

# Oracle REST Data Services Architecture, Features, & Configuration

---

**Jeff Smith**

Distinguished Product Manager

[Jeff.D.Smith@oracle.com](mailto:Jeff.D.Smith@oracle.com)

@thatjeffsmith

**Marcel Boermann-Pfeifer**

Solution Architect

[marcel.pfeifer@oracle.com](mailto:marcel.pfeifer@oracle.com)

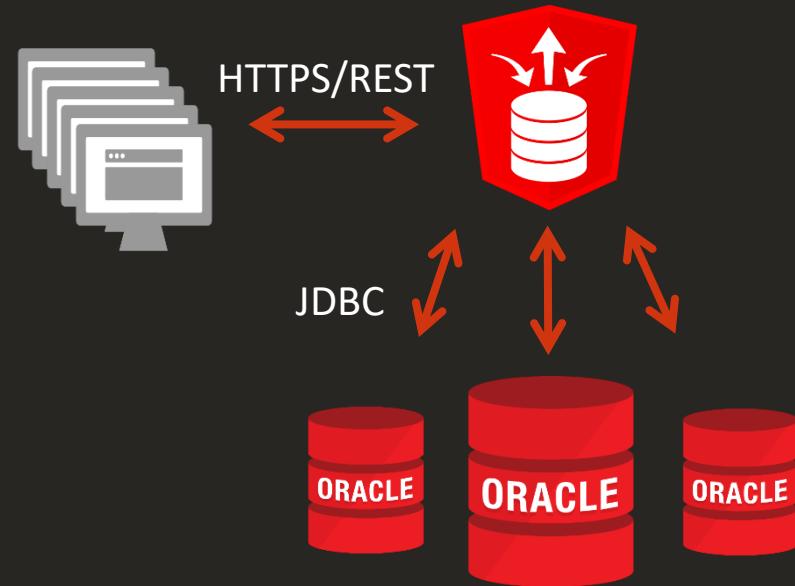
# Agenda

- Introduction to ORDS
- Installation & Configuration
- Features & Demonstration
- Get it running in SVi environment

# REST Access to your Oracle Database



- Oracle REST Data Services (ORDS)
- Auto REST enable your data tables, views, SQL, PL/SQL
- Auto paged results
- Auto JSON responses
- Complete read-write functionality
- Full dev support (GUI, CLI, API)
- Oracle Cloud and On-Premises



<https://www.oracle.com/REST>



## **Introduction**

---

Use cases, supported environments, and licensing



# Do you need a...?

- Web listener for your Oracle PL/SQL programs or APEX
- RESTful Services harness for your Oracle Database
- Web client for working with your Oracle Database
- Database management REST API
- Mongo style API for Oracle Database (SODA)

# Example: EMPS of DEPT 50

- HTTPS vs SQL
- in {json}
- Page the results
- I need links for each entry
- Handle the data types for me



A screenshot of a web browser window displaying a JSON API response. The URL is `localhost:8080/ords/hr/employees/?q={"department_id":50}`. The JSON object contains an array of employee records and a set of links for navigating the data.

```
{
  "items": [
    {
      "employee_id": 128,
      "first_name": "Matthew",
      "last_name": "Weiss",
      "email": "WEISS",
      "phone_number": "+650.123.1234",
      "hire_date": "1987-07-18T04:00:00Z",
      "job_title": "PST.MAN",
      "salary": 18431.25,
      "commission_pct": null,
      "manager_id": 100,
      "department_id": 50,
      "column": null
    },
    ... // 13 more items
  ],
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/ords/hr/employees/128"
    }
  ],
  ...
}
{
  "hasMore": true,
  "limit": 25,
  "offset": 0,
  "count": 25,
  "links": [
    {
      "rel": "self",
      "href": "http://localhost:8080/ords/hr/employees/?q=%7B%22department_id%22%3A50%7D"
    },
    {
      "rel": "edit",
      "href": "http://localhost:8080/ords/hr/employees/?q=%7B%22department_id%22%3A50%7D"
    },
    {
      "rel": "describedby",
      "href": "http://localhost:8080/ords/hr/metadata-catalog/employees"
    },
    {
      "rel": "first",
      "href": "http://localhost:8080/ords/hr/employees/?q=%7B%22department_id%22%3A50%7D"
    },
    {
      "rel": "next",
      "href": "http://localhost:8080/ords/hr/employees/?q=%7B%22department_id%22%3A50%7D&offset=25"
    }
  ]
}
```

# Example: PL/SQL web listener

Secure | https://apex.oracle.com/pls/apex/f?p=106788:59:8553756635865::NO:RP,RIR,CIR::

Feedback | Help | dpeake ▾

## Customer Tracker

Track and Manage Customers

Add Customer

Customers 19

Categories

Geographies

Referenceability

Products

Dashboard

Products 5

Partners

Competitors

Contacts 28

Activities 6

Reports

Administration

Search Customers

Product: All

Geography: All

Publicly Referenceable: All

Reference Types (any): Analyst Interview, Available for Calls, Logo, Quote, Success Story

Status: All

Sort By: Last Updated

IB

Illumina Biotech

Specialize in validation, compliance, consulting and programming.

Life Sciences, North America

CY

Cyphria

Manufacture drugs primarily for Multiple Sclerosis.

Life Sciences, North America

MM

Mogul Mashups

Produce new age marketing campaigns.

Communications, EMEA

BA

Bankgraph

Specialize in financial projections and modelling.

Financial Services, Asia Pacific

SS

Sakoro Speakers

Manufacture high-fidelity speaker systems for cinemas.

Media & Entertainment, Japan

SS

Synergy Sales Solutions

Sell financial software.

Financial Services, Asia Pacific

MG

Make Good

Provide free skills training and placements to returning veterans.

Education & Research, North America

MP

Marvel Power Systems

Manufacture automotive power plants, such as superchargers and

Automotive, South America

AA

A-Frame Aerospace

Specialize in building components for the aerospace industry.

