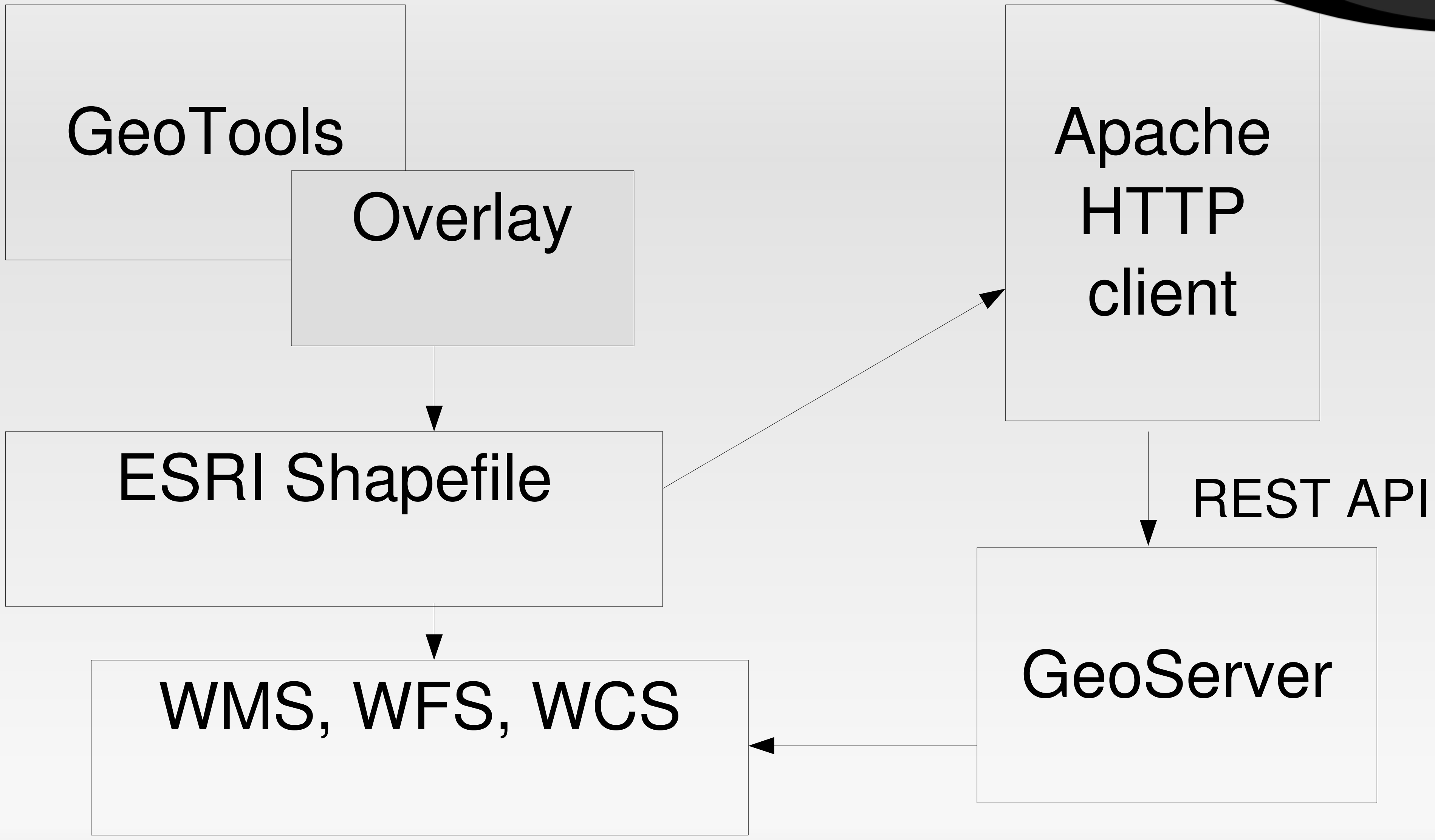
Workshop WPS

Publish output of overlay
as WMS and WFS

Principle

- Results are not send back as output
- Results are published as WMS, WFS or WCS
- Examples where to use WMS, WFS or WCS as output
  - Thousands of features
  - Large raster data

