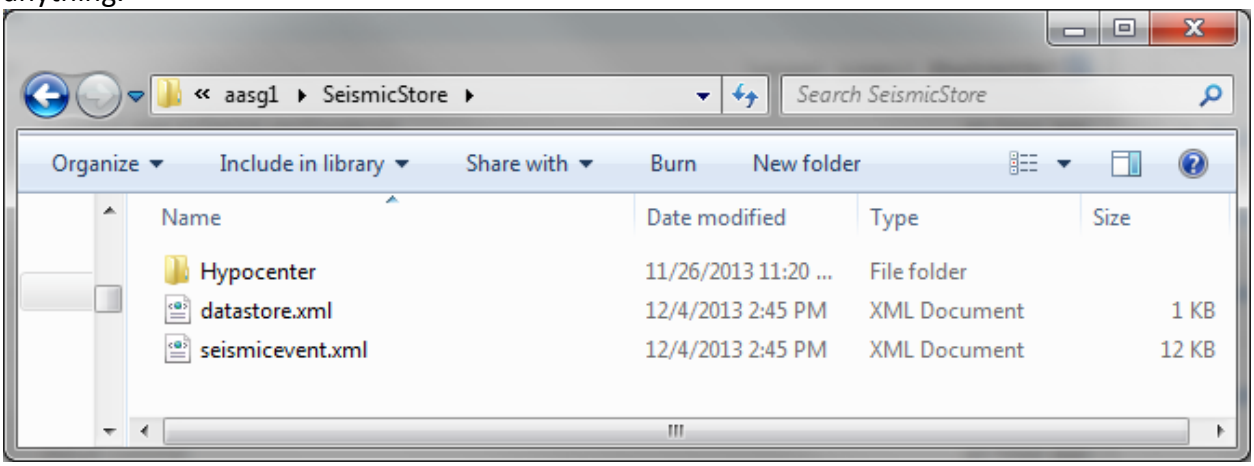
There are 2 things that the namespace has to be 1) unique to that workspace and 2) match the Target Namespace from the XSD. If you’re asking where do I find that information, go to http://schemas.usgin.org/models/ and find the SLD to download for your content model and

version. Our example will be the Seismic Event Hypocenter Version 1.7. Open the XSD in Visual Studio (VS) or Notepad++ and you will find `targetNamespace=http://stategeothermaldata.org/uri-gin/aasg/xmlschema/hypocenter/1.7` at the top where all of the namespaces are being declared. You will also notice that they declare `xmlns:aasg=http://stategeothermaldata.org/uri-gin/aasg/xmlschema/hypocenter/1.7`. So they are using the prefix "aasg". I am using "aasg1". I could have used "mnop". It doesn't matter if our prefixes match, it only matters that the target namespace URI = our namespace URI. *Also the reason that I didn't use "aasg" is that you have to create a new workspace and namespace for each layer because each layer's XSD will call for a different URI namespace, even though all of the XSDs use the same "aasg" prefix. Therefore I could only use "aasg" for 1 layer and would have to use others for the other layers.

One last thing on Namespaces, if you change the namespace `<id>` to something more readable such as `<id>NamespaceInfoImpl-aasg1</id>`, change it where the "featuretype.xml" document refers to the `<id>` as well.

Datastore Folder – (aasg1/SeismicStore)

The next folder is the datastore "SeismicStore". You should probably name this folder the same as the `<name>`element in your "datastore.xml". This folder holds 2 files: the previously mentioned "datastore.xml" and the mappingFile.xml ("seismicevent.xml") that can be named anything.



datastore.xml

Here is the entire datastore.xml:

```
<dataStore>
  <id>DataStoreInfoImpl--SeismicEvent002</id>
  <name>SeismicStore</name>
  <description>SeismicStore</description>
  <enabled>true</enabled>
  <workspace>
    <id>WorkspaceInfoImpl-aasg1</id>
  </workspace>
```

```
  <connectionParameters>
    <entry key="dbtype">app-schema</entry>
    <entry key="url">file:workspaces/aasg1/SeismicStore/seismicevent.xml</entry>
    <entry key="namespace">http://stategeothermaldata.org/uri-
gin/aasg/xmlschema/hypocenter/1.7</entry>
  </connectionParameters>
  <__default>false</__default>
</dataStore>
```

