JSON-Region-Plugin

Ein Plugin zur dynamischen Generierung
von APEX-UI-Items für die Ein-/Ausgabe
von Attributen von JSON-Feldern

Uwe Simon Database Consulting
2024-04-24
APEX-Connect-2024

Über Mich

Name: Uwe Simon

Oracle-DB: seit 1992 mit Oracle 5

- Datenmodellierung
- Performanz-Tuning
- Migrationen
- Proof of Concepts
- Performanz-Tests

APEX: 1998 mit OAS/OWS, HTML-DB, APEX

Entwicklung: SQL, PL/SQL, C++, Javascript, Java, ...

Weitere DBs: MySQL, PostgreSQL

Seit 2023 Freiberuflich

Agenda

- Idee des Plugins
- JSON-Schema und APEX-UI
- Komplexe JSON-Schema und APEX-UI
- Customizing der APEX-UI
- Oracle 23c
- Sonstiges

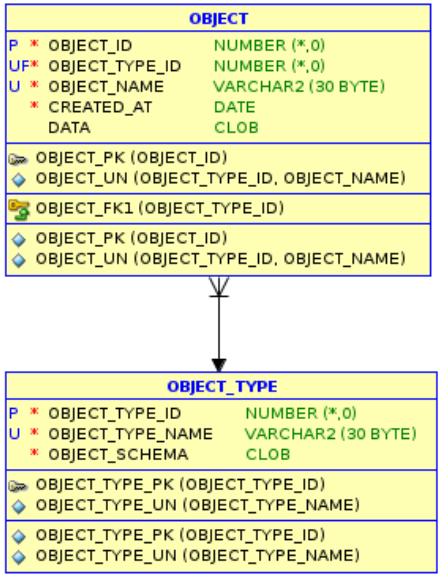
- Idee des Plugins
- JSON-Schema und APEX-UI
- Komplexe JSON-Schema und APEX-UI
- Customizing der APEX-UI
- Oracle 23c
- Sonstiges

Idee

- JSON-Felder werden in immer mehr Datenbank-Schemas genutzt
- APEX liefert keine Out-Of-The-Box-Lösung für die Ein-/Ausgabe der Attribute von JSON-Feldern
- Idee:
  - Aus einem JSON-Schema dynamisch zur Laufzeit eine APEX-UI generieren,
  - Je Datensatz ggf. je nach „Datensatztyp“ unterschiedliche JSON-Schema
  - Keine Modifikationen am APEX-Code bei Änderungen des JSON-Schema
  - Anpassungsmöglichkeiten des UI-Layouts zur Unterstützung von weiteren APEX-Item-Typen durch erweiterte Attribute im JSON-Schema

Einsatzmöglichkeiten des Plugins

- Konfigurierbare Workflows:
  Die Daten für den Workflow sind in JSON-Feldern abgelegt
- Konfigurierbare Asset-Management-Systeme:
  Attribute die vom Assettyp abhängen liegen in JSON-Feldern
- Formular-Tools:
  Formularstruktur liegt im JSON-Schema und Formulardaten liegen in JSON-Feldern
- Umfrage-Tools:
  Fragen liegen in JSON-Schema und Daten dann in JSON-Feldern
- Durch den Kunden anpassbare Anwendungen:
  Customizing erfolgt über JSON-Felder.
- ...



- Idee des Plugins
- JSON-Schema und APEX-UI
- Komplexe JSON-Schema und APEX-UI
- Customizing der APEX-UI
- Oracle 23c
- Sonstiges