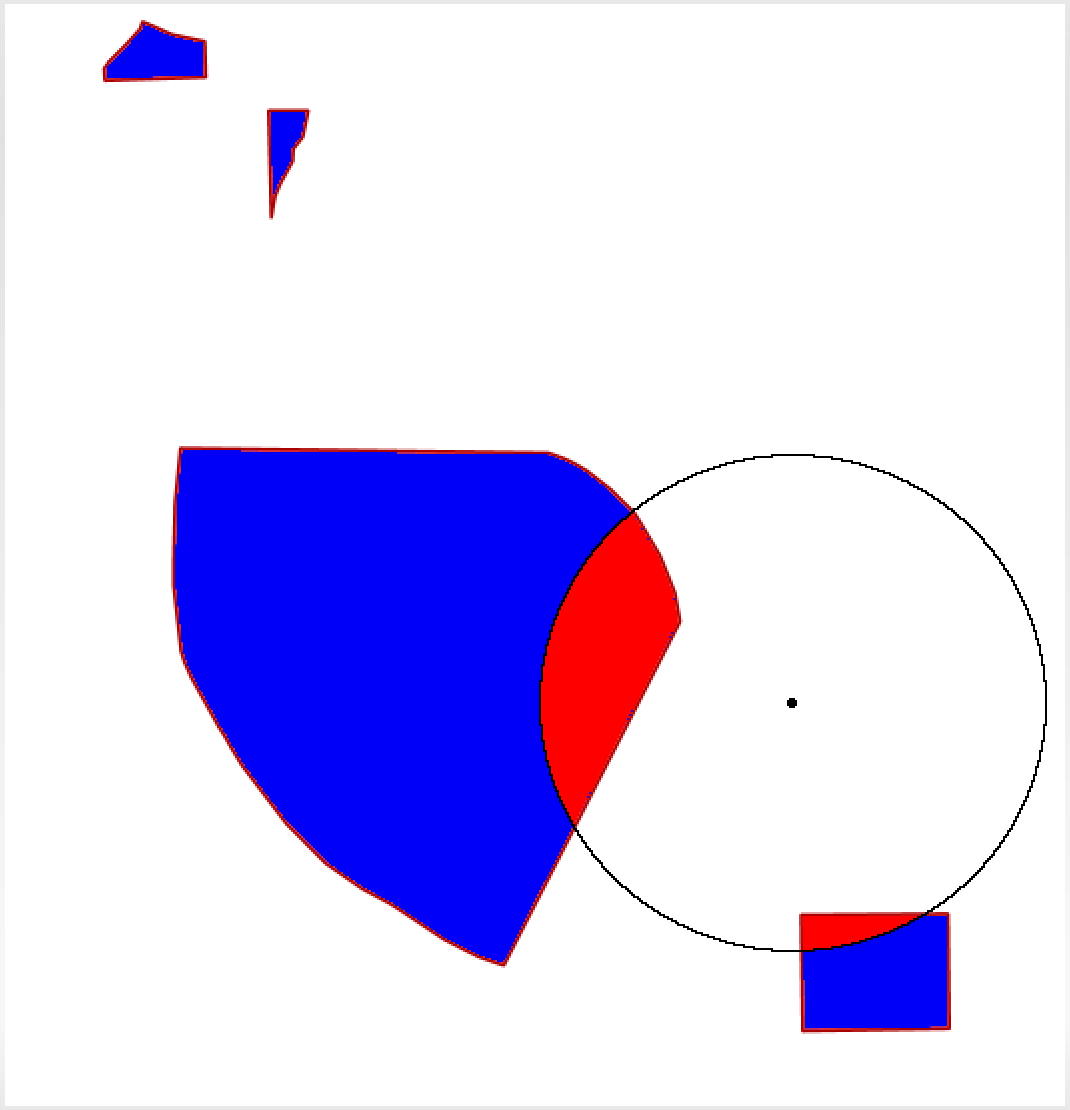
Principle


GeoTools
Overlay
ESRI Shapefile
Apache HTTP client
REST API
GeoServer
WMS, WFS, WCS

