Function lookup using qualified Name

Contact:	full name
Tracker:	http://jira.codehaus.org/browse/GEOT-3871
Tagline:	a function by any other name

Children:

Description

We have officially reached the limits of function lookup using a simple String. This limitation is causing us a bit of pain on too fronts:

- bridging between process and function is not a direct one to one unless we can qualify the function name with a namespace
- the WFS 2.0 specification offers some syntactic sugar (known as operations) as an alternate way of using a function; this also requires a namespace.

Rather than add a namespace; we are going to recycle out **Name** class which is also used for feature types; attribute types, process and anywhere else we need a qualified name. The **Name** class is similar to Java **QName** but more lightweight and easier to hack with.

Status

This proposal is being fast tracked as it is holding up the existing Join Support proposal.

Voting has not started yet:

- Andrea Aime
- Ben Caradoc-Davies
- Christian Mueller
- Ian Turton
- Justin Deoliveira +1
- Jody Garnett +1
- Simone Giannecchini

Tasks

This section is used to make sure your proposal is complete (did you remember documentation?) and has enough paid or volunteer time lined up to be a success

	no progress		done		impeded		lack mandate/funds/time		volunteer needed

1. Add **Name** to the FunctionFactory interface as shown in BEFORE / AFTER (GEOT-3871)
2. Update the docs to show an example of looking up a function using a **Name**
3. Update the **Update** instructions showing people an example of how to update their **FunctionFactory**
4. update CQL parser to respect qualified name (example "se:interpolate")

API Changes

Filter

BEFORE:

```
interface FunctionFactory {
    List<FunctionName> getFunctionNames();
    Function function(String name, List<Expression> args, Literal fallback );
}
class FunctionFinder {
    public findFunction(String name, List<Expression>  parameters)
    public findFunction(String name, List<Expression>  parameters, Literal fallback)
    ...
}
```

A quick code example:

```
Expression expr = ff.function("buffer",ff.property("geom"), ff.literal(0.3));
Expression expr2 = CQL.expression("buffer( geom, 0.3 )");
```

AFTER:

```
interface FunctionFactory {
    List<FunctionName> getFunctionNames();
    Function function(String name, List<Expression> args, Literal fallback );
    Function function(Name name, List<Expression> args, Literal fallback );
}
class FunctionFinder {
    public findFunction(String name, List<Expression>  parameters)
    public findFunction(String name, List<Expression>  parameters, Literal fallback)
    public Function findFunction(Name name, List<Expression>  parameters)
    public Function findFunction(Name name, List<Expression>  parameters, Literal
fallback)
    ...
}
```

A quick code example:

```
// qualify buffer to use the Process implementation of buffer
Expression expr = ff.function( new NameImpl("jts","buffer"), ff.property("geom"),
ff.literal(0.3), ff.literal(32), ff.literal("round") );

// unqualified name
Expression expr = ff.function( new NameImpl("buffer"), ff.property("geom"),
ff.literal(32), ff.literal("round") );

// CQL - is this possible?
Expression expr = CQL.expression("jts:buffer( GEOM, 0.3, 32, 'round'");

// backwards compatibility
Expression expr = ff.function("buffer",ff.property("geom"), ff.literal(0.3));

// factory can split on ":"
Expression expr = ff.function("jts:buffer",ff.property("geom"), ff.literal(0.3),
ff.literal(32), ff.literal("round") );
```

Documentation Changes

The following documentation pages need attention after this change:

- Function Tutorial
- Factory Introduction
- Filter (gt-opengis) Function example code need an example with Name
- Filter (gt-main) needs FunctionFactory example updated