JSON-Schema in Kürze

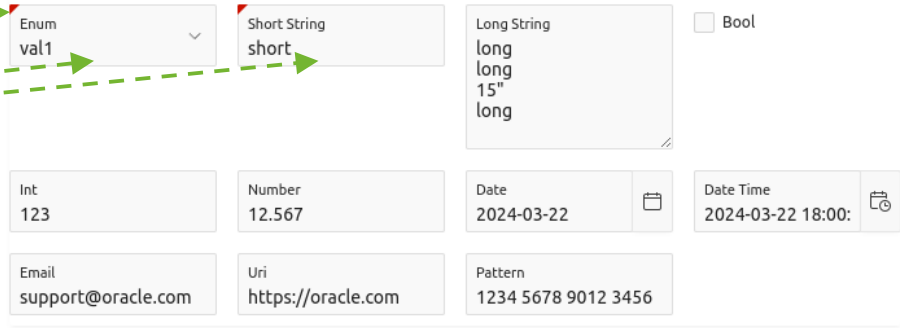
- Dokumentation zu JSON-Schema befindet sich unter https://json-schema.org/
- JSON-Schema wird zur Beschreibung und Validierung von JSON-Daten genutzt
  - Für jedes Attribut wird dabei Mussfeld (“required“), Datentyp (“type“) und Format (“format“), Aufzählung (“enum“) bzw. Muster (“pattern“) festgelegt.
  - Ein Attribut kann wiederum ein JSON-Schema bzw. ein Array sein (“type“: “object“ bzw, “array“) sein.
- Oracle23c unterstützt JSON-Schema-Validierungen auch als CHECK-Constraint einer CLOB/JSON-Spalte.

```
{
  "type": "object",
  "required": ["enum", "short_string"],
  "properties": {
    "enum":         { "type": "string", "enum": [ "val1", "val2" ]},
    "short_string": { "type": "string" },
    "long_string":  { "type": "string", "maxLength": 400},
    "bool":         { "type": "boolean"},
    "int":          { "type": "integer" },
    "number":       { "type": "number" },
    "date":         { "type": "string", "format": "date"},
    "date_time":    { "type": "string", "format": "date-time"},
    "email":        { "type": "string", "format": "email"},
    "uri":          { "type": "string", "format": "uri"},
    "pattern":      { "type": "string", "pattern": "[0-9]{4}( [0-9]{4}){3}"}
  }
}
```

Was liegt also näher, als auch JSON-Schema für die APEX-UI zu nutzen.

JSON-Schema, JSON-Daten und APEX-UI

```
{
  "type": "object",
  "required": ["enum", "short_string"],
  "properties": {
    "enum":         { "type": "string", "enum": [ "val1", "val2" ]},
    "short_string": { "type": "string" },
    "long_string":  { "type": "string", "maxLength": 400},
    "bool":         { "type": "boolean"},
    "int":          { "type": "integer" },
    "number":       { "type": "number" },
    "date":         { "type": "string", "format": "date"},
    "date_time":    { "type": "string", "format": "date-time"},
    "email":        { "type": "string", "format": "email"},
    "uri":          { "type": "string", "format": "uri"},
    "pattern":      { "type": "string", "pattern": "[0-9]{4}( [0-9]{4}){3}"}
  }
}
```



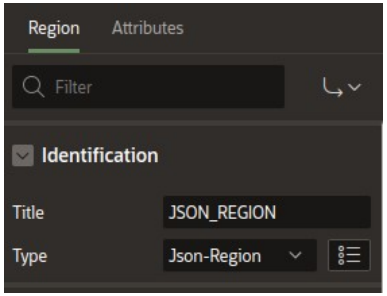

```
{
  "enum":"val1",
  "short_string":"short",
  "long_string":"long\nlong\n15\"\nlong",
  "bool":false,
  "int":123,
  "number":12.567,
  "date":"2024-03-22",
  "date_time":"2024-03-22T18:00:00",
  "email":"support@oracle.com",
  "uri":"https://oracle.com",
  "pattern":"1234 5678 9012 3456"
}
```

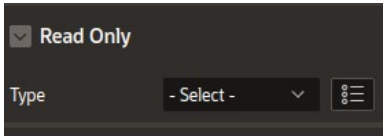
Demo

- Ich hab da mal was vorbereitet
- Die kleine Demo nutzt ein sehr einfaches Datenmodell
  - Tabelle für „generische“ Objekte mit einer JSON-Spalte (CLOB/JSON)
  - Tabelle mit Objekttypen und Spalte mit dem JSON-Schema

Konfiguration des Plugins im Dialog-Editor

- Simple Konfiguration
  - JSON-Item
  - Source Static/SQL-Query
  - Statisches JSON-Schema bzw. SQL-Query, die eine Zeile mit dem JSON-Schema zurückliefert
- Weitere Konfiguration
  - Zusätzliche Attribute im JSON behalten
  - Für Subobjekte Überschriften generieren
  - Das JSON-Feld automatisch nicht anzeigen
  - Leere Felder aus dem JSON entfernen
  - Das Readonly-Attribut für die Region wird übernommen





