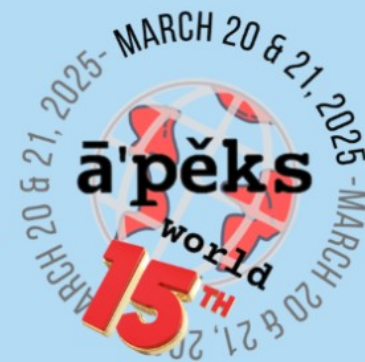




Uwe Simon

Dynamic APEX UI for JSON data with the JSON Region plugin

Thu, 20 March, 10:55 | Bach 1 & 2

‘ReeHorst’, Ede

#APEXWorld2025

About Me

- Name: Uwe Simon
- Oracle-DB: Oracle-DB experience since 1992 starting with Oracle 5
Database modelling
Performance tuning of huge multi 100TB databases
Database analysis of “unknown” mission critical databases during major incident procedures
DB-migrations
Proof of Concepts
- APEX: Started 1998 with OAS/OWS, HTML-DB, APEX ...24.2
- Development: SQL, PL/SQL, C++, JavaScript, Java, HTML/CSS, ...
- Other DBs: DB2, MySQL, PostgreSQL
- Located in Cologne
during the last years partial on-site in The Netherlands, Czech Republic, India
- In partial retirement and Freelancer since 2023



Agenda

History of the plug-in

Idea of the plug-in

JSON-Region-Plugin vs. JSON-source in APEX 24.2

JSON-Schema and APEX-UI

Complex JSON-schema and APEX-UI

Customizing the APEX-UI

Oracle 23ai

Experience during development

Miscellaneous

History of the plug-in

- The development of the JSON-Region-Plugin started October 2023
- First presentation of the plug-in at APEX-connect2024 in Germany,
- Michael Hichwa (Oracle) gave the feedback that my plug-in is great and that “Oracle definitely needs such a solution for JSON in APEX”
- Oracle introduced a solution using new JSON/Duality-source in APEX 24.2, implementing parts of the plug-ins functions
- The plug-in is continuously updated with new features for APEX 20.2-24.2



History of the plug-in

Idea of the plug-in



JSON-Region-Plugin vs. JSON-source in APEX 24.2

JSON-Schema and APEX-UI

Complex JSON-schema and APEX-UI

Customizing the APEX-UI

Oracle 23ai

Experience during development

Miscellaneous

JSON-Schema in nutshell

- Documentation of JSON-Schema could be found at <https://json-schema.org/>
- JSON-schema is a description for the structure of JSON-data and could be used for the validation of JSON-data
 - A JSON-schema is an object or an array (“type”: “object“, “type”: “array“) with attributes (“properties“, “items”)
 - For each attribute it defines it’s data type (“type“) and format (“format“), if it’s “mandatory” (“required“), if it’s an enumeration (“enum“), must match a pattern (“pattern“),
 - An attribute could be an object or an array, .
- Oracle23ai supports JSON-schema-validations on a CLOB/JSON-columns and collection tables/views, duality-views

```
{
  "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}" }
  }
}
```

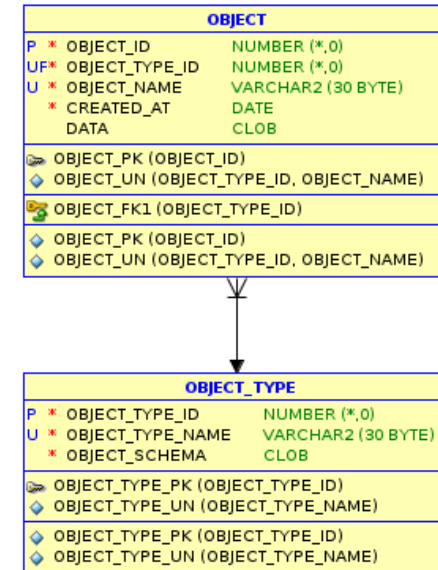
So what could be more obvious than using JSON Schema for the APEX UI.

