# Toward a PyVO API for MANGO

Laurent Michel - Gilles Landais - François Bonnarel - Mireille Louys


**tomdonaldson** commente

https://github.com/astropy/astroquery/issues/2036#issuecomment-839820679

**@gilleslandais** Where this sort of functionality belongs is an interesting question.

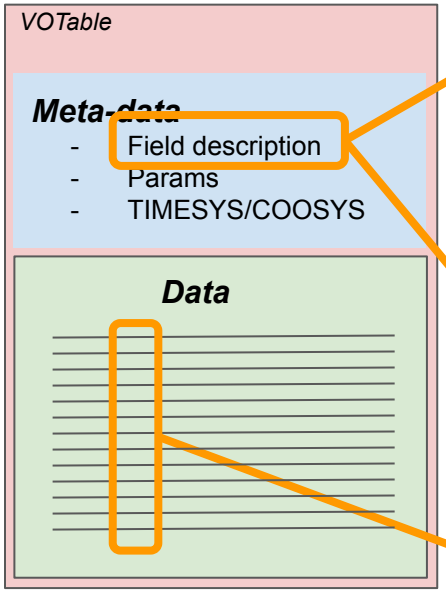
- As **@bsipocz** said, the fundamental VOTable parsing (and writing) happens in astropy core ( `astropy.io.votable` ).
- The `pyvo` package depends on `astropy` and implements some value-added features for querying using various standards which includes some semantics and assumptions about the VOTable results based on the standard used in the query. It also has a more robust representation of the VOTable itself which might be amenable to Mango metadata.
- Then `astroquery` depends on `pyvo` for its TAP and DataLink features.

It's clear that anything that is a VO standard should either be in `astropy.io.votable` or `pyvo` . Which of those is the best place is still an interesting question and will likely depend on the feature, and how fundamental it is to VOTable I/O.

For features that are specific to a data provider and will likely never be standard, then `astroquery` may be the most sensible place for the feature.

For features that are in between, i.e., working towards becoming a standard, then it's less clear, but again will depend on the feature. For example, with Mango, even with a desire to keep the functionality modularly separate from `astropy.io.votable` , that module will likely still need updates to accept the new metadata. That said, `pyvo` seems like a better sandbox for progressing a standard than `astropy` where possible.

# PyVO API at a Glance




```
results = tap_service.search("SELECT TOP 10 * FROM ivoa.obscore")

results.fieldnames
('dataproduct_type', 'dataproduct_subtype', 'calib_level', 'obs_collection',
    'obs_id', 'obs_title', 'obs_publisher_did', 'obs_creator_did',
    'access_url', 'access_format', 'access_estsize', 'target_name',
    'target_class', 's_ra', 's_dec', 's_fov', 's_region', 's_resolution',
    't_min', 't_max', 't_exptime', 't_resolution', 'em_min', 'em_max',
    'em_res_power', 'o_ucd', 'pol_states', 'facility_name', 'instrument_name',
    's_xel1', 's_xel2', 't_xel', 'em_xel', 'pol_xel', 's_pixel_scale',
    'em_ucd', 'preview', 'source_table')
```

```
results.getdesc("access_url")
<FIELD ID="access_url" arraysize="*" datatype="char" name="access_url" ucd
    ="meta.ref.url" utype="obscore:access.reference"/>
```

```
for row in results:
    print(row["s_ra"])
354.464679166665
354.464679166665
354.464679166665
354.47153333334
354.47153333334
```

