# Client-side WFS plugin

The purpose of this plugin would be to provide access to WFS servers from the client-side.

Currently we already support WFS server layers through the Geotools WFSDatastore. As we are gradually increasing the separation between client and server, and as WFS servers are gradually becoming more accessible in a web environment (through JSON support), it becomes necessary to provide WFS functionality at the client-side (as we have done before for WMS).

In an initial stage, the plugin could still proxy WFS commands to the server, but without needing to configure server-side WFS layers (as is the case now).

The following commmands would be a minimal subset that should be supported in an initial version:

- DescribeFeatureType: this command returns the featuretype description or schema of a specific layer
- GetFeature: this command returns a selection of features that conform to a certain filter

The following methods API methods are proposed for the WFSClient:

- void describeFeatureType(String baseUrl, String typeName, Callback<WfsLayerConfiguration, String> callback)
- void getFeature(WfsLayerConfiguration config, Query query, Callback<List<Feature> features, String> callback)

Both methods are asynchronous and receive callbacks with the layer configuration and list of features respectively.

WfsLayerConfiguration is a configuration object for the client-side WFS layer (analogous to WmsLayerConfiguration). The client-side WFS layer should be a specific subtype of a more general feature-based layer. The rendering of the features should be done by means of the graphics or canvas api. The fetching strategy of the layer should also be configurable.

The Query object contains a criterion (= filter in WFS terms), which can be a composition of logical, geographical and attribute filters. It also defines things like the maximum number of features, the offset index and the list of attributes to return. Question: could this be a DTO field for the WFSGetFeatureRequest or do we need a separate object ?

In a first version, the WFS functionality itself will mainly be used to enhance the existing feature info behavior of the client-side WMS layers. We may also include a simple implementation of the client-side layer based on a fetch-once strategy (suitable for layers with few features).

The support for searching features can be exposed in a separate layer interface:

```
public interface FeatureSearchSupported {

    void searchFeatures(CriterionDto filter, final Callback<FeatureCollection, String> callback);

}
```

The following API is proposed for the server-side commands:

- WFSDescribeFeatureTypeRequest
- WFSDescribeFeatureTypeResponse
- WFSGetFeatureRequest
- WFSGetFeatureResponse