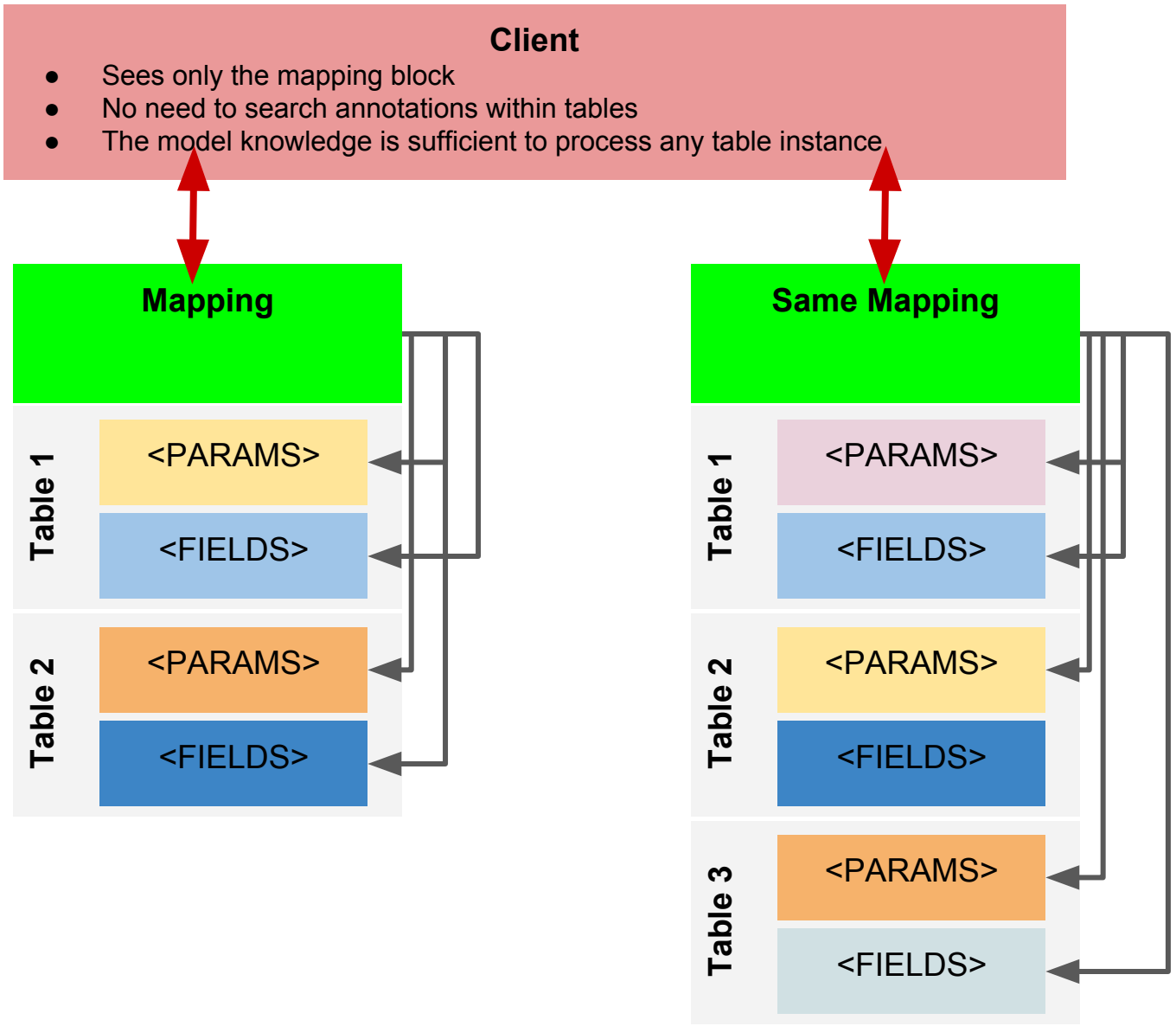
A low-fat mapping

Client Sees Only The Mapping Block



Syntax: Pro & Con

- **Cons**
  - New XML elements
  - Un bit more chatty than <GROUP>
  - Reputable for being complex
- **Pros**
  - Indépendant from data blocks
  - Easy to retrieve data elements from different tables
  - The parser has just to read the mapping blocks, indirections are resolved in the inner data model of the library.
  - Classes can have multiple instances
    - Ex: 2 positions with 2 different frames

```
<TEMPLATES tableref="ndgnsolidgdea">
    <TUPLE dmrole='root'>
        <TUPLE dmrole='timeseries:TimeSerie.dataSet'>
            <VALUE dmrole='timeseries:dataset.DataSet.calib_level' source='@calibLevel'/>
            <VALUE dmrole='timeseries:dataset.DataSet.creator' source='@creat' />
            <VALUE dmrole='timeseries:dataset.DataSet.contributor' source='@cont'/>
            <VALUE dmrole='timeseries:dataset.DataSet.publisher_did' source='@pubDID'/>
            <VALUE dmrole='timeseries:dataset.DataSet.target' source='@targ'/>
        </TUPLE>

        <TUPLE dmrole='timeseries:TimeSerie.spaceFrame' tableref='coosys'/>
        <TUPLE dmrole='timeseries:TimeSerie.timeFrame'  tableref='coosys'/>
        <TUPLE dmrole='timeseries:TimeSerie.filter'     tableref='coosys'/>
        <TUPLE dmrole='timeseries:TimeSerie.refPosition'  tableref='char' />

        <TUPLE dmrole='timeseries:TimeSerie.dependantModelDescriptor'>
            <VALUE dmrole='timeseries:dataset.DependantModelDescriptor.name' source='child'>Laurent</VALUE>
            <VALUE dmrole='timeseries:dataset.DependantModelDescriptor.ivoid' source='child'>ivo://vodml/lite/laurent</VALUE>
            <VALUE dmrole='timeseries:dataset.DependantModelDescriptor.url' source='child'>http://vodml/lite/laurent</VALUE>
        </TUPLE>

        <COLLECTION dmrole="timeseries:TimeSerie.pointsJ">