Idea of the plug-in

- APEX <24.2 offers no Out-Of-The-Box-solution for input/output of JSON-data
- The plug-in generates at runtime a dynamic APEX-UI based on a JSON-schema
- Support of all major APEX-item-types
- Use of variable JSON-schemas for each row of a table
- Modifying the JSON-schema changes the UI immediately without any modification in the APEX-code
- Transparent for user, “look like APEX-UI”, so same error-handling, same items, ...
- Customization of the UI with JSON-schema-extension by new property **“apex”: { ... }**
- Support of JSON-schema-references **“\$ref”: “...”**
- Support of conditional JSON-schema for dynamic UIs
“dependenRequired”: {},
“dependendSchemas”: {},
“\$if”: {}, “\$then”: {}, “\$else”: {}

Possible usecases of the plug-in

- Configurable workflows:
The data for the workflow is stored in JSON-columns
- Configurable Asset-Management-Systems:
- Attributes depending on asset types are stored in JSON-columns
- Form-tools:
Form-structure is a JSON-schema and form-data is stored in JSON-columns
- Polling-tools:
Questions and list of answers are stored in a JSON-schema and the data is stored in a JSON-column
- APEX-applications with customization by customer:
Customizing of the application is possible with JSON-columns.





History of the plug-in

Idea of the plug-in

JSON-Region-Plugin vs. JSON-source in APEX 24.2

JSON-Schema and APEX-UI

Complex JSON-schema and APEX-UI

Customizing the APEX-UI

Oracle 23ai

Experience during development

Miscellaneous

Plug-in vs. JSON-support of APEX-24.2

Feature	Plug-in	APEX 24.2
VARCHAR2/CLOB/JSON column	✓	✓
collection-table	✓	✓
collection-view	✓	✗
JSON-duality-view	✓	✓
Fixed JSON-schema	✓	✓ JSON-source
Variable JSON-schema	✓	✗
Evaluation of JSON-schema	At runtime	In page-designer
JSON-schema from DB (23ai)	✓	✓✗ (Duality-source only)
Dynamic UI	✓	✗
Schema references	✓	✗
Conditional JSON-schema	✓	✗
Interactive grid/report	✓✗ with JSON-Item-Plug-in	✓



History of the plug-in

Idea of the plug-in

JSON-Region-Plugin vs. JSON-source in APEX 24.2

JSON-Schema and APEX-UI

Complex JSON-schema and APEX-UI

Customizing the APEX-UI

Oracle 23ai

Experience during development

Miscellaneous

JSON-schema, JSON-data and 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
long
15'
long