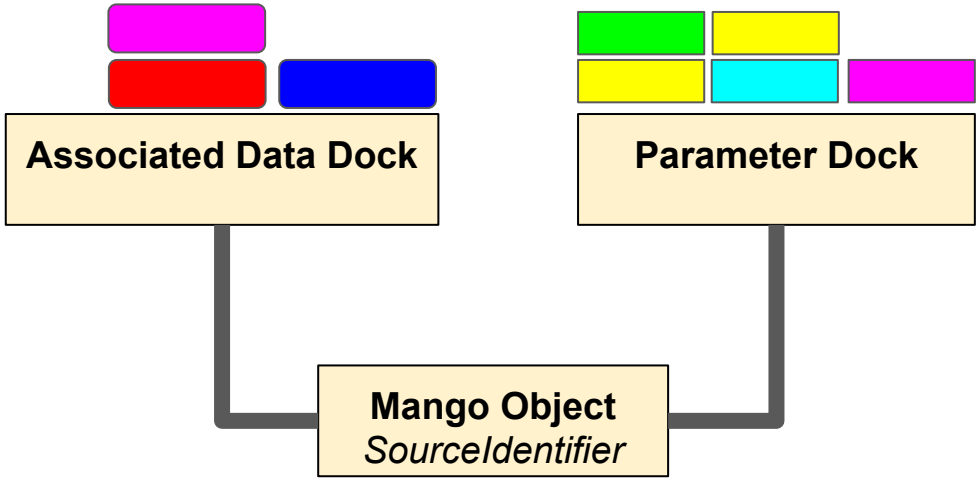
# MANGO: a Container Model

## What a MANGO object is

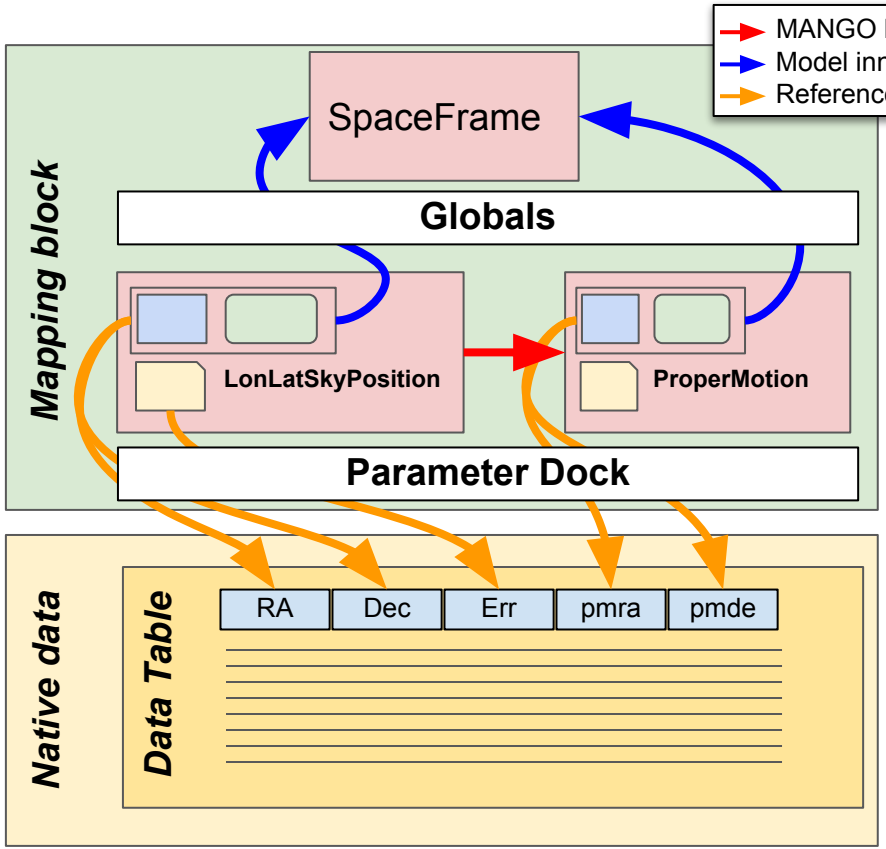
- One **identifier**
- An unbounded set of **Parameters**
  - Set of simple values (string or numerical)
- A unbounded set of **associated data**
  - Detections, spectra, link to paper...