Steps

- Overlay (point, distance → circle) with features in layer
- Store output to ESRI Shapefile
- Call GeoServer via REST API to publish ESRI Shapefile as WMS and WFS

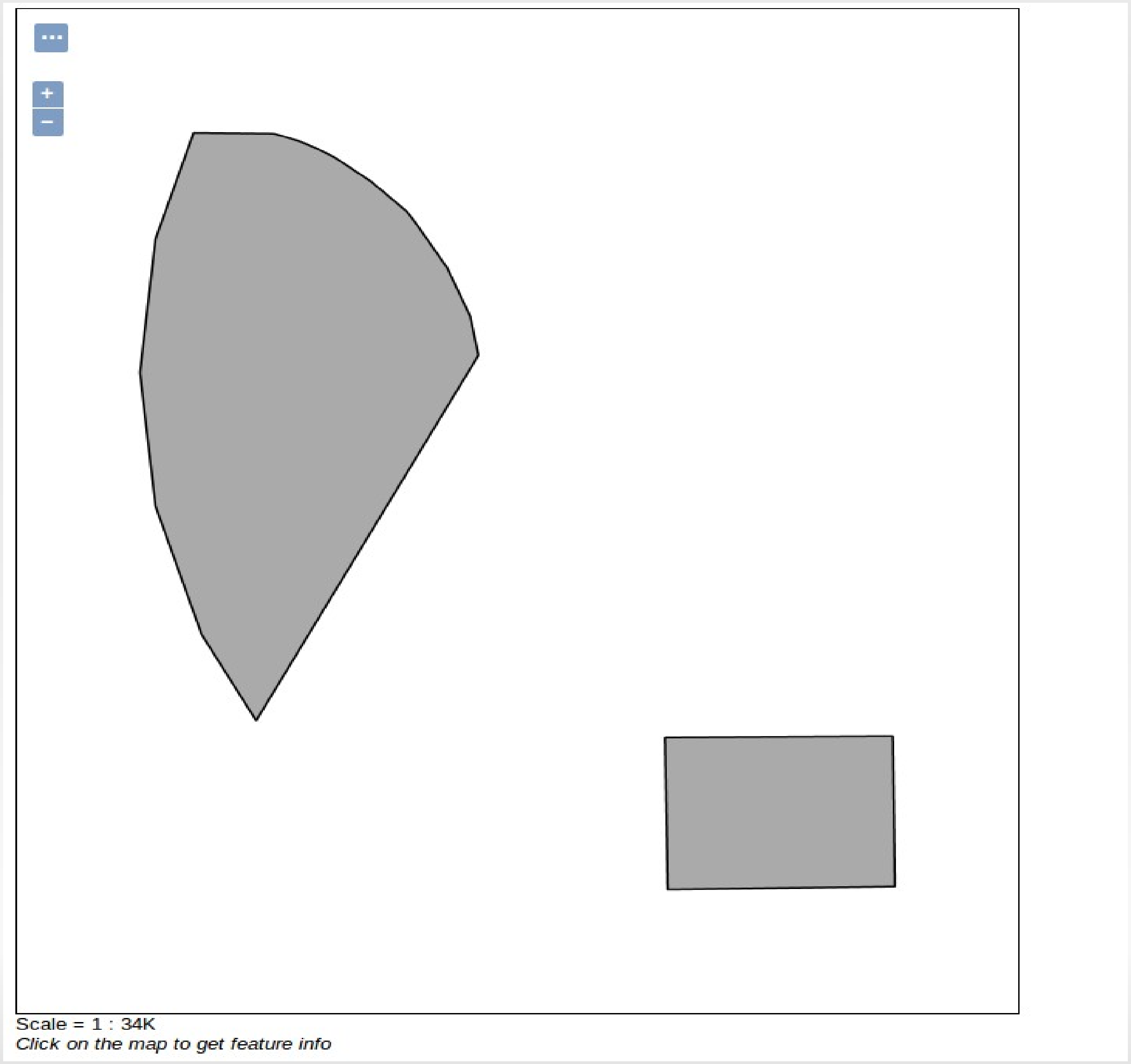
Example



Sample output

- http://localhost:8080/geoserver/w1443777998661/wms?service=WMS&version=1.3.0&request=GetCapabilities

Sample output



Scale = 1 : 34K

Click on the map to get feature info

WMS

Overlay

- Described before

Store ESRI Shapefile

- SimpleFeatureType
- List<SimpleFeature>
- ShapefileDataStoreFactory
- Transaction

Obrain SimpleFeatureType

- Use layer that was used for overlay
- Output of overlay will be same type as input – polygons (their parts) from layer restricted

```
SimpleFeatureType TYPE = fs.getSchema();
```

Get feature collection

- Use layer restricted
- Just change geometry

Get feature collection

```
Polygon p1 = (Polygon) point.buffer(distance);
List<SimpleFeature> features = new ArrayList<>();
SimpleFeatureIterator sfi = fs.getFeatures().features();
while (sfi.hasNext()) {
    SimpleFeature sf = sfi.next();
    MultiPolygon mp2 = (MultiPolygon)
  sf.getDefaultGeometry();
    Polygon p2 = (Polygon) mp2.getGeometryN(0);
    Polygon p3 = (Polygon) p2.intersection(p1);
    if (p3.getArea() > 0) {
        sf.setDefaultGeometry(p3);
        features.add(sf);
```

Place to store data

- Directories

```
long milis = System.currentTimeMillis();
String workspace = "w" +
  String.valueOf(milis);

new File("/data/install/geoserver/geoserver-
