Agenda

▶ Details

- Create Notebook
- Visualise Chart Editor

Notebook Module

▼ Details

On Modules page click on **Notebooks**.

Note: On newer Wendelin versions you may find the **Notebooks** module in the category **Others** and no longer in the category **Big Data**.

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.

Edit the Notebook

▼ Details

Fill in the Title.

The notebook 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

▶ Details

```
%% md
<div class="plot_button_div" id="plot_button_div"></div>
<div class="plot_div" id="plot_div"></div>

%% fetch
css: static/css/main.88fa11bf.css
js: jio.js
js: ndarray_bundle.js
js: wendelin.js
js: https://cdn.plot.ly/plotly-latest.min.js
js: react.development.js
js: react-dom.development.js
js: static/js/main.8b730bc1.js

%% js
CHART_ID = "my_data_chart";
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"
});

editor_jio = jIO.createJIO({
  type: "indexeddb",
  database: "graph_editor"
});

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'),
  update = false,
  button_element = document.getElementById('plot_button_div'),
  displayed_data = [],
  displayed_layout = {},
  label_list = ["date", "humidity_min","humidity_mean","humidity_max", "pressure_min","pressure_mean","pressure_max", "temperature_min","temperature_mean", "temperature_max"],
  array_start_date,
  array_last_date,
  orig_start_date = new Date(),
  orig_stop_date = new Date(),
  frequency,
  array_id = "";
orig_start_date.setMonth(orig_start_date.getMonth() - 1);


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

function getFrequency(start_date, stop_date) {
  var ratio = screen.availWidth / 1024,
    interval = stop_date - start_date;
  if (stop_date === undefined) {
    return 30000;
  }
  if (interval >= 1000 * 60 * 60 * 24 * 100 * ratio) {
    return 30000;
  } else if (interval >= 1000 * 60 * 60 * 24 * 28 * ratio) {
    return 3000;
  } else if (interval >= 1000 * 60 * 60 * 24 * 7 * ratio) {
    return 300;
  } else {
    return 60;
  }
}

function getArrayId(frequency) {
  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) {
      return result.data.rows[0].id;
    });
}

function getData(start_date, stop_date){
  var start_index,
    stop_index;
  function unpack(rows, key) {
    return rows.map(function(row) { return row[key]; });
  }
  return new RSVP.Queue()
    .push(function() {
      return getFrequency();
    })
    .push(function(result) {
      frequency = result;
      return getArrayId(frequency);
    })
    .push(function(result) {
      array_id = result;
      return wendelin.getArrayRawSlice(gadget, array_id, 0, 1);
    })
    .push(function(result) {
      array_start_date = wendelin.convertFirstColToDate([[result.data[0]]])[0][0];
```

```
      return wendelin.getArrayLastRow(gadget, array_id);
    })
    .push(function(result) {
      var array_last_date = wendelin.convertFirstColToDate([[result.data[0]]])[0][0];
      stop_date = array_last_date;
      start_date = new Date(array_last_date.getTime());
      start_date.setMonth(start_date.getMonth() - 1);
      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) {
          graph_data=[];
          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){
          return wendelin.convertFirstColToDate(result)
        })
        .push(function (graph_data) {
          var humidity_trace = {
              x: unpack(graph_data, label_list.indexOf("date")),
              y: unpack(graph_data, label_list.indexOf("humidity_mean")),
          },
          pressure_trace = {
              x: unpack(graph_data, label_list.indexOf("date")),
              y: unpack(graph_data, label_list.indexOf("pressure_mean"))
          },
          temperature_trace = {
              x: unpack(graph_data, label_list.indexOf("date")),
              y: unpack(graph_data, label_list.indexOf("temperature_mean"))
          },
          data = [humidity_trace,pressure_trace,temperature_trace];
          return data
        });
    });
}

function update_graph(start_date, stop_date) {
  displayed_layout.xaxis.range = [start_date, stop_date];
  return getData(start_date, stop_date)
    .push(function (data) {
      for (var i = 0; i < data.length; i+=1) {
        displayed_data[i] = {...displayed_data[i], ...data[i]};
      }
      return Plotly.react(graph,
        displayed_data,
        displayed_layout
      );
    });
}

function renderEditor() {
 var props = {
   ref: function ref(element) {
    window.editor = element;
   },
   data: displayed_data,
   layout: displayed_layout,
   frames: [],
   url: ""
 };
  ReactDOM.render(React.createElement(EntryPoint.default.App, props, null), graph);
}

function editGraph(edit_btn, apply_btn) {
  return new RSVP.Queue()
    .push(function () {
      return Plotly.purge(graph);
    })
    .push(function () {
      return renderEditor();
    })
    .push(function () {
      button_element.replaceChild(apply_btn, edit_btn);
    });
}

function createGraph() {
  var start_date,
    stop_date;

  Plotly.newPlot(
    "plot_div",
    displayed_data,
    displayed_layout
  );
  graph.on('plotly_relayout', function(eventdata) {
  start_date = eventdata["xaxis.range[0]"]
  stop_date = eventdata["xaxis.range[1]"]

  if(start_date == undefined){