            <TUPLE dmrole="timeseries:data.PointJ" tableref="data" />
        </COLLECTION>
        <COLLECTION dmrole="timeseries:TimeSerie.pointsH">
            <TUPLE dmrole="timeseries:data.PointH" tableref="data" />
        </COLLECTION>
        <COLLECTION dmrole="timeseries:TimeSerie.pointsK">
            <TUPLE dmrole="timeseries:data.PointK" tableref="data" />
        </COLLECTION>
        <COLLECTION dmrole="timeseries:TimeSerie.pointsL">
            <TUPLE dmrole="timeseries:data.PointL" tableref="data" />
        </COLLECTION>
        <COLLECTION dmrole="timeseries:TimeSerie.pointsM">
            <TUPLE dmrole="timeseries:data.PointM" tableref="data" />
        </COLLECTION>
    </TUPLE>
</TEMPLATES>
```

Python API (prototype)

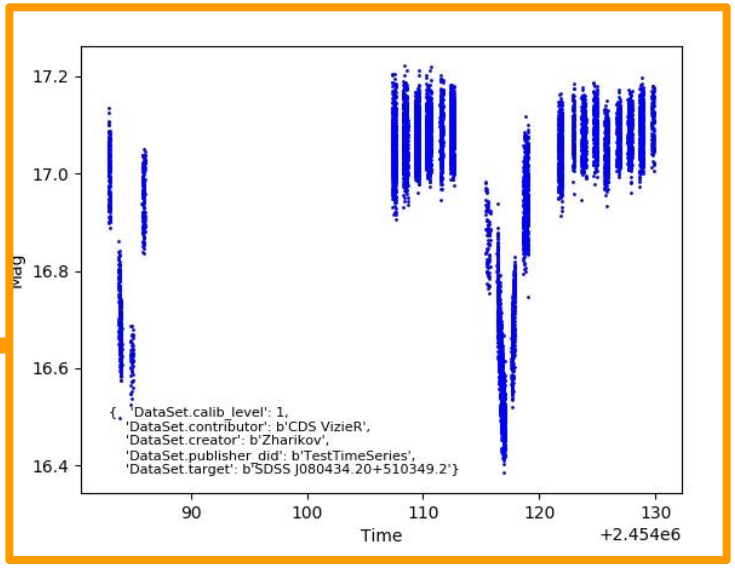
```
file_to_process = "../SDSS_J080434.20+510349.2_VizieR.xml"
mapper = Mapper("SDSS", file_to_process)
mapper.read_annotation()
print(mapper.__repr__())

# Read data and initiate an iterator
iterator = Iterator(file_to_process, mapper)
pp = pprint.PrettyPrinter(indent=4)
x = []
y = []
header = {}
while iterator.hasNext() :
    #pp.pprint( iterator.next_object())
    response = iterator.next_object()
    print(iterator.get_listed_columns())
    for k,v in response.items() :
        print(k)

    data = response["#timeseries:TimeSeries.Points#timeseries:data.Point"]
    for k,v in response.items() :
        if not k.startswith("#"):
            header[k] = v

    for point in data:
        x.append(point['Point.timestamp'])
        y.append(point['Point.observable'])
    # Let's assume we have only one TS in this file

"""
Plotting section
"""
fig, ax1 = plt.subplots()

plt.plot(x, y, "bo", markersize=1)
plt.xlabel('Time')
plt.ylabel('Mag')

ax1.text(x[0], min(y), pp.pformat(header), fontsize=8)
plt.show()
```

Parser init

Data extraction

The client has just to know a few keys pointing to real data data

Plot



Possible Workflow for a Data Provider