  2.8.2/data_dir/data/" + workspace +
  "/overlay/").mkdirs();
```

Sture data

- Parameters for storing

```
File newFile = new File(path);

Map<String, Serializable> params = new
  HashMap<String, Serializable>();
params.put("url", newFile.toURI().toURL());
```

ShapefileFactory

```
ShapefileDataStoreFactory dataStoreFactory = new
    ShapefileDataStoreFactory();

ShapefileDataStore newDataStore = (ShapefileDataStore)
    dataStoreFactory.createNewDataStore(params);

newDataStore.createSchema(TYPE);
newDataStore.forceSchemaCRS(TYPE.getGeometryDescri
    ptor().getCoordinateReferenceSystem());
```

Transaction

```
Transaction transaction = new
    DefaultTransaction("create");
String typeName = newDataStore.getTypeNames()[0];
SimpleFeatureSource featureSource =
    newDataStore.getFeatureSource(typeName);
SimpleFeatureStore featureStore = (SimpleFeatureStore)
    featureSource;
SimpleFeatureCollection collection = new
    ListFeatureCollection(TYPE, features);
featureStore.setTransaction(transaction);
featureStore.addFeatures(collection);
transaction.commit();
transaction.close();
```

Publish data

- GeoServer
- REST API

REST API

- REpresentational State Transfer
- Sources
- HTTP
  - GET
  - POST
  - PUT
  - DELETE

Create workspace

- Resource: http://localhost:8080/geoserver/rest/workspaces
- POST
- <workspace><name>name</name></workspace>

Publish ESRI Shapefile

- Resource: http://localhost:8080/geoserver/rest/workspaces/workspaceName/datastores/datastoreName/external.shp
- PUT
- Path to SHP file