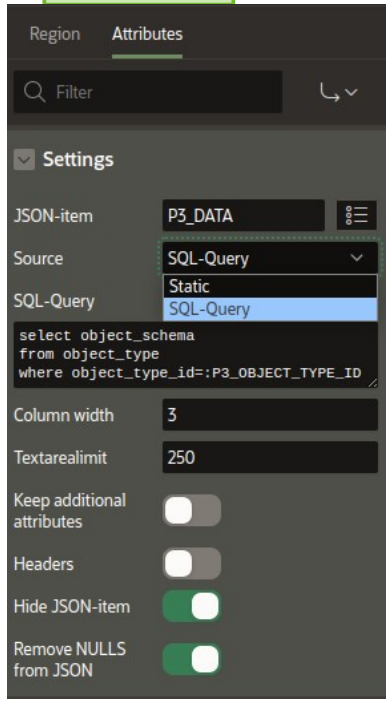
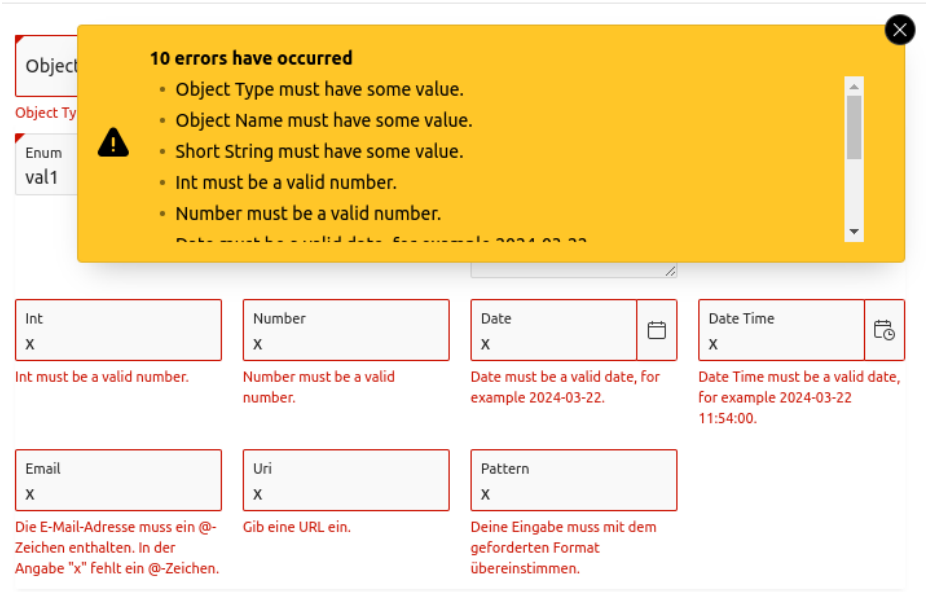


Abbildung JSON-Schema nach APEX-UI

- Die Attribute werden in der gleichen Reihenfolge wie im JSON-Schema angezeigt.
- Je nach “type“/“format“ wird per Default ein passender „APEX-Item-Typ“ für die Ein-/Ausgabe genutzt
  - string — Text Field bzw. Textarea (je nach Länge)
  - integer/number — Numerisches Feld
  - boolean — Checkbox
  - date/date-time — Date-Picker/ Date-Picker+Time
  - enum — Pulldown
  - email — Text Field mit Subtyp Email
  - uri — Text Field mit Subtype URL
  - …
- Anzeigename des APEX-Items ist per Default der Name des Attributes
  (1. Zeichen je Wort groß _- ersetzen, ... wie Default-Title im Page-Designer)

Feldvalidierungen und Fehlermeldungen

- Unterstützte Validierungen
  - Integer, Number
  - Date, Date-Time
  - Regex-Pattern
  - Email-Adresse
  - URI
  - Minimum, Maximum
  - Maximale Länge
- Standard-Fehlermeldungen bei Validierungsfehlern