Bool

Int
123

Number
12.567

Date
2024-03-22

Date Time
2024-03-22 18:00

Email
support@oracle.com

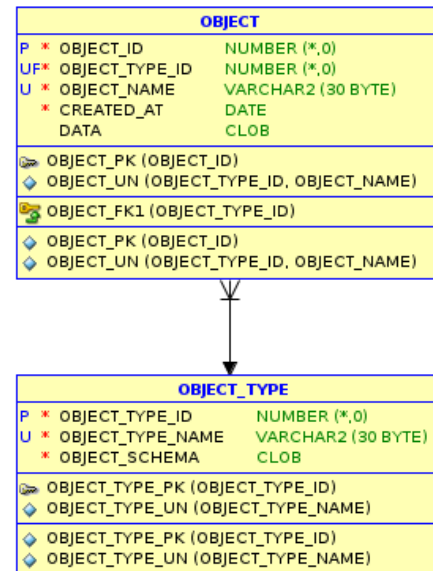
Uri
https://oracle.com

Pattern
1234 5678 9012 3456

```
{
  "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

- A demo says more than 1000 slides
- My demo uses a simple data-model
 - Table “OBJECT” for „generic“ objects in JSON-column “DATA” (CLOB/JSON)
 - Table “OBJECT_TYPE” containing the valid object-types with column “OBJECT_SCHEMA” for the JSON-schema of each type
 - JSON-collection-table/view “TABLE23AI”, “VIEW23AI”
 - JSON-duality-view “JSON23AI”
 - APEX-24.2 JSON-source, Duality-source
 - When time left: 2 applications using the plug-in



Plug-in configuration in APEX-dialog-editor

- Simple configuration
 - JSON-Item
 - Source for JSON-schema
 - Static JSON-schema
 - SQL-Query, returning the JSON-schema
 - Generated JSON-schema generated based on JSON-data
- Other configurations
 - UI: column width, limit when "textarea" is used, item-template to use
 - Generate headers for sub-objects
 - Hide the page-item containing the JSON-data
 - Keep additional attribute in JSON
 - Remove empty/null properties from JSON-data
 - Read-only-attribute of region is used for all items in the region

This screenshot shows the 'Region' tab of the APEX Dialog Editor. The 'Identification' section is expanded, showing the 'Title' field set to 'JSON_REGION' and the 'Type' dropdown set to 'Json-Region'. Below this, the 'Read Only' section is also expanded, showing a 'Type' dropdown set to '- Select -'.

This screenshot shows the 'Attributes' tab of the APEX Dialog Editor. The 'Settings' section is expanded, showing various configuration options: 'JSON-item' is set to 'P3_DATA', 'Source' is set to 'Static', and 'Static Schema' is empty. Below these, 'Merge Schema' is a toggle switch that is currently off. 'SQL-Query for referenced JSON-schema' is also empty. At the bottom, there are three input fields: 'Column Width' set to '3', 'Textarealimit' set to '250', and 'Template' set to 'Header Floating'. Finally, there are three more toggle switches: 'Keep additional attributes' (off), 'Headers' (on), and 'Remove NULLs from JSON' (on).

Transformation JSON-Schema to APEX-UI

- The attributes are displayed in the same order as defined in the JSON-schema.
- Depending on "type"/"format"/"pattern" in the JSON-schema a matching „APEX-Item-Type“ is used for input/output.
 - string "Text Field"/"Text, "Image" when "contentEncoding": "base64" and "contentType" are defined
 - integer/number "Number field"
 - boolean "Checkbox",
 - date/date-time/time "Date/Date-Picker/Time-Picker"
 - enum "Selectlist"
 - email "Text Field" with Subtype "Email"
 - uri "Text Field" with Subtype "URL"
 - ipv4/v6/UUID "Text Field" with pattern"
 - ...
- Display name of APEX-items is per default the attribute name
(1. char capital, replace _ by " ", same behaviour as for default-titles in page-designer)

item validation and error messages

- Supported validations
 - mandatory
 - integer, number, currency
 - date, date-time, time
 - regexp-pattern
 - email-address, URI
 - ipv4/v6 address, UUID
 - min, max
 - string length
 - Enum (list of values)
- Standard system error messages on validation errors

Object

Object Type
full-simple

Enum
val1

☐ Bool

Int
Int must be a valid number.

Number
Number must be a valid number.

Date
Date must be a valid date, for example 19.04.2024.

Date Time
Date Time must be a valid date, for example 19.04.2024 08:52:00.

Email
Please enter an email address.

Uri
Please enter a URL.

Pattern
Please match the requested format.

Cancel Create

9 errors have occurred

- Object Name must have some value.
- Short String must have some value.
- Int must be a valid number.
- Number must be a valid number.
- Date must be a valid date, for example 19.04.2024.

extended JSON-schema attributes

- Constant value
 - “const”: „constValue“
 - constant value for string/number/integer/boolean
- Binary data in strings (display only)
 - “contentEncoding”: “base64“
 - “contentType”: “image/png“, “image/jpg““image/gif“
 - for images on PNG, JPG or GIF format
- Recursive
 - “type”: “object“
 - “type”: “array“, “items”: [...]

JSON-schema references

- Local schema-references
 - “\$ref”: “#/defs/...”
The reference starts with “#/” and references in the current JSON-schema.
To avoid redundancies. For example when multiple addresses are required in a JSON-schema.
- Static schema-reference from DB-server
 - “\$ref”: “/defs/...”
Callback to database for selecting a static “sub-schema”, which is used in multiple JSON-schemas. The reference starts with “/”.
- Dynamic schema-reference from DB-server
 - “\$ref”: “/defs/...”
Callback to database for generating a “sub-schema”. For example to dynamically generate a select-list or a hierarchy of select-lists from a hierarchical query.
The reference starts with “/”.

Conditional JSON-schema...

- “dependentRequired”
items become mandatory when another item is not empty

```
"dependentRequired": {  
  "payment": ["card", "validity", "securitycode"]  
}
```

- “dependentSchema”
A “sub-schema” is only available, when an item is not empty.

```
"dependentSchemas": {  
  "payment": {  
    "type": "object",  
    "properties": { "creditcard": { "$ref": "#/$defs/creditcard" } }  
  }  
}
```

...Conditional JSON-schema

- “if“, “then“, “else“

Depending on a condition additional items are available.