Create workspace

- Apache HTTP client
- Create POST method
- Set authentification items
- Send request

Create POST method

```
String strURL =
    "http://localhost:8080/geoserver/rest/workspaces";

PostMethod post = new PostMethod(strURL);


post.setRequestHeader("Content-type", "text/xml");

post.setRequestEntity(new StringRequestEntity("<?xml
    version=\"1.0\"?><workspace><name>" + name +
    "</name></workspace>"));

post.setDoAuthentication(true);
```

Set up authentification

```
HttpClient httpclient = new HttpClient();

Credentials defaultcreds = new
   UsernamePasswordCredentials("admin", "geoserver");

httpclient.getState().setCredentials(new
   AuthScope("localhost", 8080, AuthScope.ANY_REALM),
   defaultcreds);
```

Send request

```
int response = httpclient.executeMethod(post);
post.releaseConnection();
```

Publish ESRI Shapefile

- Apache HTTP client
- Create PUT method
- Set authentification items
- Send request

PUT Method

```
String strURL =
   "http://localhost:8080/geoserver/rest/workspaces/" +
   workspace + "/datastores/" + datastore +
   "/external.shp";

PutMethod put = new PutMethod(strURL);

String shp = "file:///data/install/geoserver/geoserver-
   2.8.2/data_dir/data/" + workspace +
   "/overlay/overlay.shp"

put.setRequestHeader("Content-type", "text/plain");

put.setRequestEntity(new StringRequestEntity(shp));

put.setDoAuthentication(true);
```

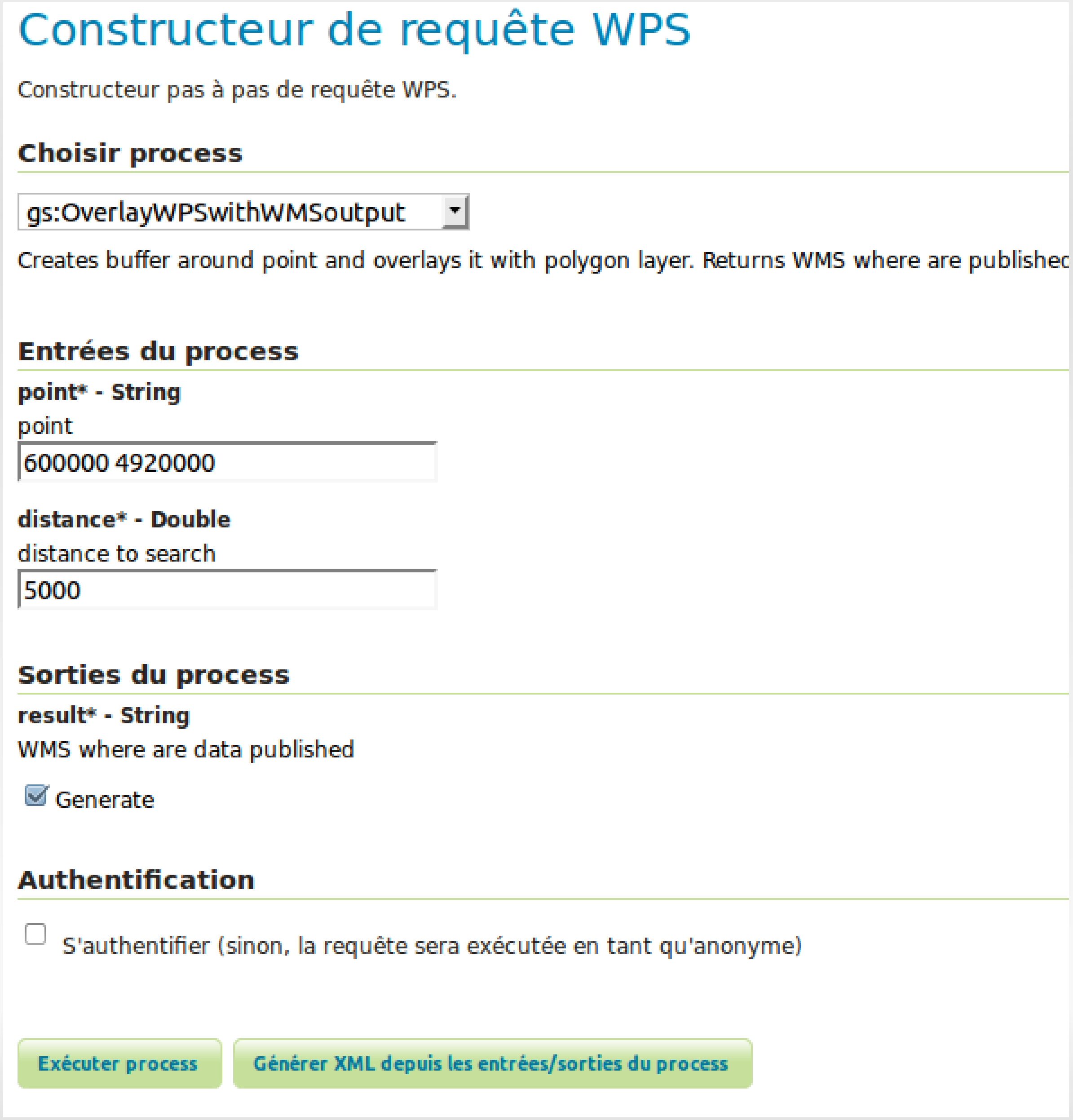
Place to service

```
@DescribeProcess(title="overlayWPS",
   description="Creates buffer around point and overlays
   it with polygon layer. Returns WMS where are published
   results of overlay.")
public class OverlayWPS implements GeoServerProcess {
 ...
}
```