Aerospace & Defense, North America

Reset

# Example: Browser based DB client

The screenshot shows the Oracle SQL Developer interface running in a browser at [localhost:8080/ords/hr/\\_sdw/?nav=worksheet](http://localhost:8080/ords/hr/_sdw/?nav=worksheet). The top navigation bar includes links for Home, Activity, Worksheet, and Data Modeler. The Worksheet tab is selected. The left sidebar displays the Navigator and Worksheets sections, with the HR schema selected. Under the HR schema, the EMPLOYEES table is expanded, showing columns like employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, etc. A search bar and refresh button are also present. The main workspace contains a query editor with the following code:

```
1 SELECT
2 FROM
3 EMPLOYEES
```

Below the query editor is a "Query Result" panel with tabs for Script Output, DBMS Output, Explain Plan, AutoTrace, and SQL History. The "Download" tab is selected, showing options for CSV, JSON, XML, and TEXT (.tsv). The results table displays 16 rows of employee data:

	employee	CSV	name	last_name	email	phone_number	hire_date	job_id
1		JSON	Scalari	King	Suppo0rt	515.123.4567	08/21/18 11:09:58 ...	AD_PRES
2		XML	Kochhar	Kochhar	NKOCHHAR	515.123.5368	09/21/89 04:00:00 ...	AD_VP
3		TEXT (.tsv)	De Haan	LDEHAAN	515.123.4569	01/13/93 05:00:00 ...	AD_VP	
4		103	Alexander	Hunold	AHUNOLD	590.423.4567	01/03/90 05:00:00 ...	IT_PROG
5		104	Bruce	Ernst	BERNST	590.423.4568	05/21/91 04:00:00 ...	IT_PROG
6		105	David	Austin	DAUSTIN	590.423.4569	06/25/97 04:00:00 ...	IT_PROG
7		106	Valli	Pataballa	VPATABAL	590.423.4560	02/05/98 05:00:00 ...	IT_PROG
8		107	Diana	Lorentz	DLORENTZ	590.423.5567	02/07/99 05:00:00 ...	IT_PROG
9		108	Nancy	Greenberg	NGREENBE	515.124.4569	08/17/94 04:00:00 ...	FI_MGR
10		109	Daniel	Faviet	DFAVIET	515.124.4169	08/16/94 04:00:00 ...	FI_ACCOUNT
11		110	John	Chen	JCHEN	515.124.4269	09/28/97 04:00:00 ...	FI_ACCOUNT
12		111	Ismael	Sciarrra	ISCIARRA	515.124.4369	09/30/97 04:00:00 ...	FI_ACCOUNT
13		112	Jose Manuel	Urman	JMURMAN	515.124.4469	03/07/98 05:00:00 ...	FI_ACCOUNT
14		113	Luis	Popp	LPOPP	515.124.4567	12/07/99 05:00:00 ...	FI_ACCOUNT
15		114	Den	Raphaely	DRAPHEAL	515.127.4561	12/07/94 05:00:00 ...	PU_MAN
16		115	Alexander	Khoo	AKHOO	515.127.4562	05/18/95 04:00:00 ...	PU_CFRK

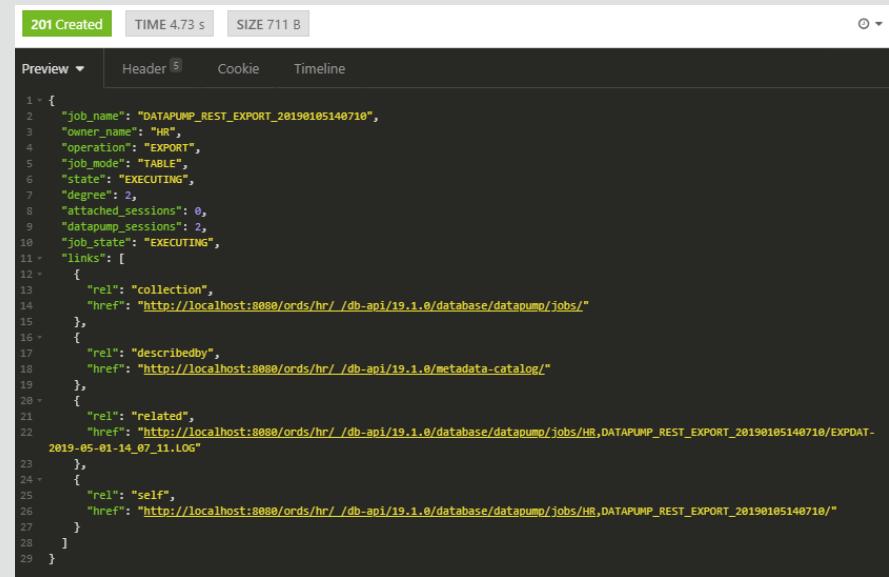
# Example: Database Management REST API

```
POST http://localhost:8080/ords/hr/_/db-api/latest/database/datapump/export
```

## POST BODY

```
{  
    "datapump_dir": "DATA_PUMP_DIR",  
    "filter": "HOCKEY_STATS, UNTAPPD",  
    "job_mode": "TABLE",  
    "threads": 2  
}
```

## Response



The screenshot shows a browser developer tools Network tab with the following details:

- Status: 201 Created
- Time: 4.73 s
- Size: 711 B

The response body is a JSON object representing a database job:

```
1  {  
2      "job_name": "DATAPUMP_REST_EXPORT_20190105140710",  
3      "owner_name": "HR",  
4      "operation": "EXPORT",  
5      "job_mode": "TABLE",  
6      "state": "EXECUTING",  
7      "degree": 2,  
8      "attached_sessions": 0,  
9      "datapump_sessions": 2,  
10     "job_state": "EXECUTING",  
11     "links": [  
12         {  
13             "rel": "collection",  
14             "href": "http://localhost:8080/ords/hr/_/db-api/19.1.0/database/datapump/jobs/"  
15         },  
16         {  
17             "rel": "describedby",  
18             "href": "http://localhost:8080/ords/hr/_/db-api/19.1.0/metadata-catalog/"  
19         },  
20         {  
21             "rel": "related",  
22             "href": "http://localhost:8080/ords/hr/_/db-api/19.1.0/database/datapump/jobs/HR_DATAPUMP_REST_EXPORT_20190105140710/EXPDAT-2019-05-01-14_07_11.LOG"  
23         },  
24         {  
25             "rel": "self",  
26             "href": "http://localhost:8080/ords/hr/_/db-api/19.1.0/database/datapump/jobs/HR_DATAPUMP_REST_EXPORT_20190105140710/"  
27         }  
28     ]  
29 }
```

# All of this is available at no additional cost

- Oracle 11gR2 and higher databases
- Support tied to your Database license (MOS)
- Classic architecture or Multitenant
- On Premise or Oracle Cloud Infrastructure
- Built by the Oracle Database Team FOR the Oracle Database

# Oracle REST Data Services



2011

## APEX Listener

- APEX based RESTful Services

2012

## APEX Listener v2

- ORDS based REST Services
- Standalone Jetty

2015

## ORDS v3.0

- APEX no longer required
- SODA
- AUTO REST

2017

## ORDS v17

- REST enabled SQL
- Type 3 JDBC Driver
- Open API (Swagger)