```
"if": { "properties": { "deliverytohome": { "const": true } } },  
"then": { "properties": { "home_address": { "$ref": "#/$defs/address" } } },  
"else": { "properties": { "delivery_info": { "type": "string" } } }
```

- “allOf“, “oneOf“, “anyOf“, “not“ in an “if“ condition
- “allOf“ for schema concatenation
Useful for simplifying complex “if/then/else” conditions



History of the plug-in

Idea of the plug-in

JSON-Region-Plugin vs. JSON-source in APEX 24.2

JSON-Schema and APEX-UI

Complex JSON-schema and APEX-UI

Customizing the APEX-UI

Oracle 23ai

Experience during development

Miscellaneous

Complex JSON-schema attributes...

Schema-references: "\$ref": "/xxx/xxx"

```
1 {
2   "type": "object",
3   "required": ["select"],
4   "properties": {
5     "lastname": {"type": "string"},
6     "firstname": {"type": "string"},
7     "birthdate": {"type": "string", "format": "date"},
8     "object_type": {"$ref": "/enums/object_type", "apex": {"newRow": true}},
9     "address": {"$ref": "/defs/address"}
10  }
11 }
12
```

```
1 {
2   "type": "object",
3   "properties": {
4     "lastname": {"type": "string"},
5     "firstname": {"type": "string"},
6     "birthdate": {"type": "string", "format": "date"},
7     "object_type": {"type": "number", "enum": [4, 5, 18, 6, ...]},
8     "apex": {"enum": {"1": "Server", "2": "Switch", "3": "Printer", "4": "Full-Example", "5": "Hotel", "6": "Person", ...}},
9     "address": {"type": "object", "properties": {"zip": {"type": "string"}, "city": {"type": "string"}, "street": {"type": "string"}}}
10  }
11 }
12
```

Lastname Last	Firstname First	Birthdate 2025-03-17	
Object Type Full-Example			
Address			
Zip Zip	City City	Street Street	

```
1 {
2   "lastname": "Last", "firstname": "First", "birthdate": "2025-03-17",
3   "object_type": 4,
4   "address": {"zip": "Zip", "city": "City", "street": "Street"}
5 }
```

...Complex JSON-schema attributes

- object/array, conditional schema:

```
1 {
2   "type": "object",
3   "required": ["lastname", "email"],
4   "dependentRequired": {
5     "creditcard": ["creditid"],
6     "creditid": ["creditcard"]
7   },
8   "properties": {
9     "lastname": {"type": "string", "maxLength": 30},
10    "firstname": {"type": "string", "maxLength": 30},
11    "email": {"type": "string", "format": "email"},
12    "knowledge": {"type": "array", "items": {"type": "string", "enum": ["DB", "APEX", "Javascript", "PL/SQL"]}},
13    "creditcard": {"type": "string", "enum": ["Visa", "Mastercard", "Amex", "Diners"]},
14    "creditid": {"$ref": "#/$defs/cardid"},
15    "office_address": {"$ref": "#/$defs/address"},
16    "deliverytohome": {"type": "boolean"}
17  },
18  "if": {
19    "properties": {
20      "deliverytohome": {"const": true}
21    }
22  },
23  "then": {
24    "properties": {
25      "home_address": {"$ref": "#/$defs/address"}
26    }
27  },
28  "$defs": {
29    "name": {"type": "string", "maxLength": 30},
30    "address": {
31      "type": "object",
32      "required": ["zipcode", "city"],
33      "properties": {
34        "country": {"type": "string"},
35        "state": {"type": "string"},
36        "zipcode": {"type": "string"},
37        "city": {"type": "string"},
38        "street": {"type": "string"}
39      }
40    },
41    "cardid": {"type": "string", "pattern": "[0-9]{4}([0-9]{4}){3}" }
42  }
43 }
```

