Agenda

- Create Notebook
- Visualise Data

▼ Details

This tutorial teaches how to visualise data.

Before doing this tutorial make sure you read and completed the following tutorial and have data in your wendelin instance:

- HowTo Transform Data
- HowTo Resample Data

Notebook Module

▼ Details

On Modules page click on **Notebooks**.

Add Notebook

▼ Details

Click on **Add** to add a new Notebook.

Create Notebook

▼ Details

Click on **Create Document** to continue.

New Notebook

▼ Details

You will notice that the notebook shows only a default title and draft state, and an empty notebook viewer. The default view of a notebook displays its execution, in this case an empty notebook result.

To edit the document and the notebook code, click on "Edit" view on the VIEWS section of the left panel menu. Or through header button "Views" -> Edit.

Notebook

▼ Details

Fill in the Title.

The visualisation code will be at the bottom text area.

Notebook uses the same technology as Iodide.

To learn more about programming in Iodide environment checkout Iodide homepage.

Visualisation Script

```
%% md
# Time Series
<div class="plot_div" id="plot_div"/>


%% fetch
js: jio.js
js: ndarray_bundle.js
js: wendelin.js
js: https://cdn.plot.ly/plotly-latest.min.js


%% js
hateoas_url = "https://" + window.location.host + "/erp5/web_site_module/default_wendelin_front/hateoas/";
jio = jIO.createJIO({
  type: "erp5",
  url: hateoas_url,
  default_view_reference: "view"
});

gadget = {
  getSetting: function(property) {
    return new RSVP.Queue()
```

```
      .push(function () {
        if (property == "hateoas_url") {
          return hateoas_url;
        }
        return;
      });
  },
  jio_getAttachment: function(id, url, parameter_dict) {
    return jio.getAttachment(id, url, parameter_dict);
  },
  jio_get: function(id) {return jio.get(id);}
}

var graph = document.getElementById('plot_div'),
  graph_data = [],
  label_list = ["date", "humidity_min","humidity_mean","humidity_max", "pressure_min","pressure_mean","pressure_max", "temperature_min","temperature_mean", "temperature_max"];

selectorOptions = {
    "buttons": [{
        "step": 'day',
        "stepmode": 'backward',
        "count": 1,
        "label": 'day'
    },/* {
        "step": 'day',
        "stepmode": 'backward',
        "count": 7,
        "label": 'week'
    },*/ {
        "step": 'month',
        "stepmode": 'backward',
        "count": 1,
        "label": 'month'
    }, {
        "step": 'year',
        "stepmode": 'backward',
        "count": 1,
        "label": 'year'
    }],
}


function update_graph(start_date, stop_date){
  return getData(start_date, stop_date)
    .push(function (data) {
      return Plotly.react(graph,
        data,
        { 'title' :'Environmental Sensor Data',
          'xaxis': {
            'autorange': true,
            'rangeselector': selectorOptions,
          }
        }
      );
    });
}

function unpack(rows, key) {
  return rows.map(function(row) { return row[key]; });
}

function getData(start_date, stop_date){
  var graph_data=[],
    ratio = screen.availWidth / 1024,
    interval,
    frequency,
    start_index,
    stop_index,
    array_id;

   if (stop_date === undefined) {
     frequency = 30000;
     start_index = 0;
     stop_index = 0;
   } else {
     interval = stop_date - start_date;
     if (interval >= 1000 * 60 * 60 * 24 * 100 * ratio) {
       frequency = 30000;
     } else if (interval >= 1000 * 60 * 60 * 24 * 28 * ratio) {
       frequency = 3000;
     } else if (interval >= 1000 * 60 * 60 * 24 * 7 * ratio) {
       frequency = 300;
     } else {
       frequency = 60;
     }
   }
  return jio.allDocs({
    query: 'portal_type:"Data Analysis Line" AND ' +
          'title: "Resampled Array (' + frequency + ' Seconds)" AND ' +
          'resource_reference:"GENERIC-INTERVAL-RESAMPLED-ARRAY" AND ' +
          'parent_source_title:"Sample Factory" AND ' +
          'simulation_state:"started"'
  })
    .push(function (result) {
      var data_analysis_line_id = result.data.rows[0].id;
      return jio.allDocs({
        query: 'portal_type:"Data Array" AND ' +
              'aggregate_related_relative_url:"' + data_analysis_line_id +'"'
      });
    })
    .push(function (result) {
      array_id = result.data.rows[0].id;
      return wendelin.getArrayRawSlice(gadget, array_id, 0, 1);
```

```
    })
    .push(function (result) {
      array_start_date = wendelin.convertFirstColToDate([[result.data[0]]])[0][0];
      if (start_index === undefined) {
        start_index = Math.max(0, Math.ceil((start_date - array_start_date) / (frequency*1000))),
        stop_index = Math.ceil((stop_date - array_start_date) / (frequency*1000));
      }
      return wendelin.getArrayRawSlice(gadget, array_id, start_index, stop_index);
    })
    .push(function(result) {
      for (i = 0; i < label_list.length; i += 1) {
        graph_data = graph_data.concat(nj.unpack(result.pick( null, label_list[i])));
      }
      return graph_data
    })
    .push(function(result){
      var filtered_graph_data = [];
      for (var i=0; i<result.length; i++) {
        if (result[i][0] != 0) {
          filtered_graph_data.push(result[i]);
        }
      }
      return wendelin.convertFirstColToDate(filtered_graph_data)
    })
    .push(function (graph_data) {
      var humidity_trace = {
          type: "scattergl",
          mode: "markers",
          name: 'Humidity(%)',
          x: unpack(graph_data, label_list.indexOf("date")),
          y: unpack(graph_data, label_list.indexOf("humidity_mean")),
          line: {color: '#17BECF'}
      }

      var pressure_trace = {
          type: "scattergl",
          mode: "markers",
          name: 'Pressure(Pa)',
          x: unpack(graph_data, label_list.indexOf("date")),
          y: unpack(graph_data, label_list.indexOf("pressure_mean")),
          line: {color: '#7F7F7F'}
      }

      var temperature_trace = {
          type: "scattergl",
          mode: "markers",
          name: 'Temperature(C°)',
          x: unpack(graph_data, label_list.indexOf("date")),
          y: unpack(graph_data, label_list.indexOf("temperature_mean")),
          line: {color: '#FF0000'}
      }

      var data = [humidity_trace,pressure_trace,temperature_trace];
      return data
    });
}

function plot () {
  return getData()
    .push(function (data) {
      Plotly.newPlot(
        graph,
        data,
        { 'title' :'Environmental Sensor Data',
          'xaxis': {
          'autorange': true,
          'rangeselector': selectorOptions,}
        });
      graph.on('plotly_relayout', function(eventdata) {
        var start_date = new Date(eventdata["xaxis.range[0]"]),
          stop_date = new Date(eventdata["xaxis.range[1]"]);

        return update_graph(start_date, stop_date);
      });
    });
}
plot();
```

▼ Details

Copy/paste the code to your notebook and save the changes.

Note: you will need to provide the URL of your instance in the script.

For that change the line :

```
hateoas_url = "https://softinstXXX.host.vifib.net/erp5/web_site_module/default_wendelin_front/hateoas/";
```

The script will automatically find the resampled arrays based on the Data Product and plot the data usingplotly library.

Notebook

▼ Details

After writing the script and saving the changes click on the**Editable** checkbox to (uncheck it) on the left side panel to see the result.

Data Visualisation

▼ Details

If everything was done correctly you will see the plot of your data.