Zapouzdření do služby / 2

```
@DescribeResult(name="result", description="WMS
  where are data published")
    public String
  execute(@DescribeParameter(name="point",
  description="point") String point,
  @DescribeParameter(name="distance",
  description="distance to search") double
  distance) {
       Examples e = new Examples();
       return e.overlay(point, distance);
    }
```

Input WPS



Constructeur de requête WPS

Constructeur pas à pas de requête WPS.

Choisir process

gs:OverlayWPSwithWMSoutput

Creates buffer around point and overlays it with polygon layer. Returns WMS where are publishec

Entrées du process

point* - String

point

600000 4920000

distance* - Double

distance to search

5000

Sorties du process

result* - String

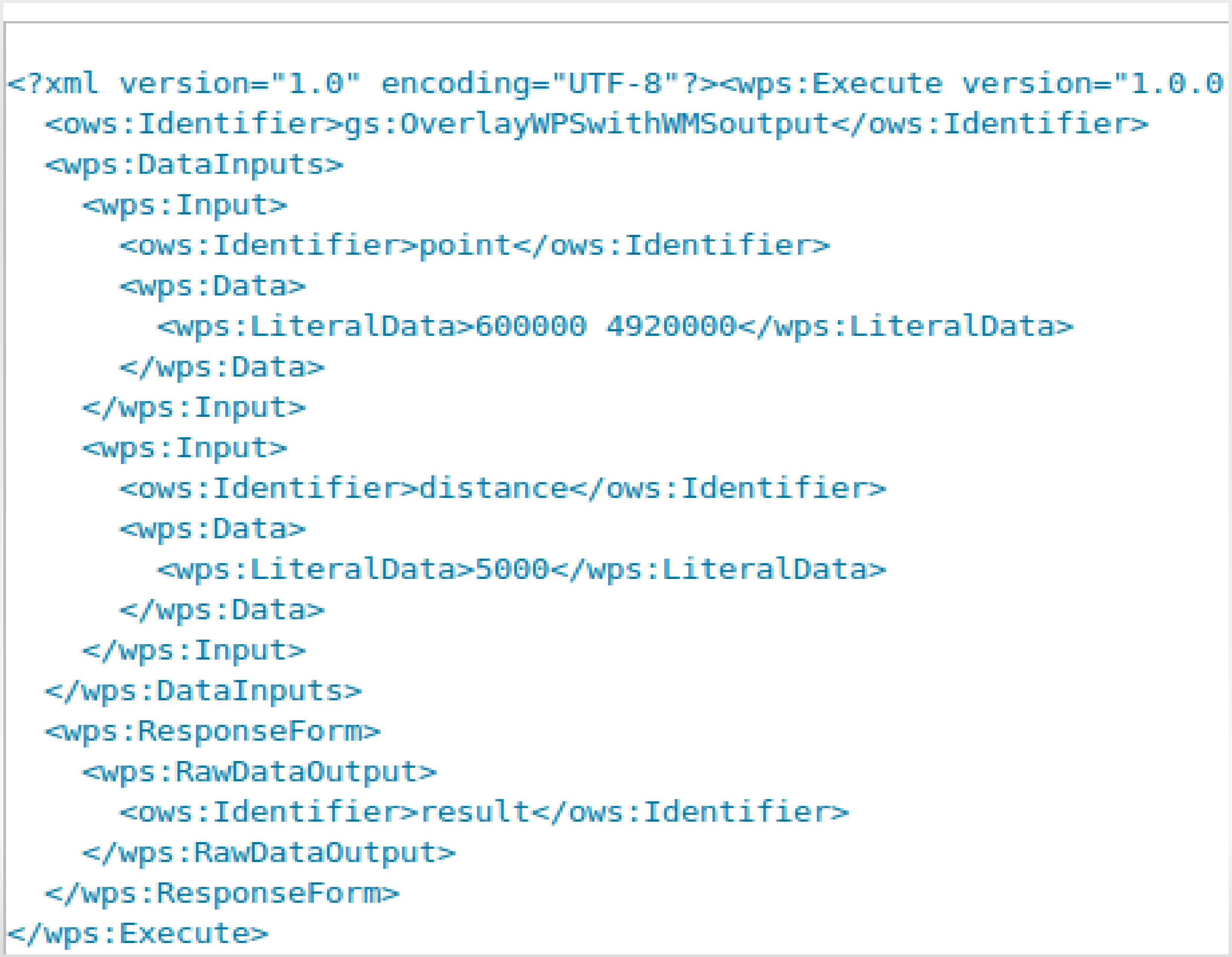
WMS where are data published

☑ Generate

Authentification

☐ S'authentifier (sinon, la requête sera exécutée en tant qu'anonyme)

