RESTful services Part I

Consuming Secure Services

ā'pěks
(#orclapex)

Hello!

I am Richard Martens

Welcome to this presentation.

You can find me at

- @rhjmartens
- smart4solutions.nl/blog

smart4 solutions
IT's in our nature

Oracle ACE Pro

Oracle ACE Program

500+ technical experts helping peers globally

The **Oracle ACE Program** recognizes and rewards community members for their technical and community contributions to the Oracle community

3 membership tiers

Oracle ACE Director | Oracle ACE Pro | Oracle ACE Associate

For more details on Oracle ACE Program:
ace.oracle.com

Oracle ACE

Nominate
yourself or someone you know:

ace.oracle.com/nominate

Connect: aceprogram_ww@oracle.com Facebook.com/OracleACEs @oracleace

Agenda

- The low-code
  - REST data sources
  - REST enabled SQL
- The high-code
- Interpreting data
  - apex_json
  - json_table

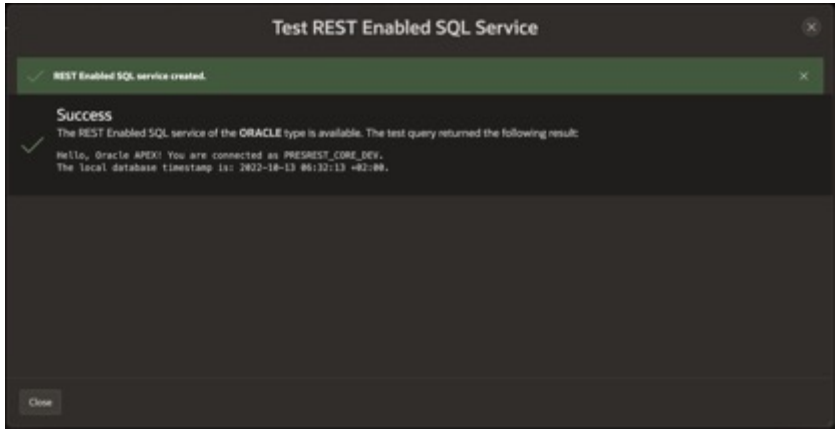
The Low-code: REST Data Sources in APEX

- Add the “REST Data Source”
  - Enter a name and the endpoint-url
  - A “remote server” will be extracted or re-used
  - Choose pagination type
  - Choose Authentication method
    - add client-id and -secret
  - Hit “Discover”
  - Hit “Create”
- Base a report on the data-source

DEMO

The Low-code: REST enabled SQL

- Add the “Rest enabled SQL” server
  - Name and Endpoint
  - Username and Password (DB Account)
- Create a Page:
  - Classic Report (with Form)
  - Interactive Report
  - Interactive Grid



- The “Rest enables SQL” Server acts as if it were a local Oracle Database

DEMO

Interpreting data: apex_json

- Can be used when unknown JSON structure.. (requires high code!!!)
- Uses JSON_PATH

- First we must parse the response as json
  - APEX_JSON.PARSE

- Determine the amount of rows
  - APEX_JSON.GET_COUNT

APEX_JSON

- GET_BOOLEAN — *gets a boolean*
- GET_CLOB — *gets a clob*
- GET_COUNT — gets the number of child elements
- GET_DATE — *gets a date*
- **GET_MEMBERS** — get attribute names of an object
- GET_NUMBER — *gets a number*
- GET_SDO_GEOMETRY — *gets a geometry*
- GET_T_NUMBER — get a number-collection [array]
- GET_T_VARCHAR2 — get a varchar2 collection [array]
- GET_VALUE — gets a json-value
- **GET_VALUE_KIND** — defines the type of the value (null=0, varchar2=5, clob=8)
- GET_VARCHAR2 — *gets a varchar2*
- See
  - Oracle Docs
  - oracle-base.com

DEMO

Interpreting data: json_table

- Resembles XML_TABLE (a lot)
- Uses JSON_PATH expressions to extract columns
- Full control on what we want to extract from the response

Interpreting data: json_table

```
select jt.*
from   json_table(q'~{ "name": "Geeta",
                       "skills": [{
                         "skill": "testing",
                         "proficiency": 30
                       }, {
                         "skill": "java",
                         "proficiency": 85
                       }, {
                         "skill": "c",
                         "proficiency": 90
                       }]
                     }~'
                  ,'$' columns(
                         "name" varchar2(30) path '$.name'
                         ,nested             path '$.skills[*]'
                            columns("skill_name" varchar2(20) path '$.skill'
                                    ,"proficiency" number path '$.proficiency')
                                 )
                  ) jt
```

DEMO

The High-code: using REST DataSource in PL/SQL

- Create a session (if we're not in APEX)
- Start the context using apex_exec.open_rest_source_query
- Determine the index-positions of the rest-sources columns
- loop "while" apex_exec.next_row(l_context)
  - Get data using:

    apex_exec.get_varchar2(l_context, t_ix('ENAME');
- Having to create the session prohibites the use outside of APEX environments

DEMO

The High-code: making a REST call

- Not applicable for a Data-Source (no stored credentials can be used)
- Do the call using APEX_WEB_SERVICE.MAKE_REST_REQUEST
- Read the response

```
declare
   -- Local variables here
   l_url      apex_appl_web_src_modules.url_endpoint%type;
   l_cred     apex_appl_web_src_modules.credential_static_id%type;
   l_response clob;
begin

   l_response := apex_web_service.make_rest_request(p_url         => 'https://
                                                   ,p_http_method => 'GET');

   for d in (select *
             from   json_table(l_response
                               ,'$.items[*]' --
                                columns(deptno varchar2 path '$.deptno'
                                       ,dname varchar2 path '$.dname'
                                       ,loc varchar2 path '$.loc')))
   loop
      dbms_output.put_line(rpad(d.deptno, 5) || rpad(d.dname, 20)|| rpad(d.loc, 20));
   end loop;

end;
```

DEMO

The High-code: making a REST call with oauth

- APEX_WEB_SERVICE.OAUTH_AUTHENTICATE

  stores token in *apex_web_service.g_oauth_token.token*
- Set token in request-headers:

  APEX_WEB_SERVICE.G_REQUEST_HEADERS (Authorization/Bearer)
- Do the actual request with MAKE_REST_REQUEST
- Read out the response with
  - JSON_TABLE or
  - APEX_JSON

DEMO

Recap

- Requesting services
  - apex_exec (apex session needed)
  - apex_webservice.make_rest_reuqest
- Focus on interpreting the JSON data
  - apex_json
  - json_table
- REST enabled SQL
  - will contain dynamic SQL
    (what if the target-database changes?)

Thanks!

Any questions ?

You can find me at

- @rhjmartens
- rmartens@smart4solutions.nl
- https://bitbucket.org/smart4solutions