```
1 {
2   "lastname": "Simon",
3   "firstname": "Uwe",
4   "email": "usdbc@magenta.de",
5   "knowledge": ["DB", "APEX", "Javascript", "PL/SQL"],
6   "creditcard": "Visa",
7   "creditid": "1234 5678 9012 3456",
8   "office_address": {
9     "country": "D", "state": "NRW", "zipcode": "50000", "city": "City1", "street": "Street1"
10  },
11  "deliverytohome": true,
12  "home_address": {
13    "country": "D", "state": "NRW", "zipcode": "50000", "city": "City2", "street": "Street2"
14  }
15 }
```

The top screenshot shows a form with fields: Lastname, Firstname, Email, Knowledge (radio buttons for DB, APEX, Javascript, PL/SQL), Creditcard (dropdown menu), Creditid, Country, State, Zipcode, City, Street, and Deliverytohome (checkbox). The 'Creditcard' dropdown is set to 'Visa'.

The bottom screenshot shows the same form, but the 'Deliverytohome' checkbox is checked. The 'Object Type' dropdown is set to 'full_complex'. The 'home_address' field is now visible, containing the same data as the 'office_address' field.



History of the plug-in

Idea of the plug-in

JSON-Region-Plugin vs. JSON-source in APEX 24.2

JSON-Schema and APEX-UI

Complex JSON-schema and APEX-UI

Customizing the APEX-UI

Oracle 23ai

Experience during development

Miscellaneous

Customizing the APEX-UI ...

- The APEX-UI provides multiple item-type for a data-type
- All APEX-specific Configurations are stored below `"apex": {...}`
 - `"itemtype": "starrating"` Numeric as "Starrating" `"apex": {"itemtype": "starrating"}`
 - `"itemtype": "switch"/"radio"` Boolean as "Switch"/"Radio"
 - `"itemtype": "password"` Password item
 - `"itemtype": "pctgraph"` Numeric as a bar in % (0-100, display only)
 - `"itemtype": "currency"` Integer/Number as currency
 - `>=APEX-23.2`
 - `"itemtype": "richtext"` The "Richtext-editor" for long strings
 - `"itemtype": "combobox"` A multiselect combobox with „Chips“
 - `„itemtype“: „qrcode“` Display of QR-Codes
 - `>=APEX 24.1`
 - `"itemtype": "Selectone"` For a single-select
 - `"itemtype": "Selectmany"` A multiselect combobox

...Customizing the APEX-UI ...

- Other attributes below „apex“
 - „label“: “Text” Text as item label
 - "newRow": true Start a new row before the item,
 - "textBefore": "Text" Static “Text” in front of item
 - "lines": 10 Number of rows for Textarea/Richtext-Editor
 - "colSpan": 6 Width of the item (1-12)
 - “readonly”: true Item is display only
 - “direction”: “horizontal” ”Radio”/”Checkbox” group horizontally

... Customizing der APEX-UI

```
1 {
2   "type": "object",
3   "required": ["enum", "short_string"],
4   "properties": {
5     "enum": { "type": "string", "enum": [ "val1", "val2" ],
6               "apex": { "itemtype": "radio", "direction": "horizontal" } },
7     "short_string": { "type": "string" },
8     "long_string": { "type": "string", "maxLength": 400,
9                     "apex": { "itemtype": "richtext", "lines": 4, "colSpan": 12 } },
10    "bool": { "type": "boolean",
11              "apex": { "itemtype": "switch" } },
12    "int": { "type": "integer", "maximum": 5,
13             "apex": { "itemtype": "starrating", "label": "*-Rating" } },
14    "number": { "type": "number" },
15    "money": { "type": "number",
16               "apex": { "format": "currency" } },
17    "date": { "type": "string", "format": "date" },
18    "date_time": { "type": "string", "format": "date-time" },
19    "email": { "type": "string", "format": "email",
20               "apex": { "textBefore": "Subtypes" } },
21    "uri": { "type": "string", "format": "uri" },
22    "pattern": { "type": "string", "pattern": "[0-9]{4}([0-9]{4}){3}" },
23    "multi": { "type": "array",
24               "items": { "type": "string", "enum": [ "val1", "val2" ] },
25               "apex": { "itemtype": "combobox", "textBefore": "Array" } }
26   }
27 }
28 }
```

Enum* ☒ val1 ☐ val2

Short String
short

Long String

Paragraph **B** *I* <>

long
long
15"
long

Bool ☒

*Rating ★★★★★

Number 123.456

Money \$100.00

Date 2024-03-22

Date Time 2024-03-22 21:30:

Subtypes

Email support@oracle.com

Uri https://oracle.com

Pattern 1234 5678 9012 3456

Array

Multi
val1 x val2 x



History of the plug-in

Idea of the plug-in

JSON-Region-Plugin vs. JSON-source in APEX 24.2

JSON-Schema and APEX-UI

Complex JSON-schema and APEX-UI

Customizing the APEX-UI

Oracle 23ai

Experience during development

Miscellaneous

JSON-Schema generated from JSON-validation

- Starting with Oracle23ai, it is possible to define an “IS JSON VALIDATE” Check-Constraint for a VARCHAR2/CLOB/JSON-columns or collection-tables.
- So, why not use the JSON-schema in this constraints for generating the APEX-UI
- Caution:**
 - Oracle doesn't support all “complex” JSON-schema configurations.
For example “\$ref”: “...” ignored / returns error
 - In JSON-duality-views Oracle uses some extensions like “extendedType“, which is supported by the plug-in