Input WPS

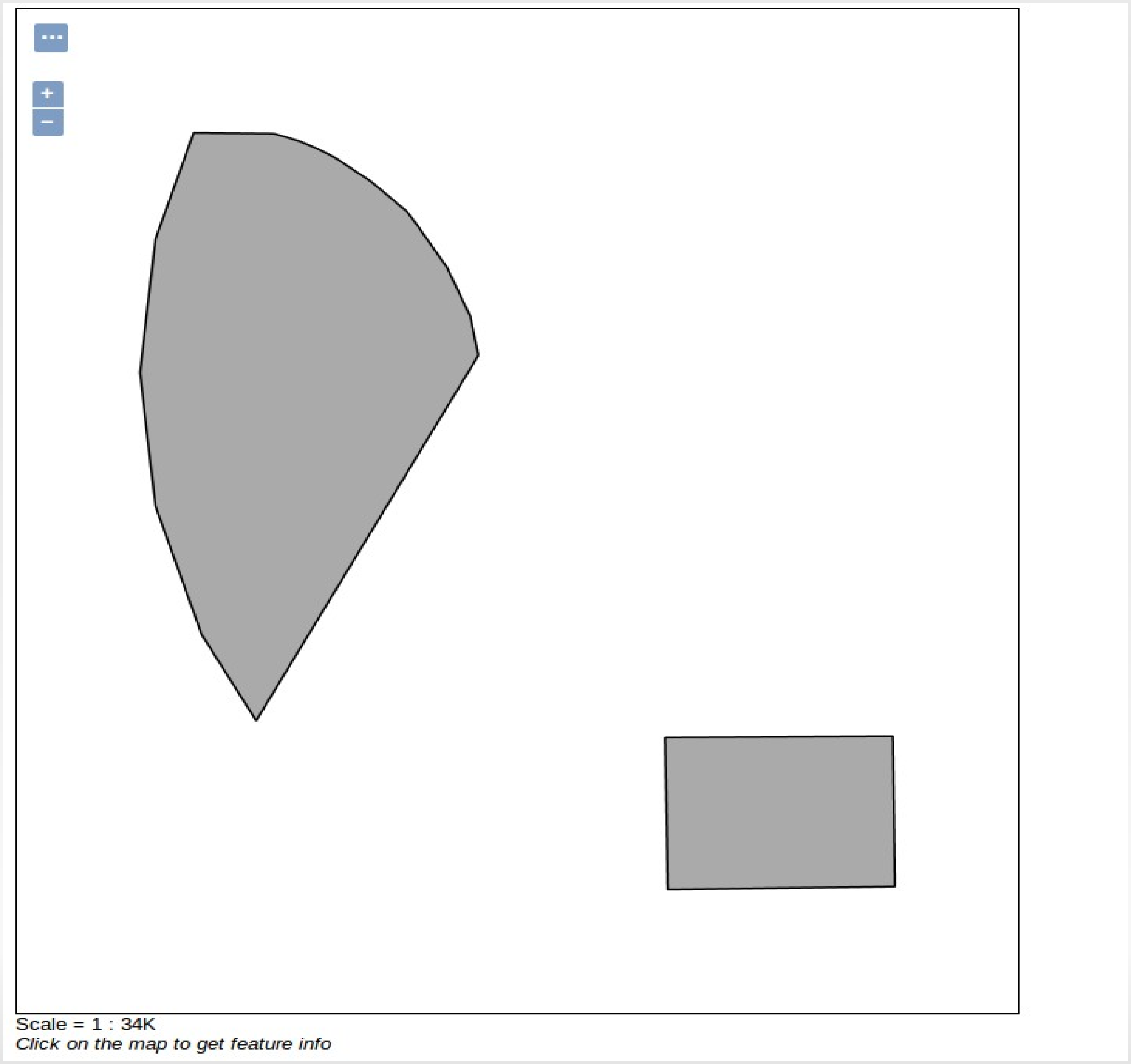


```xml
<?xml version="1.0" encoding="UTF-8"?><wps:Execute version="1.0.0
  <ows:Identifier>gs:OverlayWPSwithWMSoutput</ows:Identifier>
  <wps:DataInputs>
    <wps:Input>
      <ows:Identifier>point</ows:Identifier>
      <wps:Data>
        <wps:LiteralData>600000 4920000</wps:LiteralData>
      </wps:Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>distance</ows:Identifier>
      <wps:Data>
        <wps:LiteralData>5000</wps:LiteralData>
      </wps:Data>
    </wps:Input>
  </wps:DataInputs>
  <wps:ResponseForm>
    <wps:RawDataOutput>
      <ows:Identifier>result</ows:Identifier>
    </wps:RawDataOutput>
  </wps:ResponseForm>
</wps:Execute>
```

Output WPS

http://localhost:8080/geoserver/w1443780615378/wms?service=WMS&version=1.3.0&request=GetCapabilities

Output



Scale = 1 : 34K
Click on the map to get feature info

WMS