2018

## ORDS v18

- PL/SQL Gateway enhancements

2019

## ORDS v19

- DB API
- SQL Developer Web

2020

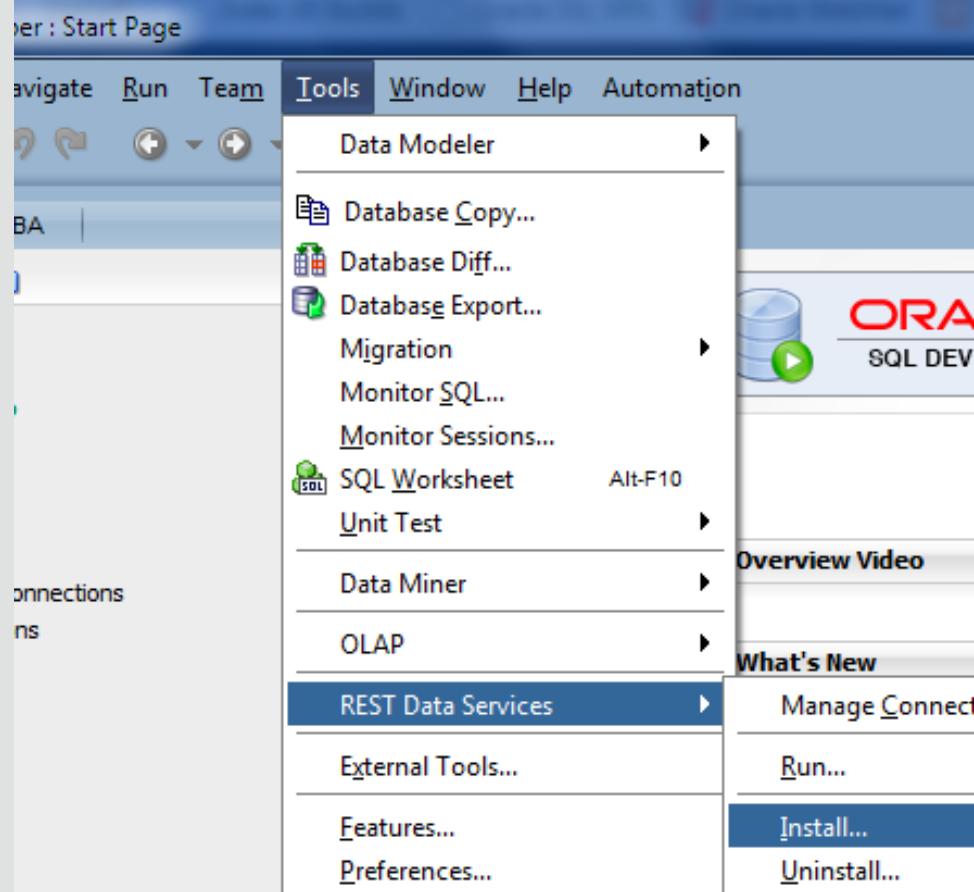
## SQL Developer Web

- REST IDE
- JSON Workshop

## Installation & Configuration

---

How to get started?



# Installation & Configuration

- Command line or graphical installers
- Silent install support
- Configure ORDS, then drop into Tomcat or WLS
  - OR run as Standalone process with embedded Jetty webserver