```
1 CREATE TABLE object23c(  
2   object_id      INTEGER GENERATED BY DEFAULT ON NULL AS IDENTITY,  
3   object_name    VARCHAR2(30) NOT NULL,  
4   data           JSON,  
5   CONSTRAINT object23c_pk PRIMARY KEY (object_id)  
6 );  
7  
8  
9 ALTER TABLE object23c ADD CONSTRAINT object23c_ck1  
10 CHECK (data IS JSON VALIDATE q'[{  
11   "type"         : "object",  
12   "properties"   : {  
13     "fruit"      : {"type"       : "string",  
14                   "minLength"   : 1,  
15                   "maxLength"   : 10},  
16     "quantity"   : {"type"       : "number",  
17                   "minimum"     : 0,  
18                   "maximum"     : 100},  
19     "orderdate"  : {"type": "string",  
20                   "default": "now",  
21                   "format": "date"}  
22   },  
23   "required"     : ["fruit", "quantity"]  
24   }]' )  
25 );
```



History of the plug-in

Idea of the plug-in

JSON-Region-Plugin vs. JSON-source in APEX 24.2

JSON-Schema and APEX-UI

Complex JSON-schema and APEX-UI

Customizing the APEX-UI

Oracle 23ai

Experience during development

Miscellaneous

• Experiences during development...

Different behaviours of UI-items in JavaScript

- `apex.item(...).setValue(...)`
 - Destroys Date/Date-Time-Picker in APEX<=22.2 when called after the UI-item is rendered
 - For item-types “QRCode” and “RichTextEditor” (introduced in APEX-23.2) must wait until the rendering of the UI-Items has finished
 - QR-Code is generated in PL/SQL, uses an AJAX-request via AJAX-callback
 - RichTextEditor is initialized asynchronously, must be finished before using `apex.item().setValue`
- `<input ...>`
 - All common browsers support additional attributes like `minLength=“..“`, `„pattern=“..“`, `type=“time“`, ...
Error-messages for this `<input>`-tags is generated by the browsers, unfortunately in the language of the browser-UI but not in the language of the current page

...Experiences during development...

- The first plug-in-version, supporting the “simple” property types „string“, „integer“, „number“, „boolean“, was implemented quite fast, the next versions supporting different APEX/DB-versions and additions APEX-item-types (‘Richtext’, ‘Switch’, ‘Combobox’, ...) sub-object and arrays took much more time.
- **APEX-JavaScript-code**
 - JavaScript-code ../images/apex/libraries is available in minimized and “readable” format. The inline Documentation inside source code not always matching 100% / is partially missing
 - Depending on item/widget different implementation pattern with different behaviour
 - Oracle changes the used JavaScript-library for advanced item-types like “Richtext”, “Date-Picker”, ... which changes in behaviours/APIs

...Experiences during development

- **PL/SQL**

- There is no PL/SQL-constant for the current APEX-Release like in JavaScript

`apex.env.APEX_VERSION`

The support of features (QR-code, collection-tables, ...) in new APEX/DB-releases requires conditional compile in PL/SQL

Workarround (here for APEX für 23.2):

```
$if wwv_flow_api.c_current>=20231031 $then  
...  
$end
```



History of the plug-in

Idea of the plug-in

JSON-Region-Plugin vs. JSON-source in APEX 24.2

JSON-Schema and APEX-UI

Complex JSON-schema and APEX-UI

Customizing the APEX-UI

Oracle 23ai

Experience during development



Miscellaneous

Miscellaneous...

- When using JSON in CLOB use check-constraint
IS JSON(STRICT)
- When using a static JSON-schema for a table. Starting with Oracle23ai use check-constraint
IS JSON VALIDATE q'[...]