## What MANGO is for

- Providing a more **elaborate** schema for **meta data**
  - Explicit association **value-errors**
  - Elaborate description of **coordinate systems**
  - How **table data relate** each to other
  - How data of different tables are **joined**
- providing **common path** to consume elaborate meta-data



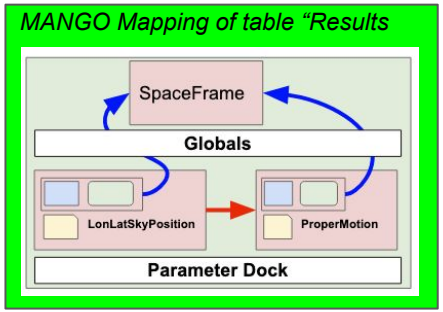
# MANGO Mapping

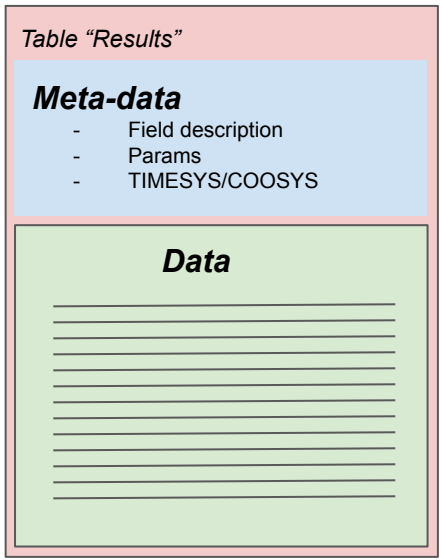



- MANGO Mapping block: an **elaborate view** on the **meta-data**
  - Bridge connecting FIELDS with model leaves
  - Complete native meta-data
  - Show how different elements are nested or connected

- **Suitable mapping design**
  - Mapping blocks have the same scope as mapped meta-data
  - Explicit connection with FIELDs
  - It can be consumed by an API very similar to the regular on

# Getting the list of the Mapped MANGO Parameters




```
mango_browser = MangoBrowser(votable_path)
mango_parameter_list = mango_browser.get_parameter_list()

[
  "#0 meta.id;meta.main",
  "#1 pos",
  "#2 phot.flux",
  "#3 phot.flux",
  "#4 phot.flux",
  "#5 phot.flux",
  "#6 phot.flux",
  "#7 phot.flux;arith.ratio",
  "#8 phot.flux;arith.ratio",
  "#9 phot.flux;arith.ratio",
  "#10 phot.flux;arith.ratio",
  "#11 time.duration;obs.exposure",
  "#12 time;obs.start",
  "#13 meta.code.qual"
]
```

**A MANGO parameter can include several columns**

- e.g. ra, dec, error

# Getting the Description of one MANGO Parameter

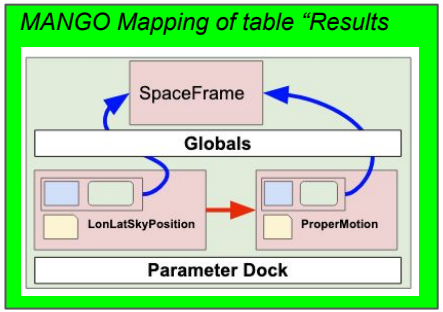



*Table “Results”*

***Meta-data***

- Field description
- Params
- TIMESYS/COOSYS

***Data***

```
DictUtils.print_pretty_json(mango_browser.get_parameter("#1 pos"))
```


```
{
  "coord_type": "mango:stcextend.LonLatPoint",
  "coords:SpaceFrame": {
    "@ID": "SpaceFrame_ICRS",
    "@dmtype": "coords:SpaceFrame",
    "coords:SpaceFrame.spaceRefFrame": {
      "@value": "ICRS"
    }
  },
  "coosys_type": "coords:SpaceFrame",
  "description": "Corrected position",
  "error_type": "meas:Error",
  "mango:stcextend.LonLatSkyPosition": {
    "field:latitude": {
      "id": "_dec_147",
      "index": 1
    },
    "field:longitude": {
      "id": "_ra_146",
      "index": 0
    }
  },
  "meas:Error": {
    "field:meas:Symmetrical.radius": {
      "id": "_poserr_148",
      "index": 2
    }
  },
  "measure_type": "mango:stcextend.LonLatSkyPosition",
  "semantic": "#postion.corrected",
  "ucd": "pos"
}
```