# java –jar ords.war install

- Follow the prompts
- Settings written out to %configdir%
- Start when finished

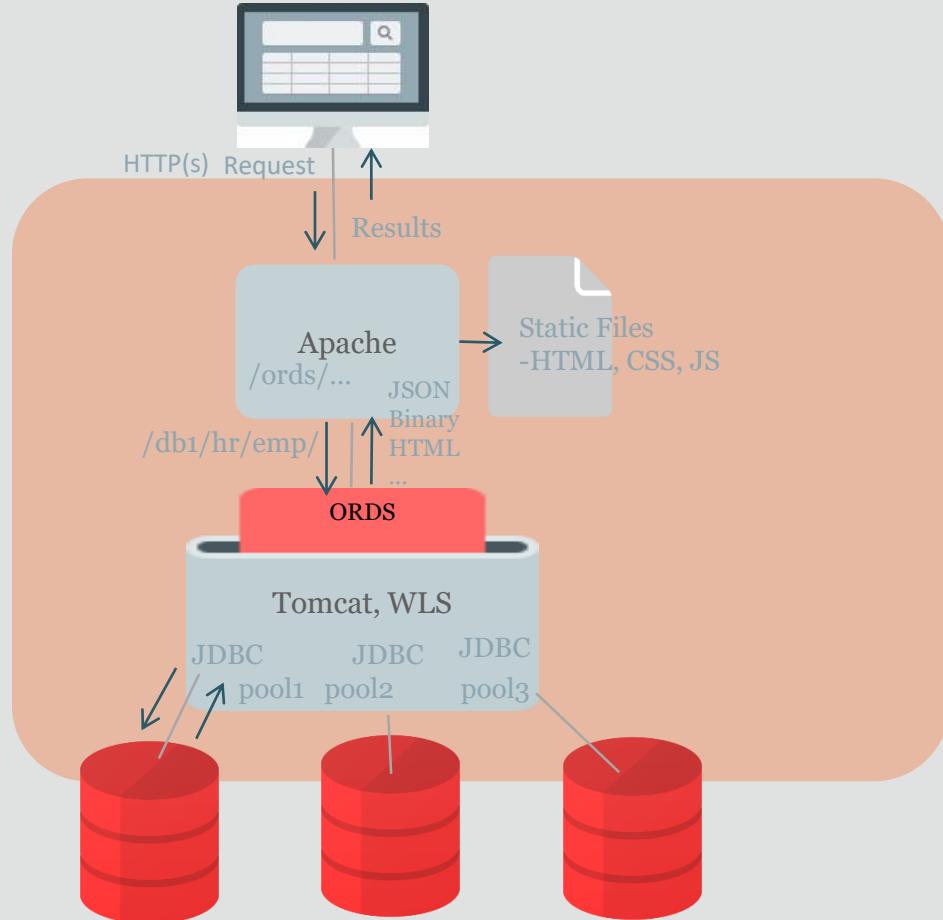
```
c:\ORDS\19.3-Final>java -jar ords.war install advanced
Using existing ORDS configuration files located at c:\ords\config\ords

Verify ORDS schema in Database Configuration apex using connection url jdbc:oracle:thin:@//localhost:1521/orcl

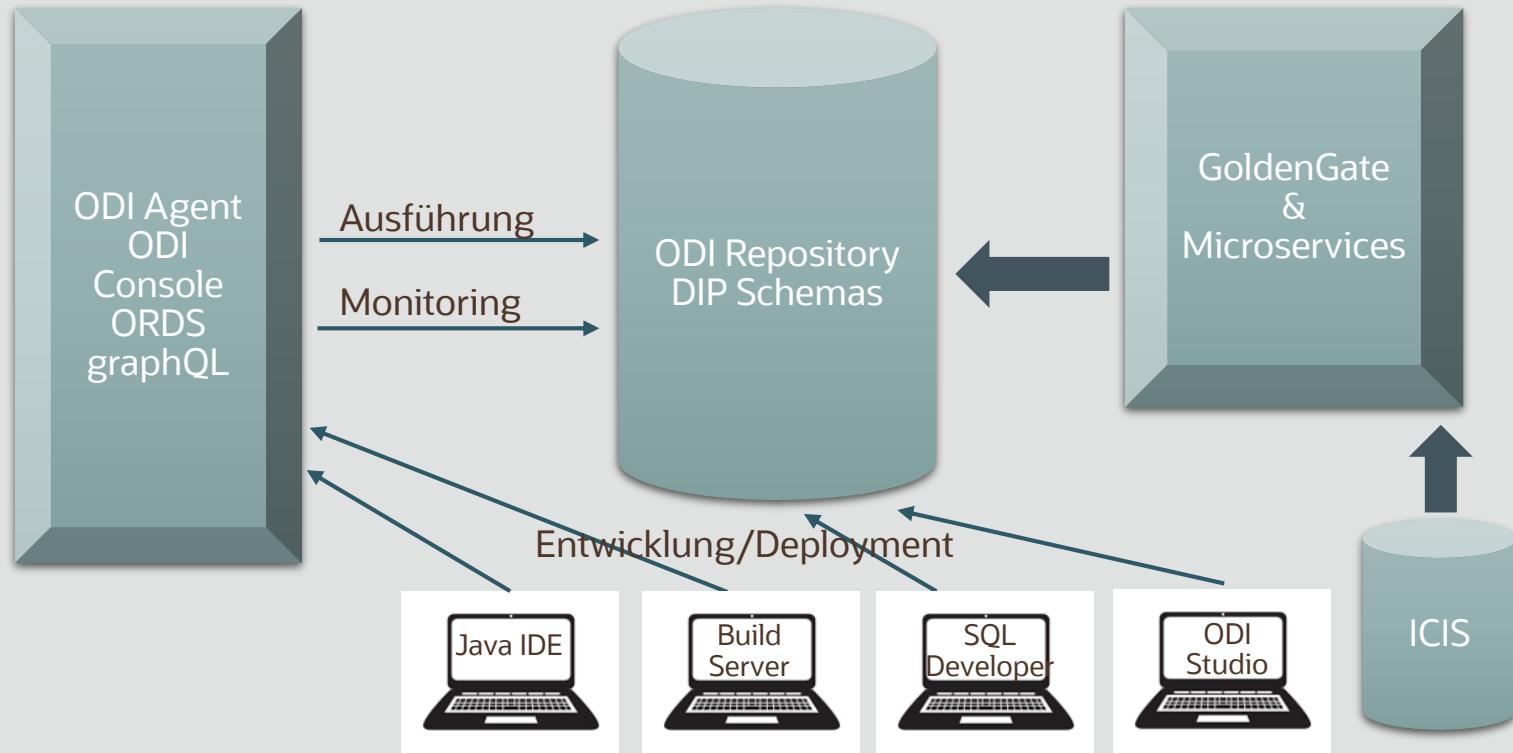
Retrieving information.
2019-12-02T19:20:17.105Z INFO  Oracle REST Data Services schema version 19.3.0.r3161548 is installed.
Enter 1 if you wish to start in standalone mode or 2 to exit [1]: ■
```

# Architecture

- Webserver layout
- Java Servlet
- Tomcat or WLS
- Standalone mode (Jetty)



# DIP schematisch bei SVi



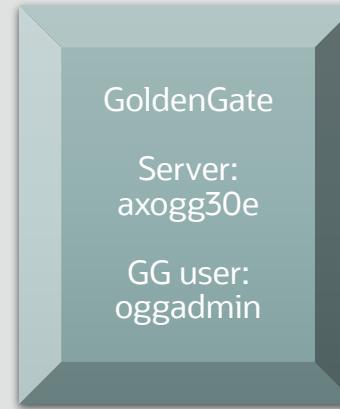
# DIP physisch bei SVi



<http://lxdip300e:7203/myapp/graphql>  
<http://lxdip300e:7103/ords/mystuff>  
<http://lxdip300e:7201/console>  
(gqldeploy / gqldeploy1)  
<http://lxdip300e:7101/console>  
Logs: /var/log/wls



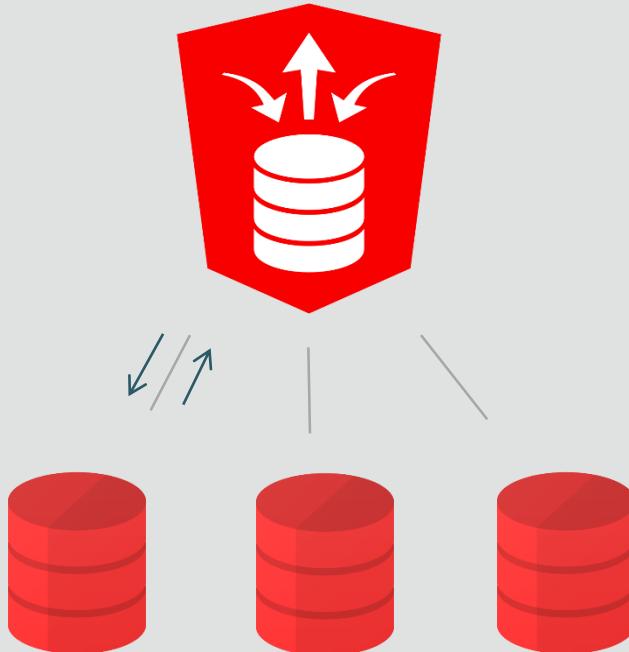
jdbc:oracle:thin:@axdip30e:1521:DPVIEW10  
SV\_ODI\_REPO / start123  
DIP\_DATASERVICE / wellcome1  
DIP\_INTEGRATION / wellcome1  
ODI\_DEMO / odi\_demo2021