Idee des Plugins

JSON-Schema und APEX-UI

Komplexe JSON-Schema und APEX-UI

Customizing der APEX-UI

Oracle 23c

Sonstiges

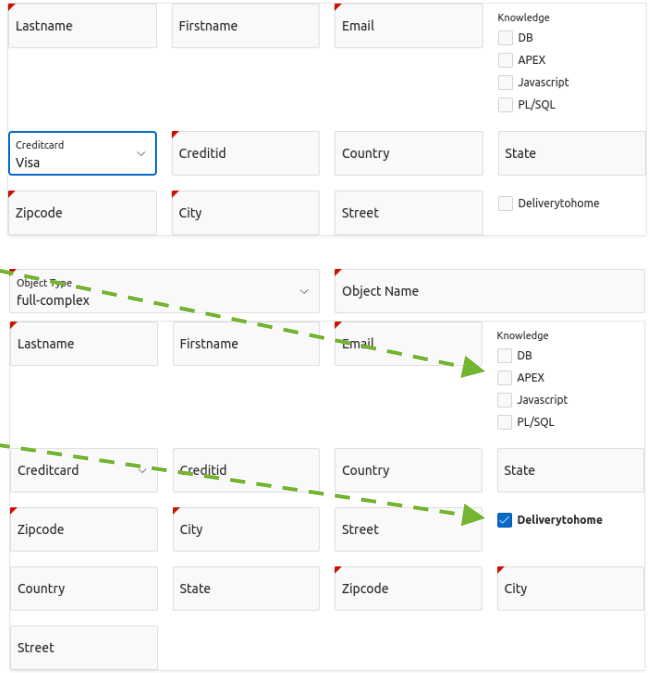
Komplexe JSON-Schema Attribute

- Konstante Werte
  - “const“: “constant Value“
- Rekursive
  - “type“: “object“
  - “type“: “array“, “items“: […] (Unterstützung nur für “multiselect“/“checkbox-group“)
- Schema-Referenzen
  - “$ref“: “#/…“ (Unterstützung nur für Referenzen im gleichen JSON-Schema)
    Vermeiden von Redundanzen in der Schema-Definition, z.B. wenn es mehrere Anschriften gibt
- Conditional Schema
  - “dependentRequired“
    Feld wird Mussfeld, wenn ein anderes Feld nicht leer ist
  - “dependentSchema“
    Subschema muss eingegeben werden, wenn ein anderes Feld nicht leer ist (Kreditkartentyp, Nr.)
  - “if“, “then“, “else“
    Je nach Wert eines Feldes, weitere Eingabefelder (z.B. bei „abweichende Rechnungsanschrift“, 2. Anschrift)

Komplexe JSON-Schema Attribute

```
{
  "type": "object",
  "required": ["lastname", "email"],
  "dependentRequired":{
    "creditcard": ["creditid"],
    "creditid":   ["creditcard"]
  },
  "properties": {
    "lastname":    {"type": "string", "maxLength": 30},
    "firstname":   {"type": "string", "maxLength": 30},
    "email":       {"type": "string", "format": "email"},
    "knowledge":   {"type":"array","items":{"type":"string", "enum":["DB","APEX","Javascript","PL/SQL"]}},
    "creditcard":  {"type": "string", "enum":["Visa", "Mastercard", "Amex", "Diners"]},
    "creditid":    {"$ref": "#/$defs/cardid"},
    "office_address":     {"$ref": "#/$defs/address"},
    "deliverytohome":     { "type": "boolean"}
  },
  "if": {
      "properties": {
        "deliverytohome": { "const": true}
      }
  },
  "then": {
     "properties": {
       "home_address": {"$ref": "#/$defs/address"}
     }
  },
  "$defs":{
      "name": {"type": "string", "maxLength": 30},
      "address": {
        "type": "object",
        "required": ["zipcode", "city"],
        "properties": {
           "country": {"type": "string"},
           "state":   {"type": "string"},
           "zipcode": {"type": "string"},
           "city":    {"type": "string"},
           "street":  {"type": "string"}
         }
       },
      "cardid": {"type": "string", "pattern": "[0-9]{4}( [0-9]{4}){3}"}
  }
}
```

```
{
 "lastname":"Simon",
 "firstname":"Uwe",
 "email":"usdbc@magenta.de",
 "knowledge":["DB","APEX","Javascript","PL/SQL"],
 "creditcard":"Visa",
 "creditid":"1234 5678 9012 3456",
 "office_address":{
   "country":"D","state":"NRW","zipcode":"50000","city":"City1","street":"Street1"
},
"deliverytohome":true,
"home_address":{
  "country":"D","state":"NRW","zipcode":"50000","city":"City2","street":"Street2"}
}
```