The `<name>` element should match the name of the Datastore folder (“SeismicStore”). The connection parameters are just telling Geoserver to look to the mappingFile.xml (“seismicevent.xml”) document for more details. `<entry key="namespace">http://stategeothermaldata.org/uri-gin/aasg/xmlschema/hypocenter/1.7</entry>` needs to be the namespace URI.

mappingFile.xml (SeismicEvent.xml)

The mappingFile.xml (seismicevent.xml) file can get really long. Some of the most important things are to include your namespaces at the top:

```
<namespaces>
                <Namespace>
                        <prefix>aasg1</prefix>
                        <uri>http://stategeothermaldata.org/uri-
gin/aasg/xmlschema/hypocenter/1.7</uri>
                </Namespace>
</namespaces>
```

This should match your workspace prefix and URI. The next section sets up your datastore connection. In this case postgis type on the localhost with the database named postgis.

```
<sourceDataStores>
                 <DataStore>
                        <id>SeismicEventStore</id>
                        <parameters>
                                <Parameter>
                                        <name>dbtype</name>
                                        <value>postgisng</value>
                                </Parameter>
                                <Parameter>
                                        <name>host</name>
                                        <value>localhost</value>
                                </Parameter>
                                <Parameter>
                                        <name>port</name>
                                        <value>5432</value>
                                </Parameter>
                                <Parameter>
                                        <name>database</name>
                                        <value>postgis</value>
                                </Parameter>
                                <Parameter>
```

```
                        <name>schema</name>
                        <value>public</value>
                    </Parameter>
                    <Parameter>
                        <name>user</name>
                        <value>gserv</value>
                    </Parameter>
                    <Parameter>
                        <name>passwd</name>
                        <value>YourPasswd</value>
                    </Parameter>
                <Parameter>
                  <name>Expose primary keys</name>
                  <value>true</value>
                </Parameter>
              </parameters>
          </DataStore>
    </sourceDataStores>
```

If your primary key is a field that you want the public to see and especially if it maps to a required field, make sure and include:

```
            <Parameter>
               <name>Expose primary keys</name>
               <value>true</value>
             </Parameter>
```

Next, you add in the XSD document that we got earlier from the http://schemas.usgin.org/models/ site:

```
<schemaUri>http://schemas.usgin.org/files/seismic-event-
hypocenter/1.7/SeismicHypocenters.xsd</schemaUri>
```

The next items to take care of are:

```
<sourceDataStore>SeismicEventStore</sourceDataStore>
<sourceType>seismiceventhypocenter</sourceType>
<targetElement>aasg1:Hypocenter</targetElement>
```

`<sourceDataStore>` element is name of the datastore that we just configured.

*Anything referring to source always refers to our real local resources (datastore, shapefile, PostGIS table, etc. . .)

`<sourceType>` is the name of the PostGIS table (or other source) where the data is coming from. This is the only time you will refer to it. *If you are using PostGIS ALWAYS USE LOWERCASE. You will run into trouble fast if you use capital letters. If your original database, tables, or fields in PostGIS have capital letters, STOP what you are doing now and change them to all lowercase.

`<targetElement>` is your workspace prefix followed by the element name described by the XSD document:

```
<xs:element name="Hypocenter" type="aasg:HypocenterType"
substitutionGroup="gml:_Feature" />
```

*Anything that refers to “target” will be referring to the XSD definitions.

Next is the attribute mapping. Let’s take a look at just one example:

```
<AttributeMapping>
          <targetAttribute>aasg1:HypocenterURI</targetAttribute>
          <sourceExpression>
            <OCQL>hypocenteruri</OCQL>
          </sourceExpression>
        </AttributeMapping>
```

`<targetAttribute>` is the workspacePrefix:FieldName. Make sure that you spell them exactly correctly with lower and upper case letters appropriately.