<http://axogg30e:9002>

oggadmin / manager

# 1:n or 1:1 (ORDS:DB)



- Each database gets a connection pool
- Each database gets a mapping pattern, e.g. /ords/db1/...
- Ideally, 2+ ORDS front-ended with a load balancer for HA

# ORDS & Database Communication

- JDBC Conn Pools
- Default Size: 10
- 1<sup>st</sup> pool => ords/
- Proxy Connects Enabled Schemas



# Configuration

- CLI for configuration changes or..
- Edit pool XML and/or Standalone properties files



The image shows a screenshot of a development environment. On the left, there is a code editor window titled 'defaults.xml' containing XML configuration code. On the right, there is a terminal window showing the command 'java -jar ords.war setup --database orcl2' being run, followed by a series of prompts for database connection details.

```
defaults.xml ×
C: > ORDS > config > ords > defaults.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
3  <properties>
4  <comment>Saved on Tue Nov 12 13:11:52 EST 2019</comment>
5  <entry key="db.connectionType">basic</entry>
6  <entry key="db.hostname">localhost</entry>
7  <entry key="db.port">1521</entry>
8  <entry key="db.servicename">orcl</entry>
9  <entry key="feature.sdw">true</entry>
10 <entry key="restEnabledSql.active">true</entry>
11 <entry key="database.api.enabled">true</entry>
12 </properties>
13 |
```

```
Command Prompt - java -jar ords.war setup --database orcl2
c:\ORDS\19.3-Final>java -jar ords.war setup --database orcl2
Specify the database connection type to use.
Enter number for [1] Basic [2] TNS [3] Custom URL [1]:1
Enter the name of the database server [localhost]:
Enter the database listen port [1521]:
Enter 1 to specify the database service name, or 2 to specify the database SID [1]:
Enter the database service name:orcl2
```

# Common Pool Configuration Settings

- `jdbc.InitialLimit`  
*# of connections to start with, defaults to 10*
- `jdbc.MaxLimit`  
Maximum # of connections per pool

Bigger Pools <> FASTER!

Help: [Real World Perf Sizing Guide](#)

# ORACLE

DATABASE WEB SERVICES

## STARTERS

SOUPS & SALADS

### AUTO TABLE \$O

GET, PUT, POST, DELETE CRUD API  
Plus Swagger Doc and CSV Batch Load

### AUTO VIEW \$O

GET All or Some rows plus  
Metadata\_catalog describe

### AUTO PL/SQL \$O

RPC via HTTP POST  
Std Out/Refcursors to [JSON]

## ENTREES

HEARTY MEALS

### RESTFUL MODULES \$CODE

You provide the SQL and/or PL/SQL for your GET, POST, PUT, DELETE handlers. You determine the HTTP response codes. You define the inputs. You define the responses and rendering. We handle the JSON in and out.

## DESSERTS

COFFEES & CAKES

### REST ENABLED SQL

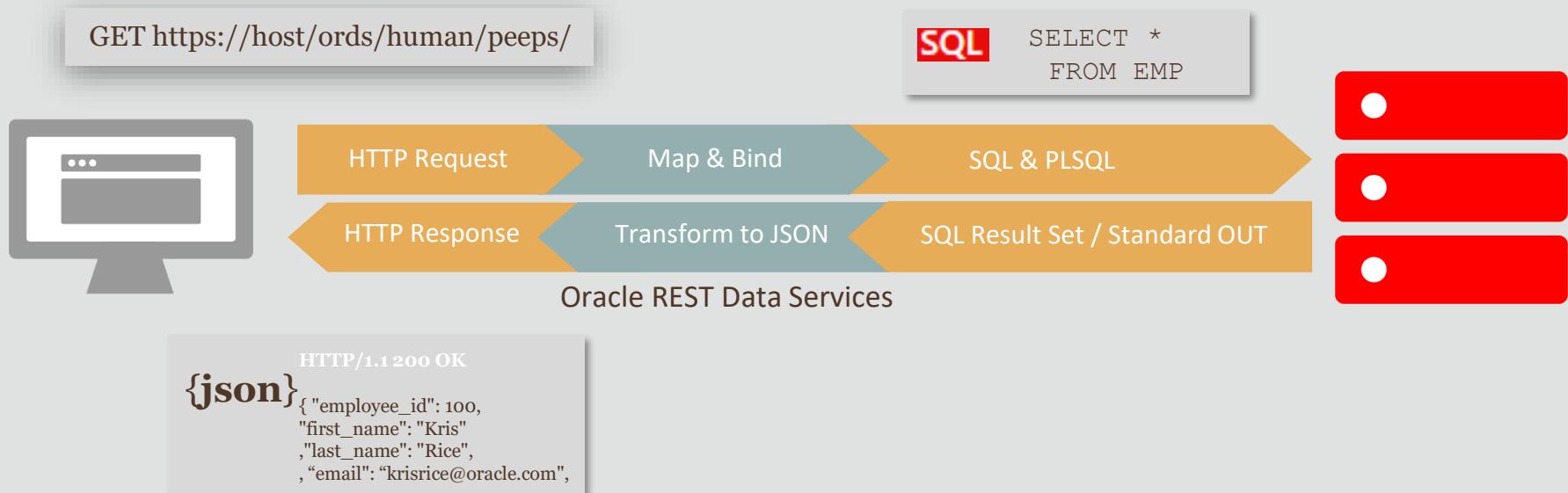
Ad Hoc PL/SQL or SQL executed via  
POST

### DB MANAGEMENT API

Manage your database via REST calls.

SQLDev  
Web

# ORDS REST Request & Response Workflow



# Unravelling an ORDS Request URI

**`https://server:8080/ords/hr/odtug/media/:id`**

- **ords** – ords.war, Java servlet
- **hr** – schema (alias!), service handler code runs as this user
- **odtug** – module
- **media/:id** – template
- Methods supported
  - GET, PUT, POST, DELETE

