Implementing an Output Streaming WPS process

Following this tutorial you will be able to publish a Buffer process as an Output Streaming WPS process, so that the WPS client can access intermediate results.

Prerequisites

- To follow this tutorial you need a 52°North WPS development environment. Have a look at A Primer on 52°North WPS in Eclipse Java "Indigo" or A Primer on 52°North WPS in Eclipse Java EE "Helios", depending on your preferred Eclipse version.
- Additionally, it is recommended to have a WPS client that supports streaming. As of now, the Quantum GIS WPS Client is the only one with this capability. Thus, if you want to visualize the result of the process in a neat way, consider installing such a client. Otherwise, you must read the response as XML.

Introduction

If you are not familiar with Streaming based WPS processes, please read this blog post to get an overview. Output Streaming WPS receives input data as a whole, splits it into chunks, and starts processing each chunk for publishing intermediate results. This reduces latency, i.e., time to access process results.

It is important to note that not all algorithms could benefit from Output Streaming. Particularly, algorithms that require focal or global knowledge are not supported. The following table shows some examples of algorithms supported (Raster based algorithms are in green).

Type (Knowledge)	Example	Supports Output Streaming?
1:1 (Local)	Simple Buffer, Reclassify	Yes
1:M (Local)	Explode line	Yes
M:1 (Focal)	Kernel based operations	No
M:1 (Global)	Shortest path	No
M:M (Global)	Polygon intersection, Cost distance	No

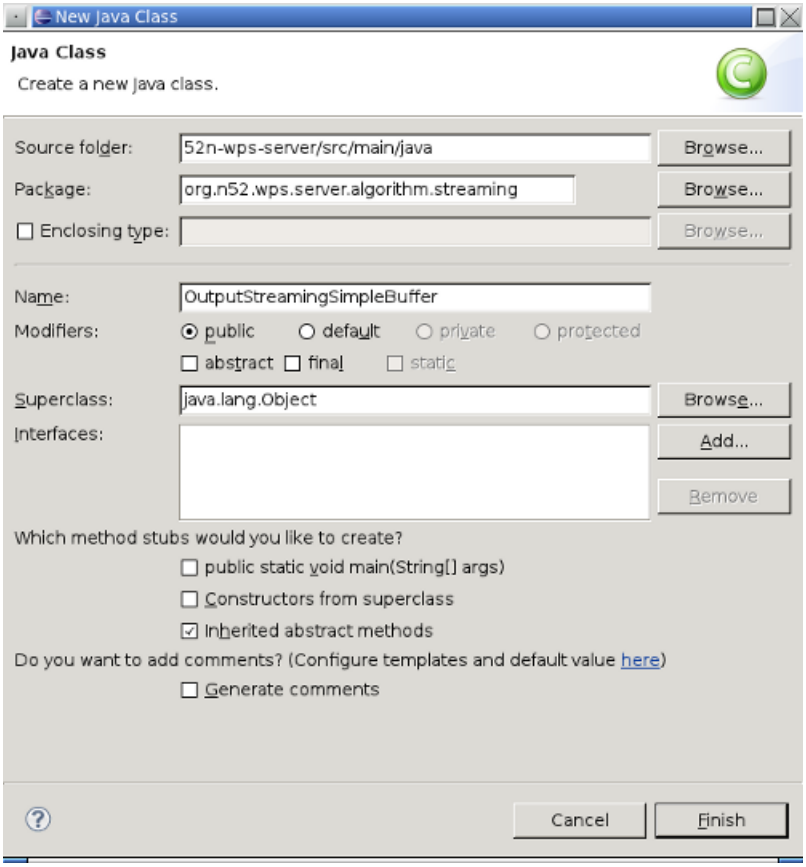
The Simple Buffer algorithm is supported because it uses local knowledge (it takes one input feature and produces an output feature), so you can continue with the tutorial steps:

1. Create a base process

Your Output Streaming WPS process will be based on a conventional WPS process. If necessary, create one following this tutorial. Fortunately, the Simple Buffer is already implemented in the 52°North WPS framework, so you can go to the step 2.

