

Use of FilterCapabilities

Motivation:	Make use of FilterCapabilitiesImpl
Contact:	Jody Garnett
Tracker:	http://jira.codehaus.org/browse/GEOT-1637
Tagline:	what filters are available?

This page represents the **current** plan; for discussion please check the tracker link above.

Motivation

Some parts of migrating to opengis Filter are done; this is one aspect of the move that has not yet been addressed.

- opengis **FilterCapabilities** does not have any knowledge of **Filter** interfaces; it is slightly higher level in places (reporting back concepts like hasSimpleArithmetic) and slightly lower level in others (reporting back what kind of geometry operands are supported)
- Our SQL generation code uses a deprecated **FilterCapabilities** data structure built around a mask of deprecated **FilterTypes** constants.
- We have an **FilterCapabilitiesImpl** for the opengis **FilterCapabilities** data structure; but it is not used yet
- There is no sane way to map between the names used by **FilterCapabilities** Operation entries (like "NullCheck") and GeoAPI Filter interfaces such as **PropertyIsNull**)
- opengis **Function** does not report on the number of parameters; that is the responsibility of the filter capabilities data structure **FunctionName**. Code is currently downcasting to the deprecated **FunctionExpression** in order to get at this information.
- opengis **FilterCapabilities** tracks function name and argument count; but gives no indication what the arguments are for.

For reference:

FilterCapabilities	GeoAPI Javadocs
filterCapabilitiesExample.xml	Example filter capabilities XML file

Description

The proposal is to have a **Capabilities** wrapper / builder around a opengis **FilterCapabilities** data structure. This proposal only covers creating the utility class; hooking this up to FilterToSQL is another matter.

The idea is to easily switch back and forth between operator names and Filter interface classes (since the mapping is too scary for anyone to remember). As long as we need this class we can make the methods line up with the old **FilterCapabilities** class (to help people migrate).



```
Capabilities capabilities = new
Capabilities( filterCapabilities );
capabilities.addType( Beyond.class ); // add
to SpatialCapabilities
capabilities.addType(
PropertyIsEqualTo.class ); // add to
ScalarCapabilities
capabilities.addName( "NullCheck" ); // will
enable PropertyIsNull use
capabilities.addName( "MUL" ); // will
enable hasSimpleArithmetic
capabilities.addName( "random" ); // a
function returning a random number
capabilities.addName( "Length", 1 ); //
single argument function
capabilities.addName( "toDegrees", "radians"
); // single argument function
capabilities.addName( "length", "expression"
);

if( capabilities.fullySupports( filter ) ){
    // encode as SQL
}

Capabilities capabilities2 = new
Capabilities();

capabilities2.addAll( capabilities );
capabilities2.addAll( Capabilities.LOGICAL
); // Capabilities.LOGICAL is a
FilterCapabilities
```

OUT OF SCOPE

Above we are defining some methods and classes constants to match the old `FilterCapabilities` class, we could go even further make everything method and field compatible but that would waste our time:

```
capabilities2.addType( Capabilities.LIKE );  
// Capabiities.LIKE is now  
PropertyIsLike.class
```

Status

This proposal has been accepted as the voting period has now ended.

- [Andrea Aime](#) +0
- [Ian Turton](#) +0
- [Justin Deoliveira](#) +0
- [Jody Garnett](#) 0 (as a consequence of starting without the voting process)
- [Martin Desruisseaux](#) +0
- [Simone Giannecchini](#) +0

Tasks

	no progress	✓	done	✗	impeded	⚠	lack mandate /funds/time	?	volunteer needed
--	-------------	---	------	---	---------	---	--------------------------	---	------------------

1. ✓ Write a Capabilities wrapper around FilterCapabilities data structure to perform tests in a method compatible way with the old `FilterCapabilities` class
2. ✓ Make FilterCapabilitiesImpl data structure mutable so Capabilities wrapper can build it up as needed
3. ✓ Add static NAME fields to the GeoAPI interfaces we are trying to map between; even if just to minimize cut and paste mistakes
4. ✓ Change FilterCapabilities data structure to use Collection (so we can use a Set behind the scenes)
5. ✓ Change Operator interfaces to implement equals and hashCode based on getName()
6. Add FunctionFinder.getFilterCapabilities()
7. ✓ Add Function.getArgumentNames() to GeoAPI
8. Migrate **FilterToSQLSDE** to use new data structure; as an example
9. Create Capabilities and FilterCapabilities user documentation
10. Suffer a performance review from Andrea - and probably introduce HashMaps rather than HashSets to make lookup faster

API Changes

Create a Capabilities wrapper around FilterCapabilities capturing the mapping from Filter interfaces to Operation.getName().