`<sourceExpression><OCQL>` is simply the field name from your source data. (It’s the field from your PostGIS table) Again, don’t capitalize. Use all lowercase!

Creating an Attribute Map

I’ve found the easiest way to create this attribute map is to copy the field list from the premade excel template into an empty excel sheet. Use the formula =lower(A1) in cell B1. A1 will hold your `<targetAttribute>` values (without the prefix). B1 will hold your `<sourceExpression>` values.

	A	B
1	FieldOne	fieldone
2	FieldTwo	fieldtwo
3	FieldThree	fieldthree

Then copy and paste that into Notepad++ and Find+Replace the following (Be sure to change the Search Mode from Normal to Extended in the Find+Replace menu):

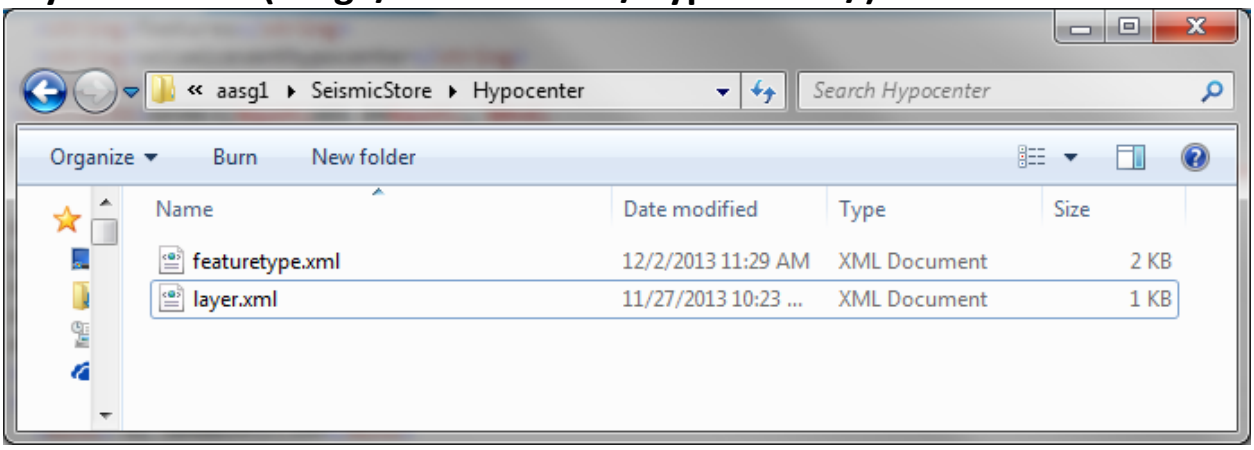
Find what: `\r\n`
Replace with:
```
</OCQL>\r\n</sourceExpression>\r\n</AttributeMapping>\r\n<AttributeMapping>\r\n
<targetAttribute>aasg2:
```

Find what: `\t`
Replace with: `</targetAttribute>\r\n\t<sourceExpression>\r\n\t\t<OCQL>`

Optional Formatting:
Find what: __ -Actually use 2 spaces “ “ without the quotes here
Replace with: \t

You will have to clean up the 1st and last fields, but these rules will add the extra formatting and change your list into a properly formatted xml map. Just place the whole thing between the `<FeatureTypeMapping></FeatureTypeMapping>` and you're done with that file and the datastore folder.