    start_date = orig_start_date
  }
  if(stop_date == undefined){
    stop_date = orig_stop_date
  }

  start_date = new Date(start_date);
  stop_date = new Date(stop_date);

  return update_graph(start_date, stop_date);
```

```
  });
}

function applyChangesToGraph(edit_btn, apply_btn) {
  var stored_data = [], temp;
  return new RSVP.Queue()
    .push(function () {
      return window.editor.getContent();
    })
    .push(function(result) {
      displayed_data = result.data;
      displayed_layout = result.layout;
      for (var i = 0; i < displayed_data.length; i+=1) {
        stored_data[i] = {};
        for (var key in displayed_data[i]) {
          if (key == "x") {
            stored_data[i][key] = []
          } else if (key == "y") {
            stored_data[i][key] = [];
          } else {
            stored_data[i][key] = displayed_data[i][key];
          }
        }
      }
      editor_jio.put(
        CHART_ID,
        {graph_model: {
          layout: result.layout,
          data: stored_data
          }
        }
      );
      return ReactDOM.unmountComponentAtNode(graph);
    })
    .push(function () {
      return createGraph();
    })
    .push(function(result) {
      button_element.replaceChild(edit_btn, apply_btn);
    });
}

function createButtons() {
  var plotly_apply_btn = document.createElement("BUTTON"),
    plotly_edit_btn = document.createElement("BUTTON");
  plotly_edit_btn.innerHTML = "Edit Chart";
  plotly_edit_btn.type = "button";
  plotly_apply_btn.innerHTML = "Apply";
  plotly_apply_btn.type = "button";
  plotly_edit_btn.addEventListener("click", editGraph.bind(null, plotly_edit_btn, plotly_apply_btn));
  plotly_apply_btn.addEventListener("click", applyChangesToGraph.bind(null, plotly_edit_btn, plotly_apply_btn));
  button_element.appendChild(plotly_edit_btn);
}

function getDisplayedLayout() {
  return {
    'title' :'Environmental Sensor Data',
    'xaxis': {
      'autorange': true,
      'rangeselector': selectorOptions,
    },
    'yaxis': {
      'autorange': true
    }
  };
}

function getDisplayedDataTemplate() {
    var humidity_trace = {
      type: "scattergl",
      mode: "markers",
      name: 'Humidity(%)',
      x: [],
      y: [],
      line: {color: '#17BECF'}
    }, pressure_trace = {
      type: "scattergl",
      mode: "markers",
      name: 'Pressure(Pa)',
      x: [],
      y: [],
      line: {color: '#7F7F7F'}
    }, temperature_trace = {
      type: "scattergl",
      mode: "markers",
      name: 'Temperature(C°)',
      x: [],
      y: [],
      line: {color: '#FF0000'}
    };
    return [humidity_trace,pressure_trace,temperature_trace];
}

function plot() {
  var start_date = orig_start_date,
    stop_date = orig_stop_date;
  return editor_jio.get(CHART_ID)
    .push(function(result) {
      displayed_layout = result.graph_model.layout;
      displayed_data = result.graph_model.data;
      if (displayed_data == undefined) {
        displayed_data = getDisplayedDataTemplate();
      }
```

```
      if (displayed_layout == undefined) {
        displayed_layout = getDisplayedLayout();
      }
      return getData(start_date, stop_date);
    }, function(error) {
      displayed_layout = getDisplayedLayout();
      displayed_data = getDisplayedDataTemplate();
      return getData(start_date, stop_date);
    })
    .push(function (data) {
      for (var i = 0; i < data.length; i+=1) {
        displayed_data[i].x = data[i].x;
        displayed_data[i].y = data[i].y;
      }
      Plotly.newPlot(
        "plot_div",
        displayed_data,
        displayed_layout
      );
      graph.on('plotly_relayout', function(eventdata) {
        start_date = eventdata["xaxis.range[0]"]
        stop_date = eventdata["xaxis.range[1]"]

        if(start_date == undefined){
          start_date = orig_start_date
        }
        if(stop_date == undefined){
          stop_date = orig_stop_date
        }

        start_date = new Date(start_date);
        stop_date = new Date(stop_date);

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

function main() {
  return new RSVP.Queue()
    .push(function() {
      return plot();
    })
    .push(function() {
      return createButtons();
    });
}

main();
```

Data Visualisation

▼ Details

If everything was done correctly you will see the plotly chart editor for your data.

Chart Editor

▼ Details

Now let's edit the chart to have a better data visualization. Click on the chart editor "Edit Chart" button to display the edit menu and check all the available customizations.

Chart Editor

▼ Details

There are several aspects to edit like the graph structure, the colors and styles, add annotations or directly edit the JSON of the graph. Let's change the trace type for one of the values. Let's use a bars visualization for the humidity trace.

Chart Editor

▼ Details

After editing something in the graph, you can see the preview to check how it looks.

Chart Editor

▼ Details

We can change the rest of the trace types to get a better visualization of the data. Let's use Area type for Pressure and Bars for Temperature. See the preview result.

Chart Editor

▼ Details

Once you are done with the customization, click on the chart "Apply" button to persist your changes. See the final chart view.