Where possible make this method compatible with the old `FilterCapabilities` class.

```

import
org.opengis.filter.capabilities.FilterCapabi
lties;
public class Capabilities {
    public Capabilities( FilterCapabilities
contents );
    FilterCapabilitiesImpl getContents();

    addType(Class);
    supports(Filter);
    fullySupports(Filter);
    fullySupports(Expression);
    addAll( Capabilities );
    addAll( FilterCapabilities );

    .. add additional methods as needed..

    addName(String);
    addName(String, int);
    addName(String, String...);
    toOperationName(Filter);
    toOperationName(Class);
}

```

Add NAME constants to non abstract GeoAPI Filer interfaces

BEFORE:

```
@XmlElement("PropertyIsNull")
public interface PropertyIsNull extends
Filter {
    Expression getExpression();
}
```

AFTER:

```
@XmlElement("PropertyIsNull") // XML Element
from filter.xsd
public interface PropertyIsNull extends
Filter {
    /** Operator name used to check
FilterCapabilities */
    public static String NAME = "NullCheck";
// XML Element from capabilities.xsd
    Expression getExpression();
}
```

[Adrian Custer](#) asked about the difference here:

- "PropertyIsNull" is from filter.xsd from which we get our Filter interfaces
- "NullCheck" is from filterCapabilities.xsd from which we get our Operators (describing what filters are allowed).

Ask Operator to be a data object

The interface **Operator** is part of the filterCapabilities.xsd based data structures.

BEFORE:

```
public interface Operator {
    String getName();
}
```

AFTER:

```
public interface Operator {
    String getName();
    public boolean equals(Object obj);
    public int hashCode();
}
```

Make FilterCapabilitiesImpl mutable

Change the FilterCapabilitiesImpl classes to be mutable, with both a copy constructor and an addAll method for use when combining several FilterCapabilities data structures into one.

BEFORE:

```
public class FilterCapabilitiesImpl
implements FilterCapabilities {
    public FilterCapabilitiesImpl();
    FilterCapabilitiesImpl( String,
ScalarCapabilities, SpatialCapabilities,
IdCapabilities);

    getVersion();
    IdCapabilitiesImpl getIdCapabilities();
    ScalarCapabilitiesImpl
getScalarCapabilities();
    SpatialCapabilitiesImpl
getSpatialCapabilities()
}
```

AFTER:

```

public class FilterCapabilitiesImpl
implements FilterCapabilities {
    public FilterCapabilitiesImpl();
    FilterCapabilitiesImpl( String,
ScalarCapabilities, SpatialCapabilities,
IdCapabilities);
    FilterCapabilitiesImpl(
FilterCapabilities );

    setVersion(String);
    getVersion();

    setId(IdCapabilities );
    IdCapabilitiesImpl getIdCapabilities();

    setScalar(ScalarCapabilities);
    ScalarCapabilitiesImpl
getScalarCapabilities();

    setSpatial(SpatialCapabilities)
    SpatialCapabilitiesImpl
getSpatialCapabilities()

    addAll(FilterCapabilities);
}

```

Add Optional getArgumentsNames() to FunctionName:

This is an UNRELATED CHANGE THAT WAS FUN (and will let me write a query builder).

BEFORE:

```
public interface FunctionName extends
Operator {
    int getArgumentCount();
}
```

AFTER:

```
public interface FunctionName extends
Operator {
    int getArgumentCount();
    List<String> getArgumentNames(); // may
be null if unknown
}
```

Access Functions data structure from FuntionFinder

The interface **Functions** is part of the filterCapabilities.xsd based data structures.

BEFORE:

```
class FunctionFinder {
    FunctionFinder(Hints)
    Function findFunction(String);
    Function findFunction(String, List);
}
```

AFTER:

```
class FunctionFinder {  
    FunctionFinder(Hints)  
    Function findFunction(String);  
    Function findFunction(String, List);  
    Functions getFunctions();  
}
```

Documentation Changes

- Create a Page for FilterCapabilities
- Create a Page for FilterToSQL