Jsonnet Library Code Glimpse

```
  /* Creates an SQL target */
  target(
    rawSql,
    format='time_series',
    alias=null,
  ):: {
    [if alias != null then 'alias']: alias,
    format: format,
    [if rawSql != null then 'rawSql']: |||
          ... RAW SQL ...
          |||,
    groupBy:
      if group_tags != null then
        [{ type: 'tag', params: [tag_name] } for tag_name in group_tags] +
        [{ type: 'fill', params: [fill] }]
  },
} + { addMappings(mappings):: std.foldl(function(p, m) p.addMapping(m), mappings,
self),}
```

Exported Dashboard Json


```
{
    "__inputs": [],
    "__requires": [],
    "annotations": {
        "list": []
    },
    "editable": true,
    "gnetId": null,
    "graphTooltip": 0,
    "hideControls": false,
    "id": null,
    "links": [],
    "panels": [{
        "datasource": {},
        "fieldConfig": {
            "defaults": {
                "links": [],
                "mappings": [],
                "max": 100,
                "min": 0,
                "thresholds": {
                    "mode": "absolute",
                    "steps": []
                },
                "unit": "percent"
            }
        },
        "gridPos": {
            "h": 9,
            "w": 12,
            "x": 0,
            "y": 0
        },
        "id": 1,
        "links": [],
        "options": {
            "reduceOptions": {
                "calcs": [
                    "mean"
                ],
                "fields": "",
                "values": false
            },
            "showThresholdLabels": false,
            "showThresholdMarkers": true
        },
        "pluginVersion": "7.3.4",
        "targets": [],
        "title": "Panel 2",
        "transparent": false,
        "type": "gauge"
    }],
    "schemaVersion": 26,
    "style": "dark",
    "tags": [],
    "templating": {
        "list": []
    },
    "time": {
        "from": "now-6h",
        "to": "now"
    },
    "timepicker": {},
    "timezone": "",
    "title": "example",
    "uid": "GAhKcCNVk",
    "version": 2
}
```

Deserialization Challenges

```
"legend": {
  "show": false
},
```

```
"legend": {
  "avg": false,
  "current": false,
  "max": false,
  "min": false,
  "show": true,
  "total": false,
  "values": false
},
```

```
"tooltip": {
  "show": true,
  "showHistogram": false
},
```

```
"tooltip": {
  "shared": true,
  "sort": 0,
  "value_type": "individual"
},
```

Deserialization Challenges

```
"annotations": {
  "list": [
    {
      "builtIn": 1,
      "datasource": "-- Grafana --",
      "enable": true,
      "hide": true,
      "iconColor": "rgba(0, 211, 255, 1)",
      "name": "Annotations & Alerts",
      "type": "dashboard"
    }
  ]
```

```
public class Annotations {
  @SerializedName("list")
  List<example.model.grafana.List> list = new ArrayList<>();

  public List<example.model.grafana.List> getList() { return list; }

  public void setList(List<example.model.grafana.List> data) {
    list = data; }

}
```

Clean Jsonnet Example

```
dashboard.new(
  title = 'SQL heatmap',
  tags = ['SQL'])
.addRows([
  row.new(title = r.row_title)
  .addPanels(
   [
      heatmap.new(
        title = metric.name,
      )
      .addTarget(
        grafana.sql.target(
          query = |||
          ... RAW SQL ...
          |||
))
```

Alternative JSON Template Languages

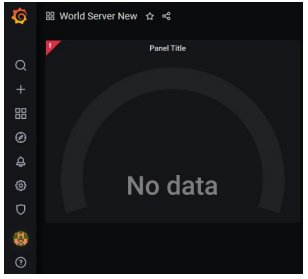
- Generic:

```
const template = parse("{{foo:bar}}");
```

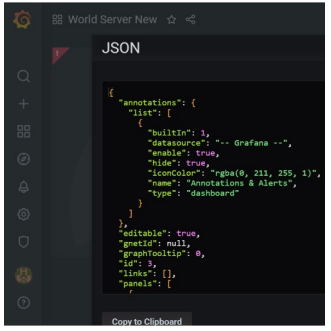
- YAML based:

```
- name: overview
  dashboard:
    title: overview dashboard
    time_options: [1h]
    refresh_intervals: [5m]
    templates:
      - template-name:
            metric-prefix: '{metric-prefix}'
    time:
      from: now-12h
      to: now
    rows:
      - row-name
```

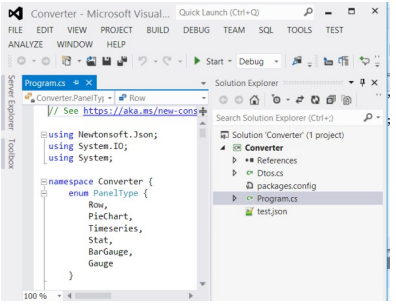
Grafana Dashboard Jsonnet Workflow with a Custom Tool



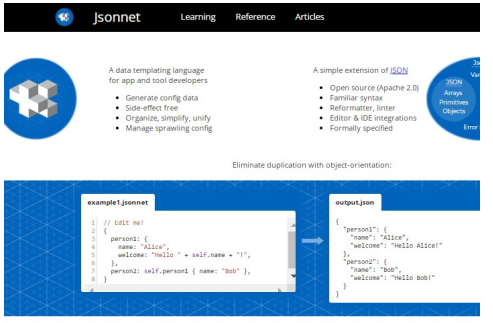
1



2



3

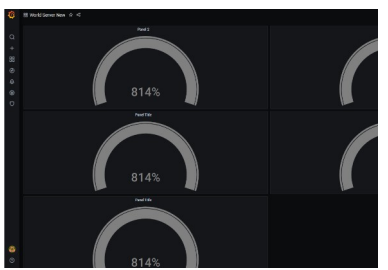


4

1) dashboard crafted manually
2) dashboard export
3) read JSON,write jsonnet file
4) convert jsonnet to JSON
5) Import dashboard into Grafana
   3a) https://json2csharp.com

3a

5



Auto-generated Jsonnet Sample

```
grafana.dashboard.new(
  'World Server New',
  tags=[],
  refresh='5s',
  time_from='now-1h',
  time_to='now',
  timepicker=grafana.timepicker.new(
    refresh_intervals=['30s', '1m', '5m', '15m', '30m', '1h', '2h', '1d', '2d', '7d'],
  ),
).addPanel(
  gridPos={h: 9, w: 12, x: 0, y: 0},
  panel=grafana.gaugePanel.new(title='Panel 2',pluginVersion='7.3.4',
  thresholdsMode='absolute', datasource=prometheus)
  .addTarget(
    prometheus.target(
      expr ='',
      hide='False',
    )
  )
)
```

Perl Catalyst Template Language

```
  # message assignment
  if page.message_ids and page.message_ids.size( ) > 0;
    foreach message_id in page.message_ids;
      page.messages.push(page.message_texts.${message_id});
    end;
  end;
  # building link assignment
  linkvars = [ { login    = id.login }
               { session  = session_id }
               { obj_cust = sel.obj_cust }

           ]
arg = []; foreach pair in linkvars; arg.push(pair.keys.0 _ '=' _ pair.values.0); end;

[% # Add a link to delete a book %]
  <a href="[% c.uri_for(c.controller.action_for('delete'), [book.id])%]">Delete</a>
</td>
```