Layer Folder – (aasg1/SeismicStore/Hypocenter/)



Next, is the layer which we have named "Hypocenter". I had a little trouble when I moved the workspaces to my server. I found that the layer folder's name should be the same as the `<name>` element in the layer.xml file as well as the same as the `<targetElement>` from the mapping file ("seismicevent.xml") which is also the name of our layer from the XSD document. Without doing this, GeoServer had trouble finding the file, even though it was fine on my local test machine.

layer.xml

The only thing you need to do here is make sure the `<name>` is the same as the layer name from the XSD like I just mentioned. You also need a unique `<id>` and featuretype `<id>` elements that can be anything, they just need to be globally unique. Also the featuretype `<id>` must be the same where referenced in the "featuretype.xml"

featuretype.xml

The `<name>` and `<nativeName>` need to be the exact same as the layer name specified from the XSD which we have already established in this case is "Hypocenter". All of the other information is data about the layer and the extent. The easiest way to fill all of this out is to publish the data 1st like normal as a regular PostGIS layer and copy everything over and just make sure the `<name>,<nativeName>,<namespace><id>,<featureType><id>,` and `<store class="dataStore"><id>` are all consistent with what we've used before.

Styles Folder

I almost forgot the workspace/styles folder. I skipped it because we need to first get the data to load before we style it. These files don't have to be located here. It could be located in your normal styles folder, but I placed them here so that the styles would be with their workspaces.

Style.xml – (EventHypocenter.xml)

Make sure the `<workspace><id>` is correct and set the `<filename>` to the filename of your .sld file (ex: EventHpocenter.sld)

Style.sld – (EventHypocenter.sld)

This is where your style is. If you are using 1 style for the entire layer, you don't need to do anything. If you are styling based on attributes, you need to make 1 little tweak. Add the namespace and prefix to the `<sld:StyledLayerDescriptor>` element. Here is an example:

```
<sld:StyledLayerDescriptor xmlns:aasg=http://stategeothermaldata.org/uri-gin/aasg/xmlschema/hypocenter/1.7 xmlns:gml=http://www.opengis.net/gml etc. . . >
```

*The prefix "aasg" does not have to match anything. Our prefix here is "aasg" our workspace prefix is "aasg1" and our XSD namespace prefix is "aasg". None of this matters as long as the URI (`http://stategeothermaldata.org/uri-gin/aasg/xmlschema/hypocenter/1.7`) matches

Now add your prefix to all of the `<ogc:PropertyName>` elements. For example, my datatable has the field "magnitude" that we mapped to "aasg1:Magnitude" in our mapping file. Our entry would be this: `<ogc:PropertyName>aasg:Magnitude</ogc:PropertyName>` . Again, not to confuse; aasg1 means nothing to this SLD because it hasn't been defined here. I defined "aasg" just to prove a point that the prefix both needs to be defined and does not necessarily have to match the namespace/workspace prefix.

Congratulations! It's that easy! It practically does itself, right?

Here's some rules to remember:

1) Make a new workspace for each layer that requires a different namespace URI
2) Workspace namespace URI must = targetNamespace from the XSD but the prefix does not need to match (Ex: aasg1 <> aasg)
3) Style.sld must declare the namespace URI. Same rule as workspace namespace. Prefixes need not match, only URI. In fact, style prefix doesn't even need to be the same as the workspace/namespace prefix.
4) datastore.xml `<name>` = datastore folder name
5) Mapping-file.xml `<sourceType>` = PostGIS table name
6) Mapping-file.xml `<targetElement>` = namespacePrefix:XSDLayerName
7) Layer.xml `<name>` = `<nativeName>` = XSD layer name

8) Make sure all Workspace, Namespace, Datastore, Layer, Feature, and Style `<id>` elements are unique to that Workspace, Namespace, Datastore, Layer Feature, and Style. Also, make sure they are consistent everywhere they are referenced in the workspace.

After you make the 1st workspace, I found it the easiest to just copy and paste the entire workspace directory and then Find+Replace all of the Workspace, Namespace, Datastore, Layer, Feature, and Style `<id>` and `<name>` elements. Then make sure all of the previous rules are followed and make your new attribute map. Finally, set the correct extent in the featuretype.xml.

Other helpful links:
http://lab.usgin.org/node/117
https://github.com/ccaudill/GeoserverWorkspace
http://lab.usgin.org/groups/best-practices-usgin-web-service-hosting/geoserver-app-schema-set-progress