+++ title = “Legacy data source plugins” keywords = [“grafana”, “plugins”, “documentation”] aliases = [“/docs/grafana/latest/plugins/developing/datasources/”] +++

Legacy data source plugins

Data source plugins enable people to develop plugins for any database that communicates over HTTP. Its up to the plugin to transform the data into time series data so that any grafana panel can then show it.

Data source development

> Our goal is not to have a very extensive documentation but rather have actual code that people can look at. Example implementations of a data source can be found in these repos:
>
> - simple-json-datasource
> - simple-datasource
> - simple-json-backend-datasource

To interact with the rest of grafana the plugins module file can export 4 different components.

- Datasource (Required)
- QueryCtrl (Required)
- ConfigCtrl (Required)
- AnnotationsQueryCtrl

Plugin json

There are two data source specific settings for the plugin.json

```
"metrics": true,
"annotations": false,
```

These settings indicate what kind of data the plugin can deliver. At least one of them has to be true.

Data source

The javascript object that communicates with the database and transforms data to times series.

The Data source should contain the following functions:

```
query(options); // used by panels to get data
testDatasource(); // used by data source configuration page to make sure the connection is u
annotationQuery(options); // used by dashboards to get annotations
metricFindQuery(options); // used by query editor to get metric suggestions.
```

testDatasource

When a user clicks on the *Save & Test* button when adding a new data source, the details are first saved to the database and then the `testDatasource` function that is defined in your data source plugin will be called. It is recommended that this function makes a query to the data source that will also test that the authentication details are correct. This is so the data source is correctly configured when the user tries to write a query in a new dashboard.

Query

Request object passed to datasource.query function:

```
{
  "range": { "from": "2015-12-22T03:06:13.851Z", "to": "2015-12-22T06:48:24.137Z" },
  "interval": "5s",
  "targets": [
    { "refId": "B", "target": "upper_75" },
    { "refId": "A", "target": "upper_90" }
  ],
  "format": "json",
  "maxDataPoints": 2495 // decided by the panel
}
```

There are two different kinds of results for data sources: time series and table. Time series is the most common format and is supported by all data sources and panels. Table format is only supported by the InfluxDB data source and table panel. But we might see more of this in the future.

Time series response from datasource.query. An array of:

```
[
  {
    "target": "upper_75",
    "datapoints": [
      [622, 1450754160000],
      [365, 1450754220000]
    ]
  },
  {
    "target": "upper_90",
    "datapoints": [
      [861, 1450754160000],
      [767, 1450754220000]
    ]
  }
]
```

Table response from datasource.query. An array of:

```
[
  {
    "columns": [
      {
        "text": "Time",
        "type": "time",
        "sort": true,
        "desc": true
      },
      {
        "text": "mean"
      },
      {
        "text": "sum"
      }
    ],
    "rows": [
      [1457425380000, null, null],
      [1457425370000, 1002.76215352, 1002.76215352]
    ],
    "type": "table"
  }
]
```

Annotation Query

Request object passed to datasource.annotationQuery function:

```
{
  "range": { "from": "2016-03-04T04:07:55.144Z", "to": "2016-03-04T07:07:55.144Z" },
  "rangeRaw": { "from": "now-3h", "to": "now" },
  "annotation": {
    "datasource": "generic datasource",
    "enable": true,
    "name": "annotation name"
  },
  "dashboard": DashboardModel
}
```

Expected result from datasource.annotationQuery:

```
[
  {
    "annotation": {
      "name": "annotation name", //should match the annotation name in grafana
      "enabled": true,
      "datasource": "generic datasource"
```

```
    },
    "title": "Cluster outage",
    "time": 1457075272576,
    "text": "Joe causes brain split",
    "tags": ["joe", "cluster", "failure"]
  }
]
```

QueryCtrl

A JavaScript class that will be instantiated and treated as an Angular controller when the user edits metrics in a panel. This class has to inherit from the `app/plugins/sdk.QueryCtrl` class.

Requires a static template or `templateUrl` variable which will be rendered as the view for this controller.

ConfigCtrl

A JavaScript class that will be instantiated and treated as an Angular controller when a user tries to edit or create a new data source of this type.

Requires a static template or `templateUrl` variable which will be rendered as the view for this controller.

AnnotationsQueryCtrl

A JavaScript class that will be instantiated and treated as an Angular controller when the user chooses this type of data source in the templating menu in the dashboard.

Requires a static template or `templateUrl` variable which will be rendered as the view for this controller. The fields that are bound to this controller are then sent to the Database objects annotationQuery function.