```
1 ALTER TABLE object ADD CONSTRAINT object_ck_1 check (data IS JSON(STRICT));
2
3
4 ALTER TABLE object23ai ADD CONSTRAINT object23ai_ck_1 CHECK (data IS JSON VALIDATE q'[
5 {
6   "type"      : "object",
7   "properties": {"fruit"   : {"type"      : "string",
8                               "minLength" : 1,
9                               "maxLength" : 10},
10                  "quantity": {"type"      : "number",
11                               "minimum"   : 0,
12                               "maximum"   : 100},
13                  "orderdate": {"type": "string",
14                                "default": "NOW",
15                                "format": "date"}
16 },
17   "required"   : ["fruit", "quantity"]
18 }
19 ]');
```

...Miscellaneous

- When a JSON-column can contain data with different JSON-schema, the data could be verified with a row-trigger
- Often the JSON-schema attribute “enum” must be in sync with a lookup-table.
2 solutions
 - A statement-Trigger on the lookup-table
 - Use a “dynamic” “\$ref”: “/enum/....” to generate the values for an “enum” or a list of dependend “enum”

```
1 CREATE OR REPLACE TRIGGER object_tr
2 BEFORE INSERT OR UPDATE ON object
3 FOR EACH ROW
4 DECLARE
5     l_schema object_type.object_schema%TYPE;
6     l_ret     PLS_INTEGER;
7 BEGIN
8     SELECT object_schema INTO l_schema
9     FROM object_type ot
10    WHERE ot.object_type_id=new.object_type_id;
11    IF NVL(JSON_VALUE(l_schema, '$.apex.validate'),'true') = 'true' THEN
12        l_ret:= DBMS_JSON_SCHEMA.is_valid(:new.data, l_schema, DBMS_JSON_SCHEMA.RAISE_ERROR);
13    END IF;
14 END;
15 /
```

```
1 SELECT json_region_generate_enum(q'l
2     SELECT relation_type_id, relation_type_name
3     FROM relation_type
4     ORDER BY relation_type_name
5     ]', NULL)
6 FROM DUAL;
```

```
1 {
2     "type": "number",
3     "enum": [1, 4, 5, 6, 2],
4     "apex": {"enum": {"1": "Laptop", "4": "Printer", "5": "Server", "6": "Server", "2": "Switch"}}
5 }
```

Known usages of the JSON-Region-Plugin

- USA:
One of the hugest county offices
Workflow of complex forms translated into JSON
- India:
PoC for a generic workflow
- Germany:
Flows4Apex: An open-source BPEL-engine for APEX

Known issues with plugin

JSON-collection-table:

```
INSERT INTO table23ai VALUES('{"fruit":"Banana","quantity":10,"orderdate":"2025-02-20"}');  
UPDATE table23ai SET data='{"fruit":"Banana","quantity":10,"orderdate":"2025-02-20"}'  
where rowid='AAArpxAAQAAAAL1AAA';
```

Oracle DB < 23.7: JSON-collection-table:

INSERT, UPDATE

ORA-00932: inconsistent datatypes: expected VARCHAR, BLOB, CLOB, FILE, BINARY, JSON – got –
ORA-54059: cannot update an immutable column ("UWE"."TABLE23AI"."RESID")

Oracle DB >= 23.7: JSON-collection-table with JSON VALIDATE constraint:

INSERT works

UPDATE

ORA-00932: inconsistent datatypes: expected VARCHAR, BLOB, CLOB, FILE, BINARY, JSON – got –

Known issues APEX 24.2

- JSON-Source:
 - Defining a source with a complex JSON-schema returns – not very helpful - error **Could not compute a Data Profile for the new JSON Source, because of the following error: ORA-06503: PL/SQL: function returned without value.**
- Duality-View:
 - While defining a source on an Oracle >=23.7 DB, I get the – the not very helpful – error **ORA-06503: PL/SQL: function returned without value**
In Oracle 23.7 the output of `dbms_json_schema.describe` has changed.
- Reason:
JSON-schema contains some unknown keywords for the JSON-schema-parser of APEX 24.2, should be fixed with an upcoming patch.

Others

- Ideas for further improvements
 - Support of JSON-schema from OpenAPI <https://swagger.io/specification/>
 - Support of JSON-schema from JSON-Form <https://jsonforms.io/>
 -
- Flows4Apex <https://flowsforapex.org/> uses the JSON-Region-Plugin
 - Here is a stand-alone APEX-application “JSON-Simple-Form-Builder” for defining and testing of forms. It also generates the JSON-schema used by the “JSON-Region-Plugin”
 - There is a “Simple Process Starter” application, which uses forms generated by the JSON-Region-Plugin

Nearly done

Thank You for your attention

Time for questions ??

Bedankt voor uw aandacht

Tijd voor vragen ??

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/>



Please fill in your
evaluations