# Getting the Value of  one MANGO Parameters

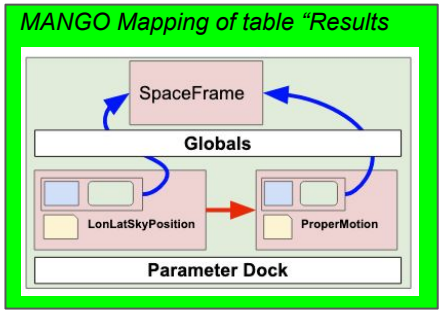



*Table “Results”*

***Meta-data***

- Field description
- Params
- TIMESYS/COOSYS

***Data***

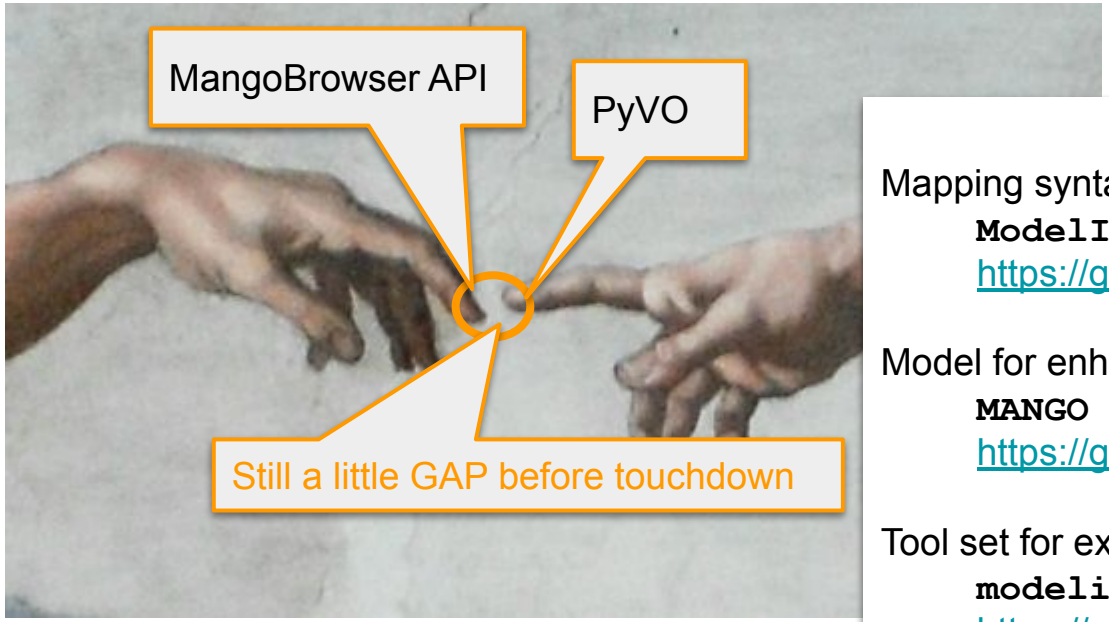
```
DictUtils.print_pretty_json(mango_browser.get_data_by_key("#1 pos"))

{
  "data": [
    [
      340.91055060369,
      -17.071667101891,
      1.50765
    ],
   ...
  ],
  "head": [
    "field:longitude [#1 pos]",
    "field:latitude [#1 pos]",
    "error: field:meas:Symmetrical.radius [#1 pos]"
  ],
  "selected_index": [
    0,
    1,
    2
  ]
}
```

# AstroPy/MangoBrower Similarities

| PyVO ResultSet | | MangoBrowser | |
|---|---|---|---|
| **fieldnames** | Name list | **get_parameter_list**() | Identifier list |
| **getdesc**(fieldname) | <FIELD> instance | **get_parameter**(param_id) | Complete parameter description in a Python dictionary |
| Iterate on **row**[fieldname] | List of individual values | **get_data_by_key**(param_id) | List of value vectors Implementing an iterator would be easy |

