Parameter Widget Examples

This document is a guide to the various types of parameter widgets available, and the syntax for creating them.

String types

Parameters that have a string value should be of the phylum `string`

Default values

Default values for strings can be specified by either an attribute:

```
default="hello"
```

or for selection (see below), by specifying `default="true"` for a specific value:

```
<value default="true">choice 1</value>
```

Freeform strings

Freeform strings do not need to specify any other specific format or type information.

```
<parameter phylum="string" cardinality="single" key="result"
             friendly-name="Result Name" />
```

String selection

To select a particular string from a list, the format should be specified as `select`, and a set of values should be included.

```
<parameter phylum="string" format="select"
             cardinality="single" key="test" friendly-name="Test" >
      <values>
         <value>choice1</value>
         <value>choice2</value>
         <value>choice3</value>
      </values>
</parameter>
```

Extension point string selection

To select an extension point to be returned as a string (selected by name, value returned by tag), use format `extension-point`, and an extension element must be specified:

```
<parameter phylum="string" format="extension-point" cardinality="single" key="extpt" friendly-name="Ext pt">
        <extension point="ncsa.gis.ui.initFileActions"/>
</parameter>
```

Multiple String selection

String lists can be defined using cardinality="multiple"
Choices can be set in the same method as string selection.

```
<parameter group="Required" phylum="string" format="select" cardinality="multiple" key="list" friendly-name="
Field Select">
    <dependent-param>bridgeDamage</dependent-param>
    <values source="parameter" sourcekey="bridgeDamage" sourcetype="fields"/>
</parameter>
```

Fields from a dataset string selection

You can also use the string selection to choose between fields/columns of a dataset:

```
       <parameter phylum="string" format="select" cardinality="single" key="dislocationZoneFieldId" friendly-
name="Field to use as preferred zone Id" default="tract">
               <dependent-param>buildingPopulationDislocation</dependent-param>
               <values source="parameter" sourcekey="buildingPopulationDislocation" sourcetype="fields" />
       </parameter>
```

The dependent-param element indicates which parameter will hold the dataset that you want to choose fields from. The value element must also specify that parameter in the `sourcekey` attribute. Setting `sourcetype` to `fields` indicates that you want to use the fields from the dataset.

String maps

Any sort of data that, at it's lowest level, is a map of Strings to Strings should use the `stringmap` phylum.

Keys based on a separate dataset parameter

To create a string map widget that has keys based off of certain records or fields from a separate dataset parameter, use format `parameter-keys`.

Also, the dependent-param element must be specified to indicate a field that this field is dependent on.

Keys based on fields of a dataset, values one of a set of hardcoded choices.

```
<parameter  phylum="dataset" format="shapefile"
            cardinality="single" key="bridges" friendly-name="Bridges">

<parameter phylum="stringmap" format="parameter-keys" cardinality="single" key="map" friendly-name="Agg Type">
   <dependant-param>bridges</dependant-param>
   <keys source="parameter" sourcekey="bridges" sourcetype="fields"/>
   <values source="values">
      <value default="true">min</value>
      <value>max</value>
      <value>sum</value>
   </values>
</parameter>
```

Boolean values

All boolean values should be specific by phylum `boolean`.

Checkboxes

Checkboxes are the default widget for boolean values.

```
<parameter phylum="boolean" cardinality="single" key="choice"
             friendly-name="Check Me"   />
```

Dataset values

Parameters should use the phylum `dataset` for dataset values.

Specified by schema type

To select a dataset specified by one or more schema type, you can omit a format,
and just specify the types desired.

```
<parameter  phylum="dataset"
            cardinality="single" key="bridges" friendly-name="Bridges">
     <types>
        <type>bridge</type>
     </types>
   </parameter>
```

By format of dataset (feature dataset, fragility dataset, etc)

In this case, `types` should be omitted, and the `format` should be specified.
The format should match the string returned by `getDatasetFormat()` in the various dataset types.

Important note: Feature datasets use the key `shapefile`, whether or not they are backed by a shapefile.

```
<parameter phylum="dataset" format="mapping"
            cardinality="single" key="mapping" friendly-name="Fragility Mapping">
   </parameter>
```

By feature dataset's geometry type

In this case, `types` should be omitted, and the `format` should be specified as `shapefile`.
Instead of `types`, `geometry-types` should be specified.

```
<parameter  phylum="dataset" format="shapefile"
            cardinality="single" key="polygon" friendly-name="Polygon ">
   <geometry-types>
      <geometry-type>Polygon</geometry-type>
   </geometry-types>
</parameter>
```

Multiple datasets

Any of these dataset types can be set to use multiple datasets by setting cardinality to `multiple`.

```
<parameter cardinality="multiple"
             phylum="dataset" key="hazard" friendly-name="Hazard">
   <types>
      <type>hazardRaster</type>
      <type>deterministicHazardRaster</type>
   </types>
</parameter>
```

View Options

Groups

Parameters can be combined into related groups by adding a `group` attribute.

```
<parameter group="Extra Info" phylum="string" key="myExtraParam" friendly-name="Extra stuff"/>
```

Hiding groups

Groups can be set to only appear when a certain field has a certain value:

```
<group-show-when group="Socioeconomic Disruption Metrics" node="objectiveOne" value="Include"/>
```

Where `node` is the parameter to depend on, and value is the value of the field that indicates this parameter should be shown. You can also be shown on every value EXCEPT a specific value, by adding a ! in front:

```
value="!Not Used"
```

Hiding parameters based on selection

Any parameter can by dynamically hidden or shown based on the selection of another parameter. Just add the `show-when` tags to the parameter definition:

```
<parameter group="Optimization Method Parameters" phylum="string" key="dmRetrofitMethod" friendly-name="Direct
Method Retrofit Method">
        <show-when parameter='optimizationMethod' value='Direct Method'/>
</parameter>
```

This will show the dmRetrofitMethod parameter only when the parameter `optimizationMethod` has been set to `Direct Method`

Flexible map-based parameters

To enable more flexibility where tasks don't have to have getters and setters for each parameter, you can specify that a parameter be added "as a map" using `as-map="true"`. This will add it to a special parameter called `parameterInputMap`, which is a map of the keys of any parameters specified as as-map, mapped to their values. This way tasks can have one setter (`setParameterInputMap(Map<String,Object>)`, which is defined in SimpleFeatureTask for convenience), and any number of previously unspecified parameters can be added to this.

Example:

```
<parameter group="Liquefaction" as-map="true" phylum="dataset" key="liquefaction.groundwatermap" friendly-name="
Groundwater map" >
        <types>
                <type>groundWaterDepthMap</type>
        </types>
</parameter>
```