Idee des Plugins

JSON-Schema und APEX-UI

Komplexe JSON-Schema und APEX-UI

Customizing der APEX-UI

Oracle 23c

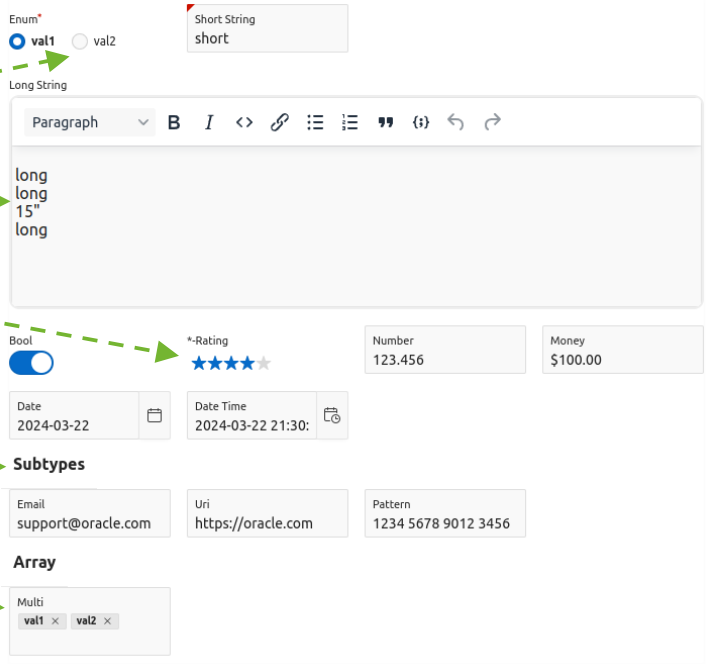
Sonstiges

Customizing der APEX-UI ...

- APEX hat in der UI für einige Datentypen mehrere Darstellungsformen
- APEX-spezifische Konfiguration unter “apex“: {…}
- Attribute “itemtype“ zur Konfiguration des APEX-UI-Items
  - “itemtype": "starrating" Integer-Feld als Starrating
  - "itemtype": "switch" Boolean-Feld als Switch
  - "itemtype": "richtext" Für lange Strings Richtext-Editor
  - "itemtype": "combobox" Für Multiselect Combobox (mit „Chips“, ab APEX23.2)
- Weitere Attribute unter „apex“
  - „label“: “Text“ Text als Label für das Feld
  - "newRow": true Neue Zeile vor dem Feld,
  - "textBefore": "Text" statische Text vor dem Feld
  - "lines": 10 Anzahl der Zeilen bei Textarea/Richtext-Editor
  - "colSpan": 6 Breite des Feldes (1-12)
  - “readonly“: true Feld ist nur zur Anzeige
  - “direction“: “horizontal“ Radio/Checkbox horizontal

… Customizing der APEX-UI

```
{
  "type": "object",
  "required": ["enum", "short_string"],
  "properties": {
    "enum":         { "type": "string", "enum": [ "val1", "val2" ],
                      "apex": {"itemtype": "radio", "direction": "horizontal"}},
    "short_string": { "type": "string" },
    "long_string":  { "type": "string", "maxLength": 400,
                      "apex": {"itemtype": "richtext", "lines": 4, "colSpan": 12}},
    "bool":         { "type": "boolean",
                      "apex": {"itemtype": "switch"}},
    "int":          { "type": "integer", "maximum": 5,
                      "apex": {"itemtype": "starrating", "label": "*-Rating"}},
    "number":       { "type": "number" },
    "money":        { "type": "number",
                      "apex": {"format": "currency"}},
    "date":         { "type": "string", "format": "date"},
    "date_time":    { "type": "string", "format": "date-time"},
    "email":        { "type": "string", "format": "email",
                      "apex": {"textBefore": "Subtypes"}},
    "uri":          { "type": "string", "format": "uri"},
    "pattern":      { "type": "string", "pattern": "[0-9]{4}( [0-9]{4}){3}"},
    "multi":  { "type": "array",
                "items": { "type": "string", "enum":  ["val1",  "val2"]},
                "apex":  {"itemtype": "combobox", "textBefore": "Array"}
              }
    }
}
```