# Here we Are




Mapping syntax proposal:

**`ModelInstanceInVot`**

https://github.com/ivoa-std/ModelInstanceInVot

Model for enhancing meta-data:

**`MANGO`**

https://github.com/ivoa-std/MANGO

Tool set for exercising data annotation

**`modelinstanceinvot-code`**

https://github.com/ivoa/modelinstanceinvot-code

# Backup

# API: Proof of Concept

- Non normative: based on dictionaries
  - Easy ot process on many languages
  - Model roles used as keys

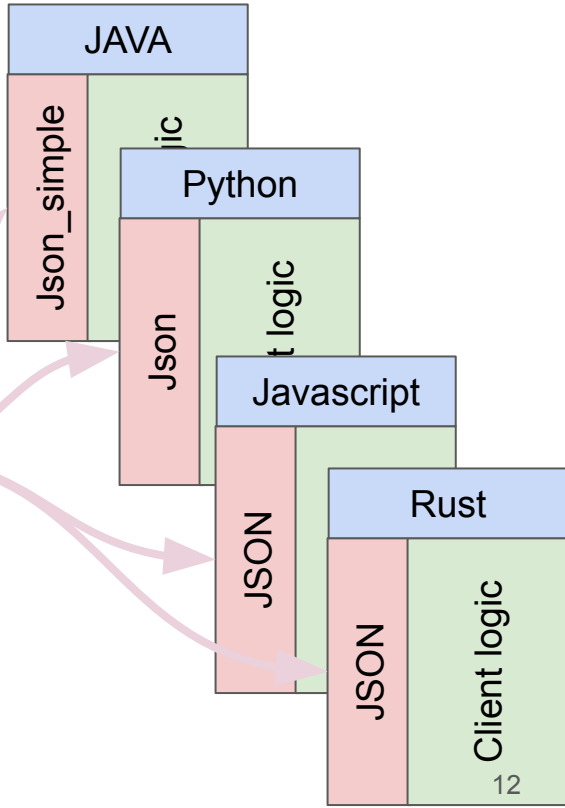
```
<INSTANCE dmrole="mango:Parameter.measure"
    dmtype="mango:Parameter">
    <ATTRIBUTE dmrole="mango:Parameter.semantic"
        dmtype="ivoa:string" value="#position" />
    <ATTRIBUTE dmrole="mango:Parameter.ucd"
        dmtype="ivoa:string" value="pos.eq;meta.main" />
    <ATTRIBUTE dmrole="mango:Parameter.description"
        dmtype="ivoa:string" value="this is the position" />
    <INSTANCE dmrole="mango:Parameter.measure">
    <INSTANCE dmrole="meas:Measure.error">
</INSTANCE>
```

***XML mapping block***

XML2JSON
+
Formatting (Py)

```
"#1 pos": {
  "coord_type": "mango:stcextend.LonLatPoint",
  "coords:SpaceFrame": {
    "@ID": "SpaceFrame_ICRS",
    "@dmtype": "coords:SpaceFrame",
    "coords:SpaceFrame.equinox": {
      "@dmtype": "coords:Epoch",
      "@value": "NoSet"
    },
    "coords:SpaceFrame.refPosition": {
      "@dmtype": "coords:StdRefLocation",
      "coords:StdRefLocation.position": {
        "@dmtype": "ivoa:string",
        "@value": "NoSet"
      }
    },
    "coords:SpaceFrame.spaceRefFrame": {
      "@dmtype": "ivoa:string",
      "@value": "ICRS"
    }
  },
  "coosys_type": "coords:SpaceFrame",
  "description": "Corrected position",
  "error_type": "meas:Error",
  "mango:stcextend.LonLatSkyPosition": {
    "field:latitude": {
      "id": "_dec_147",
      "index": 1
    },
    "field:longitude": {
      "id": "_ra_146",
      "index": 0
    }
  },
  "meas:Error": {
    "field:meas:Symmetrical.radius": {
      "id": "_poserr_148",
      "index": 2
    },
    "unit": "NotSet"
  },
  "measure_type": "mango:stcextend.LonLatSkyPosition",
  "semantic": "#position.corrected",
  "ucd": "pos"
},
```