2. Creating a new class

   Create a new class in the `org.n52.wps.server.algorithm.streaming` package. You can call it `OutputStreamingSimpleBuffer.java`



3. Extend the abstract class for Output Streaming

   Make your new class extend the abstract class called `AbstractVectorOutputStreamingAlgorithm` and add the unimplemented methods, this way:

```
package org.n52.wps.server.algorithm.streaming;

public class OutputStreamingSimpleBuffer extends AbstractVectorOutputStreamingAlgorithm {

    @Override
    public String getBaseAlgorithmName() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public String getInputStreamableIdentifier() {
        // TODO Auto-generated method stub
        return null;
```

```
    }

    @Override
    public String getOutputIdentifier() {
        // TODO Auto-generated method stub
        return null;
    }

}
```

You need to overwrite those three methods as indicated in the next steps.

4. Overwrite the `getBaseAlgorithmName` method

   Make the `getBaseAlgorithmName` method return the base process' full name. In this case, the full name of the Simple Buffer process is `org.n52.wps.server.algorithm.SimpleBufferAlgorithm`. So, just write

   ```
   return "org.n52.wps.server.algorithm.SimpleBufferAlgorithm";
   ```

5. Overwrite the `getInputStreamableIdentifier` method

   Make the `getInputStreamableIdentifier` method return the name of the parameter that will be split. For this, have a look at the `SimpleBufferAlgorithm` class and look for input identifiers.

   As you can see, there are two input identifiers: `data` and `width`. As Output Streaming WPS processes always split spatial data (that's what they were created for), you must choose the identifier `data`. So, just write

   ```
   return "data";
   ```

   in the `getInputStreamableIdentifier` method of your `OutputStreamingSimpleBuffer` class.

6. Overwrite the `getOutputIdentifier` method

   Make the `getOutputIdentifier` method return the output identifier of the base process. Proceed as in the previous step to get the output identifier of the `SimpleBufferAlgorithm` class.

   Yeap, it is `result`, so write this in the `getOutputIdentifier` method:

   ```
   return "result";
   ```

   You are almost done, the name of the new process must be added to a 52°North WPS framework config file, see next step.

7. Add the process to the config file

   Open the `wps_config.xml` file in `52n-wps-app/src/main/webapp/config/` and look for the `AlgorithmRepositoryList` tag in the middle of the file. You need to add a new `Property` element to the repository called `LocalAlgorithmRepository`. The new `Property` element is:

   ```
   <Property name="Algorithm" active="true">org.n52.wps.server.algori
   ```

   After adding it, the `wps_config.xml` file must look like this:

   ```
   <Repository name="LocalAlgorithmRepository"
       className="org.n52.wps.server.LocalAlgorithmRepository" active="true">
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.SimpleBufferAlgorithm</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.coordinatetransform.CoordinateTransformAlgorithm</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.simplify.DouglasPeuckerAlgorithm</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.intersection.IntersectionAlgorithm</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.convexhull.ConvexHullAlgorithm</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.raster.AddRasterValues</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.test.DummyTestClass</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.spatialquery.IntersectsAlgorithm</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.spatialquery.TouchesAlgorithm</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.SnapPointsToLinesAlgorithm</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.raster.ndwi</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.streaming.OutputStreamingSimplifyDouglasPeucker</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.streaming.OutputStreamingNDWI</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.streaming.FullStreamingSnapPointsToLines</Property>
       <Property name="Algorithm" active="true">org.n52.wps.server.algorithm.streaming.OutputStreamingSimpleBuffer</Property>
   </Repository>
   ```

8. Compile and deploy

   Compile and deploy the application. A new process must appear once you call the `GetCapabilities` request.

Simple, isn't it?

For testing the Output Streaming WPS process you've just created, go to the 52°North WPS Test Client, copy the content of this document into the input box, and send it. You will get an XML response with the URL of a `Playlist` that is updated every time intermediate results are available.

As stated before, it is recommended to use the Quantum GIS WPS Client for visualizing incoming intermediate results. A couple of screencasts are available for both Vector and Raster processes.

You will notice that there is a new parameter compared to the base process. It is called `NumberOfChunks` and you can use it to set the number of chunks (groups of features) the process will create and append to the `Playlist`. The more chunks the smaller the intermediate results, which reduces latency.