# Getting Started

1 Configure ORDS  
– defaults and DB  
pool, drop into  
Tomcat or start  
standalone

2 REST Enable a  
SCHEMA

3 Publish a RESTful  
Service or REST  
Enable a schema  
object

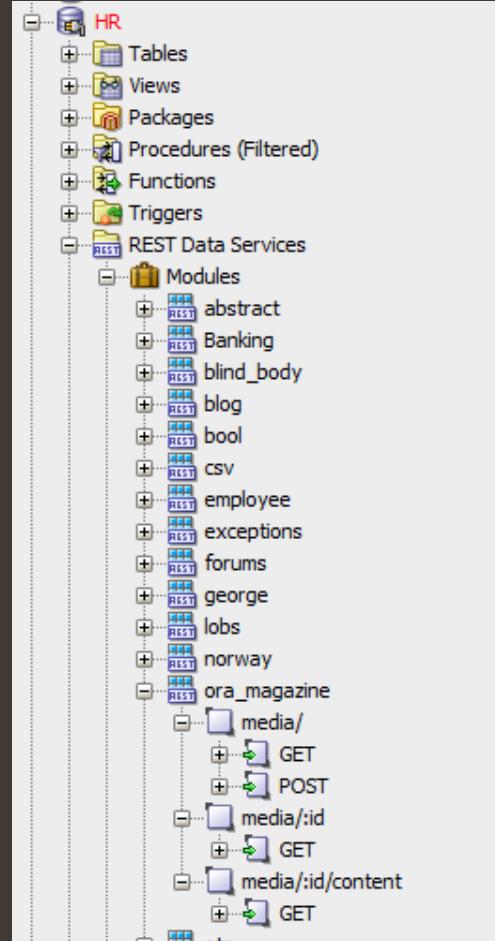
```
DECLARE<|
  ··PRAGMA AUTONOMOUS_TRANSACTION; <|
BEGIN<|
<|
  ···ORDS.ENABLE_SCHEMA(p_enabled->·TRUE, <|
  ·······p_schema->·'ORDS_DEMO', <|
  ·······p_url_mapping_type->·'BASE_PATH', <|
  ·······p_url_mapping_pattern->·'autodemo', <|
  ·······p_auto_rest_auth->·TRUE); <|
  ···<|
  ···commit; <|
<|
END;
```

# REST Enable Schema

- Endpoints defined per schema
- Services executed as that user



Privileges  
Resources



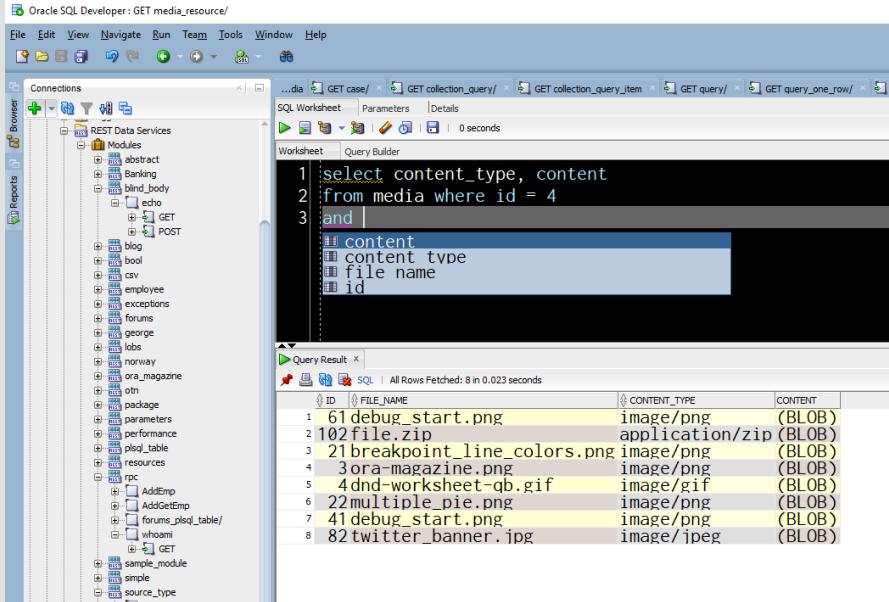
# Schema Based Services

- Automatic REST (tables, views, stored procs)
- RESTful Services

HTTP REQ => Database Work

HTTP_CALL	THE_CODE
GET /test/describe	select c.column_name, case
GET /test/version	select banner from v\$version
GET /bool/true/	declare x boolean;begin x :=
GET /test3/handlers	select 1 from dual
GET /employee/employee-no-nulls/	SELECT EMPLOYEE_ID ,FIRST_N
GET /george/ib_test2/	select * from IB_TEST2
GET /test2/sysdate	select sysdate from dual
POST /parameters/headers/:id	begin owa_util.status_line
GET /test2/media/:id	SELECT content_type, imageFl
GET /forums/onecall/	select * employees
GET /test2/case/	select first_name "A", last
POST /blind/echo	begin owa_util.status_line
GET /test2/case/	

# IDE



## Oracle SQL Developer

- Full ORDS Integration
- Develop RESTful Services
- REST Enable Objects
- Manage ORDS Roles and Privileges

# Browser Based IDE

The screenshot shows the Oracle Database Actions REST interface. At the top, there's a navigation bar with links for Database Actions, REST, Overview, Modules, Security, and a dropdown menu. Below the navigation is a summary section titled 'Objects' with four categories: MODULES (55), ROLES (30), PRIVILEGES (32), and CLIENTS (2). Each category has a small circular icon with a colored dot (blue for modules, red for roles, orange for privileges, purple for clients) and a count. Below this is a 'Security' section with two cards: 'Metadata Catalog' (locked padlock icon, requires authorization, URL: http://localhost:8080/ords/hr/metadata-catalog/) and 'Modules' (green circle icon, 54/55 published REST modules available for consumption). There's also a 'Module Security' card showing 10/55 REST modules secured by privilege. The bottom section is 'Recent Objects', a table with columns for Object Type, Name/URI, and Updated On. It lists four entries: 1 Handler (GET /employees/{id}), 2 Template (employees/{id}), 3 Module (livelabs.kay.awesome (/kay/)), and 4 Handler (GET /employees/makes/at/least).