***JSON serialization***
*Keys are DM roles*



# API: Output keep connected to native data

***Extracting a position from MANGO annotation***

```
mango_data = mango_browser.get_data(measure_type="mango:stcextend.LonLatSkyPosition")
DictUtils.print_pretty_json(mango_data)
```

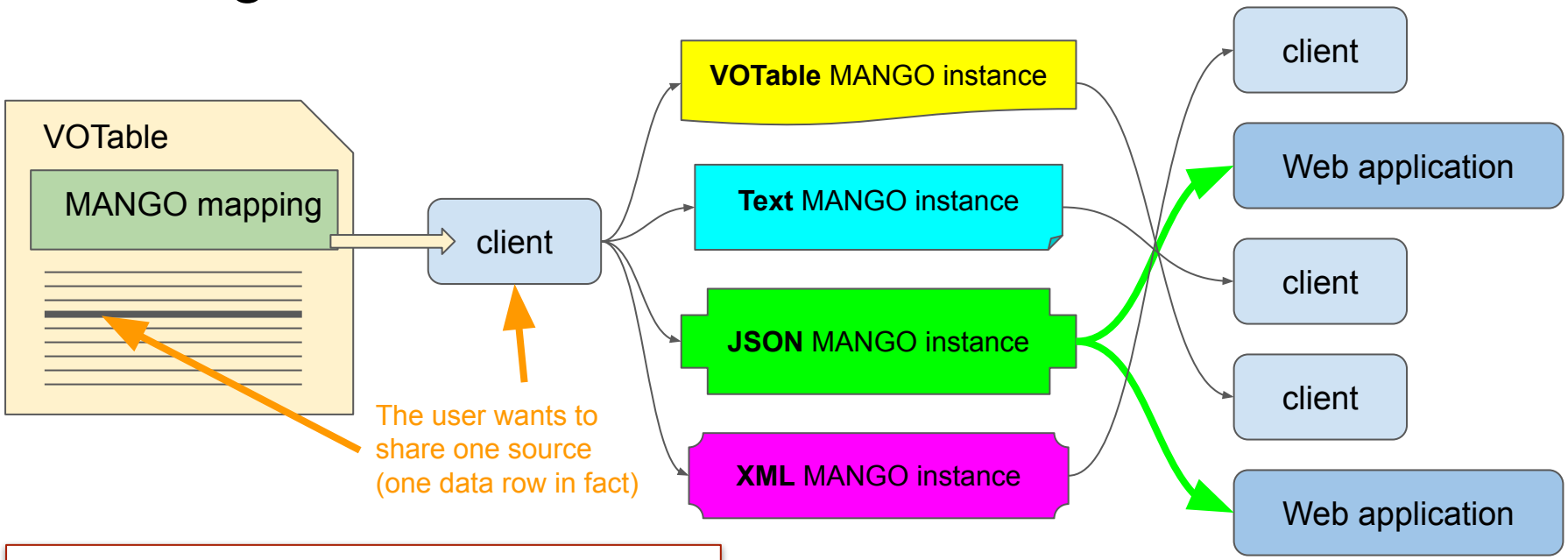
```
{
  "data": [
    [
      340.91055060369,
      -17.071667101891,
      1.50765
    ]
  ],
  "head": [
    "field:longitude [#1 pos]",
    "field:latitude [#1 pos]",
    "error: field:meas:Symmetrical.radius [#1 pos]"
  ],
  "selected_index": [
    0,
    1,
    2
  ]
}
```

Sky position read (limited here to one row)

Model attribute references
Labels can be used as keys to get more information

VOTable column Indices

# Sharing MANGO instances




- Using an **integrated model** facilitate the data sharing (e.g.SAMP) with **different serialisation** modes
- **Less easy** with **sparse** model **components**