Document Filters

Filters provide a mechanism for Technical Document Authors to associate Characteristics (metadata) with Document Elements and use these characteristics to automatically filter the content. This is useful when technical documents can serve multiple, but related, purposes. For example, a technical document that describes several similar models of the same part. Content that is specific to any model (or models) can have filters associated with it. When an end user wishes to see only the content that is either associated with all models or is particular to a specific model, they can identify the appropriate filters and the Technical Document Editor will display the correct content accordingly.

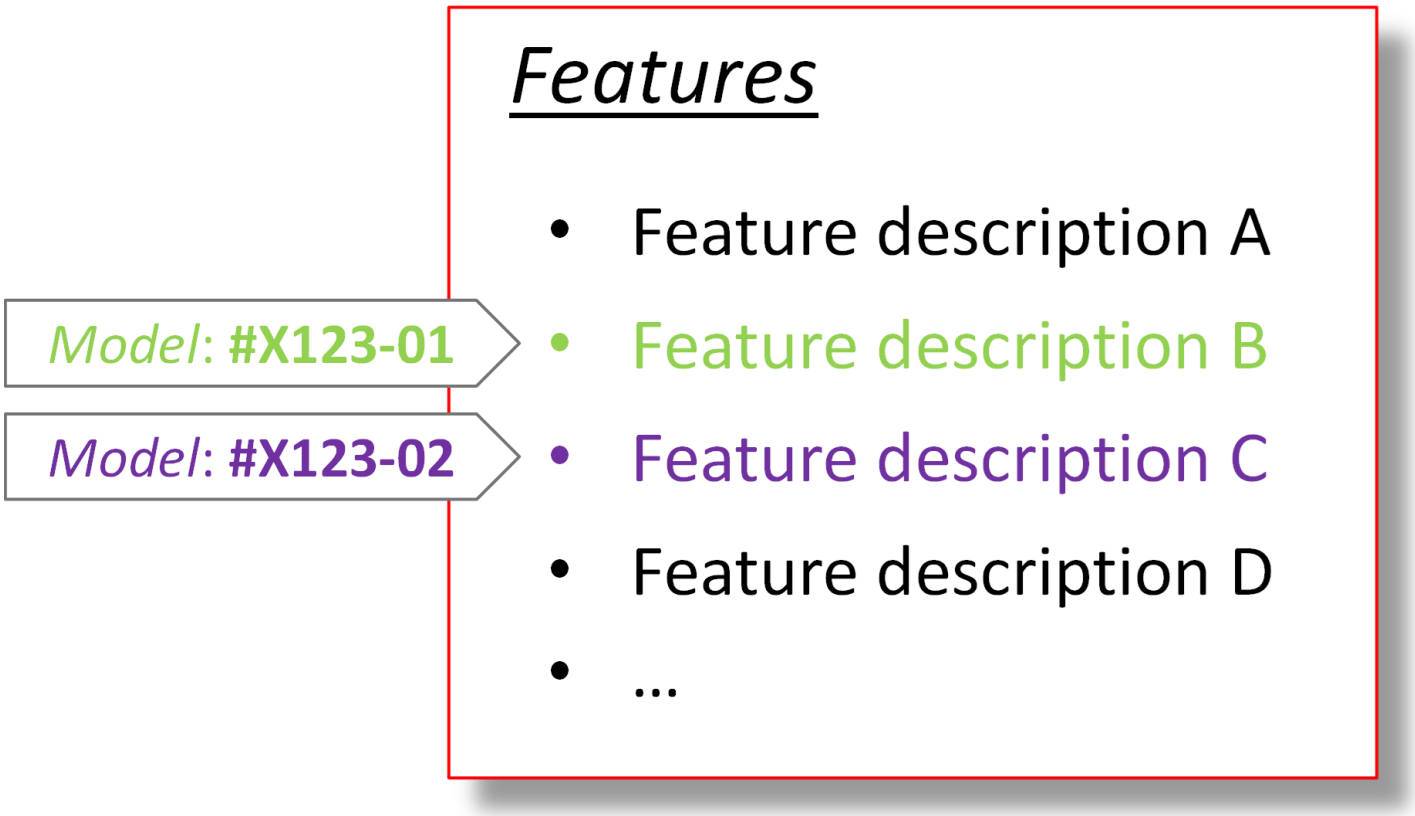

Figure: Filter Example

Referring to the figure above, the list of features associated with Model 'X123-0***1***' is A, B, and D. The list of features associated with Model 'X123-0***2***' is A, C, and D. The list of available *Filter Names* and associated *Filter Values* is set in the Method `tp_GetOptionFamilies` .

Creating the Filter List

The Method `tp_GetOptionFamilies` determines the set of Filters (Options) by returning a JSON-formatted string containing a name/value list where the *name* specifies the Name of the Filter/Option, and the *value* specifies the list of valid values. The default implementation for this Method is as follows:

```
/*
var optionFamilies = {
"Region": [ "Africa", "Asia", "Central America", "Eastern Europe", "European Union", "Middle East", "Nort
"Zone": ["A", "B", "C"],
"Model": ["X", "Y", "C"]
};
*/
Dictionary<string, List<string>> optionFamilies = new Dictionary<string, List<string>> ();
optionFamilies.Add("Region", new List<String>() {  "Africa", "Asia", "Central America", "Eastern Europe",
optionFamilies.Add("Zone", new List<String>() { "A", "B", "C" } );
optionFamilies.Add("Model", new List<String>() { "X", "Y", "Z" } );
String json = new System.Web.Script.Serialization.JavaScriptSerializer().Serialize(optionFamilies);
return this.getInnovator().newResult(json);
```

The Method uses a `Dictionary` class to construct the data for the returned JSON data structure. It is possible to return a JSON-formatted string as well using the example pattern described by the comment at the top of the method code. Implementations can use this example Method as a guide for constructing static Filters, stored the list data in custom Items and retrieve the values in this Method, and/or apply whatever custom logic is necessary to construct a dynamic set of data.