Object Type	Name/URI	Updated On
1 Handler	GET (employees/{id})	19 days ago
2 Template	employees/{id}	19 days ago
3 Module	livelabs.kay.awesome (/kay/)	19 days ago
4 Handler	GET (employees/makes/at/least)	25 days ago

## SQLDev Web

- Develop REST Services
- Manage ORDS Roles, Privileges, OAuth2

*Coming Soon: Manage REST Enable Objects (20.4)*

# CLI & PL/SQL APIs

```
c:\Users\jdsmit\Desktop\ords173>java -jar ords.war help
java -jar ords.war <COMMAND> [Options] [Arguments]

The following commands are available:

  configdir      Set the value of the web.xml
                  config.dir property

  help           Describe the usage of this
                  program or its commands

  install        Installs Oracle REST Data
                  Services

  map-url        Map a URL pattern to the
                  named database connection

  nosqladd       Add NoSQL store configuration

  nosqldel       Delete NoSQL store
                  configuration

  oam-config     Configure web.xml to support
                  Oracle Access Manager
                  Identity Asserter on Oracle
                  WebLogic

  plugin         Package one or more plugin
                  jar files into ords.war

  schema         Install or upgrades ORDS
                  schema
```

```
DECLARE «|
  ··PRAGMA AUTONOMOUS_TRANSACTION; «|
BEGIN «|
«|
  ···ORDS.ENABLE_SCHEMA(p_enabled->·TRUE, «|
  ·············p_schema->·'ORDS_DEMO', «|
  ·············p_url_mapping_type->·'BASE_PATH', «|
  ·············p_url_mapping_pattern->·'autodemo', «|
  ·············p_auto_rest_auth->·TRUE); «|
  ·····«|
  ···commit; «|
«|
END;
```

# Choose your own adventure!

## AUTO REST Advantages

- CRUD APIs, no code
- Single call to create
- Maintained by ORCL
- Lots of features
- Optimized

## RESTful Service Advantages

- You're in charge
  - Inputs, outputs, error handling, response codes, formatting
- Full access to SQL/PLSQL
- Easily exported, source controlled
- Transparent

# AUTO REST Features

# Automatic – ORDS owns the code

- Auto REST Table
  - Full CRUD API, Data Loading, Metadata (DESC)
- Auto REST View
  - Read interface only (GET)
- Auto PL/SQL (RPC)
  - POST to execute stored PL/SQL
  - We accept {json} in, map to input params, grab output and {json} out

# AutoREST Table

/ords/hr/employees/  
/ords/hr/employees/:id

- GET
- PUT
- POST
- DELETE

The screenshot shows the Oracle SQL Developer interface with the HR schema loaded. The top section displays the ER diagram for the HR schema, showing the relationship between the EMPLOYEES and DEPARTMENTS tables. The EMPLOYEES table has columns: EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, and DEPARTMENT\_ID. The DEPARTMENTS table has columns: DEPARTMENT\_ID, DEPARTMENT\_NAME, MANAGER\_ID, and EXTRA\_COLUMN. Various constraints like EMP\_EMAIL\_UK, EMP\_EMP\_ID\_PK, and EMP\_DEPT\_FK are shown. The bottom section shows a table view of the EMPLOYEES data:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	50	King	SKING	515.123.4567	17-JUN-1987	00.00.00 AD PRES	24000
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-1989	00.00.00 AD VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-1993	00.00.00 AD VP	17000
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-1990	00.00.00 IT PROG	9000
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-1991	00.00.00 IT PROG	6000
105	David	Austin	DAUSTIN	590.423.4569	25-JUN-1997	00.00.00 IT PROG	4800
106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-1998	00.00.00 IT PROG	4800
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-1999	00.00.00 IT PROG	4200
108	Nancy	Greenberg	NGREENBE	515.123.4560	17-AUG-2004	00.00.00 IT MGR	12000

## RESTful Services Wizard - Step 1 of 2

### Specify Details

- [Specify Details](#)
- [RESTful Summary](#)

Enable object

Object alias

Authorization required

```
DECLARE
    PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN

    ORDS.ENABLE_OBJECT(p_enabled => TRUE,
                       p_schema => 'HR',
                       p_object => 'CONF_ATTENDEE',
                       p_object_type => 'TABLE',
                       p_object_alias => 'conf_attendee',
                       p_auto_rest_auth => FALSE);

    commit;

END;
```

Help

< Back

Next >

Finish

Cancel

# GET on a REST Enabled TABLE



A screenshot of a web browser window displaying a JSON response from a REST API. The URL in the address bar is `localhost:8080/ords/hr/beers/?q={"rating_score": "$eq":5}`. The JSON data shows two beer items. The first item has a rating score of 5.0 and was created at 2011-08-30T17:52:25Z. The second item has a rating score of 5.0 and was created at 2011-09-26T19:08:48Z.

```
{
  "items": [
    {
      "beer_name": "Devil's Tramping Ground Tripel",
      "brewery_name": "Aviator Brewing Company",
      "beer_type": "Belgian Tripel",
      "beer_abv": 9.2,
      "beer_ibu": 31,
      "comments": null,
      "venue_name": "Tribeca Tavern",
      "venue_city": "Cary",
      "venue_state": "NC",
      "venue_country": "United States",
      "venue_lat": 35.7921,
      "venue_lng": -78.8479,
      "rating_score": 5.0,
      "created_at": "2011-08-30T17:52:25Z",
      "checkin_url": "https://untappd.com/c/1868560",
      "beer_url": "https://untappd.com/beer/9366",
      "brewery_url": "https://untappd.com/Brewery/1394",
      "brewery_country": "United States",
      "brewery_city": "Fuquay Varina",
      "brewery_state": "NC",
      "flavor_profiles": null,
      "purchase_venue": null,
      "serving_type": null,
      "checkin_id": 1868560,
      "bid": 9366,
      "brewery_id": 1394,
      "photo_url": null,
      "id": 3001,
      "links": [
        {
          "rel": "self",
          "href": "http://localhost:8080/ords/hr/beers/https%3A%2F%2Funtappd.com%2Fc%2F1868560"
        }
      ]
    },
    {
      "beer_name": "Seeing Double IPA",
      "brewery_name": "Foothills Brewing",
      "beer_type": "IPA - Imperial / Double",
      "beer_abv": 8.3,
      "beer_ibu": 91,
      "comments": null,
      "venue_name": "Spirits Pub & Grub",
      "venue_city": "Cary",
      "venue_state": "NC",
      "venue_country": "United States",
      "venue_lat": 35.7919,
      "venue_lng": -78.7654,
      "rating_score": 5.0,
      "created_at": "2011-09-26T19:08:48Z"
    }
  ]
}
```

- No code
- SELECT \* ...
- Query Predicates with ?q={...}
- Paged results
- Links to items

## ORDS generated API for CONF\_ATTENDEE 1.0.0

[ Base URL: localhost:8080/ords/hr/conf\_attendee ]

Schemes

HTTP

default

GET

/

POST

/

GET

/{id}

PUT

/{id}

DELETE

/{id}

Models

CLOB string

DATE string

pattern: ^\d{4}-\d{2}\d-[0-23]\dT\d{2}\d\d:\d{2}\d-\d{2}\d\d(\.\d+)?(\Z|([-+]\d{2}\d\d)\d(\.\d+)?(\Z|([-+]\d{2}\d\d)\d))\$