- Idee des Plugins
- JSON-Schema und APEX-UI
- Komplexe JSON-Schema und APEX-UI
- Customizing der APEX-UI
- Oracle 23c
- Sonstiges

Schema aus JSON-Validierung

- Mir Oracle23c kann man im Check-Constraint einer JSON-Spalte auch das Schema angeben.
- Was liegt da näher, als dieses auch für die APEX-UI zu nutzen
- Achtung:
  - Leider unterstützt Oracle nicht die kompletten Möglichkeiten des JSON-Schema z.B. wird “$ref“: “…“ ignoriert
  - Es gibt Oracle-spezifische Erweiterungen z.B. “extendedType“: “...“ (Wird von Plugin unterstützt)

```
CREATE TABLE object23c(
  object_id      INTEGER GENERATED BY DEFAULT ON NULL AS IDENTITY,
  object_name    VARCHAR2(30) NOT NULL,
  data           JSON,
  CONSTRAINT object23c_pk PRIMARY KEY (object_id)
);


ALTER TABLE object23c ADD CONSTRAINT object23c_ck1
  CHECK (data IS JSON VALIDATE q'[{
    "type"       : "object",
    "properties" : {
      "fruit"    : {"type"      : "string",
                    "minLength" : 1,
                    "maxLength" : 10},
      "quantity" : {"type"      : "number",
                    "minimum"   : 0,
                    "maximum"   : 100},
      "orderdate": {"type": "string",
                    "default": "now",
                    "format": "date"}
      },
      "required"   : ["fruit", "quantity"]
    }]'
  );
```

Idee des Plugins

JSON-Schema und APEX-UI

Komplexe JSON-Schema und APEX-UI

Customizing der APEX-UI

Oracle 23c

Sonstiges

Sonstiges

- JSON in CLOB

  Für den CLOB den Check-Constraint IS JSON(STRICT) nutzen

- Häufig müssen in einem JSON-Schema “enum“-Attribute mit Lookup-Tabellen synchron gehalten werden

  Lösung: Statement-Trigger auf Lookup-Tabelle

```
1  ALTER TABLE object ADD CONSTRAINT object_ck_1 check (data IS JSON(STRICT));
2
3
4  CREATE TABLE hotel_feature(
5    feature VARCHAR2(100) NOT NULL,
6    CONSTRAINT hotel_feature_pk PRIMARY KEY(feature)
7  );
8
9  CREATE OR REPLACE TRIGGER hotel_feature_tr
10   AFTER INSERT OR UPDATE OR DELETE ON hotel_feature
11 DECLARE enum VARCHAR2(32000);
12 BEGIN
13   SELECT listagg('"'||REPLACE(feature,'"', '\"')||'"', ',') WITHIN GROUP (ORDER BY feature)
14   INTO enum
15   FROM hotel_feature;
16   UPDATE object_type SET object_schema =
17     json_mergepatch(object_schema, '{"properties": {"features": {"items": {"enum":['||enum||']}}}}')
18   WHERE object_type_name='Hotel';
19 END;
20 /
```

Bekannte Nutzungen des Plugins

- USA: Eine der größten Kommunal-Behörden
  Workflow mit komplexen auf JSON basierenden Formularen
  Go-Live April/May

- Indien:
  PoC: Generische Workflows

- Deutschland:
  PoC: keine Info erhalten

Weitere Ideen

- Formatierung von JSON-Spalten in Listen/Reports mittels JSON-Path (Rel 0.9.0)

```
"apex": {
  "display": { "list1": "Model: #$.model#, Vendor: #$.vendor#" }
}
```

- Images (Rel 0.9.0)
  - JSON-Schema unterstützt Strings mit base64 codierten Daten.

```
{
    "type": "string", "contentEncoding": "base64", "contentMediaType": "image/png"
}
```

- JSON-Relational-Duality

  UI aus der Oracle23c JSON-Duality-View generieren

- Weitere Unterstützung von “array“

  Analog Interactive Grid

Das Ende

Fragen und Antworten

APEX:	https://apex.world (JSON-Region)
Github:	https://github.com/simonuwe/oracle-apex-json-region
Email:	usdbc@magenta.de
LinkedIn:	https://www.linkedin.com/in/uwe-simon-cologne/