NUMBER number

PAYOUT1 ▾ {

```
CONF_ID      NUMBER number
PERSON_ID    NUMBER number
SPEAKER      NUMBER number
REGISTRATION_DATE DATE string
ATTENDED     NUMBER number
PAID         NUMBER number
NOTE          CLOB string
```

}

- Schema level Metadata
- Table Metadata
- Get ( Select )
  - Query ( Filtering/Order/ASOF )
- Insert
- Update
- Delete
- Load CSV

# Query the TABLE

- All rows /
- One row /:id – PK Value
  - No PK, default to ROWID
  - Multi-column PK /x,y,z
- Some rows /?q={json}

example:

[/ords/odidemo/cust360/?q={"\\$eq": {"custid":102}}](/ords/odidemo/cust360/?q={)

The screenshot shows a browser window with the URL `localhost:8080/ords/hr/peeps/?q={"$eq":{"first_name":"David"}}`. The page displays a JSON array of employee records. A red arrow points from the URL query parameter to the corresponding WHERE clause in the SQL statement on the right.

```
SELECT *  
FROM EMPLOYEES  
WHERE FIRST_NAME = 'David'
```

```
[{"employee_id": 105, "first_name": "David", "last_name": "Austin", "email": "DAUSTIN", "phone_number": "590.423.4569", "hire_date": "1997-06-25T04:00:00Z", "job_id": "IT_PROG", "salary": 9868.75, "commission_pct": null, "manager_id": 103, "department_id": 60, "column1": null, "links": [{"rel": "self", "href": "http://localhost:8080/ords/hr/peeps/105"}]}, {"employee_id": 151, "first_name": "David", "last_name": "Bernstein", "email": "DBERNSTE", "phone_number": "011.44.1344.345268", "hire_date": "1997-03-24T05:00:00Z", "job_id": "SA_REP", "salary": 19507.43, "commission_pct": 0.25, "manager_id": 145, "department_id": 80, "column1": null, "links": [{"rel": "self", "href": "http://localhost:8080/ords/hr/peeps/151"}]}, {"employee_id": 165, "first_name": "David", "last_name": "Lee", "email": "DLEE", "phone_number": "011.44.1346.529268", "hire_date": "2000-02-23T05:00:00Z", "job_id": "SA_REP", "salary": 13970.31, "commission_pct": 0.1, "manager_id": 147, "department_id": 80, "column1": null, "links": [{"rel": "self", "href": "http://localhost:8080/ords/hr/peeps/165"}]}]
```

# Update a ROW

The screenshot shows the Postman interface with the following details:

- Method:** PUT
- URL:** <http://localhost:8888/ords/peeps/autodemo/1234567890>
- Body Type:** JSON (application/json)
- Request Body Content:**

```
1 {  
2     "times": "1999-07-10T15:00:00.511Z",  
3     "dates": "2017-04-28T20:50:08Z",  
4     "numbers": 1234567890,  
5     "chars": "UPDATE: this is just a demo"  
6 }
```
- Headers:**
  - Access-Control-Allow-Credentials → true
  - Access-Control-Allow-Origin → chrome-extension://fhbjgbiflinjbdggehcddcbnccccdop
  - Access-Control-Expose-Headers → Content-Location, Content-Type, ETag, Access-Control-Allow-Origin, Access-Control-Allow-Credentials, Vary
  - Content-Location → <http://localhost:8888/ords/peeps/autodemo/1234567890>
  - Content-Type → application/json
  - ETag → "HyclsJefw858D0lhVTI3hzKBS2603X7aSoVus+xr6XS2uEITV2boUzK4Ve2WB57qXBlk9vmf6GHG2l2PnYHw=="
  - Transfer-Encoding → chunked
  - Vary → Origin
- Status:** 200 OK
- Time:** 450 ms

Table Columns in the PUT body  
{"column\_name":  
column\_value}

METHOD : PUT /:PK  
REQUEST BODY : JSON  
RESPONSE: 200 OK

- Location (Header)
- JSON (Body)

# Existing PL/SQL APIs?

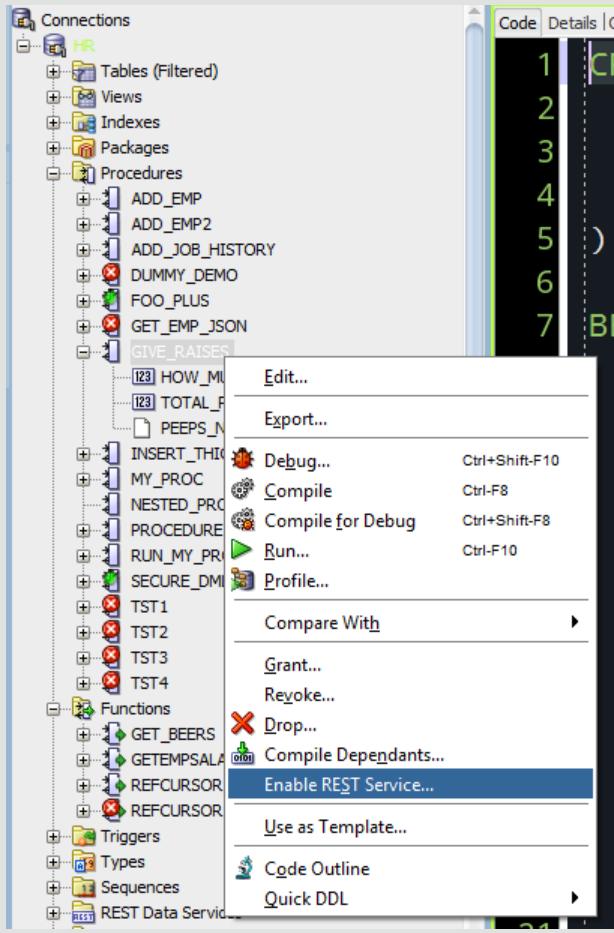
- We auto-magically handle PL/SQL too
- RPC -> POST via HTTPS /ords/hr/procedureA
- Responses & Results (OUTs/RETURNS/REFCURSORs), in {JSON}

# RPC over HTTPS (POST)

ORDS takes parameters as JSON,  
executes PL/SQL, grabs output,  
sends back down as JSON

## OUT INTEGER & SYS\_REFCURSOR

```
{  
    "total_payroll": 631230,  
    "peeps_numbers": [  
        {  
            "id": 81,  
            "name": "Dummy4",  
            "salary": 0,  
            "hire_date": "2017-06-20T13:29:00Z"  
        },  
        {  
            "id": 65,  
            "name": "Bart",  
            "salary": 0,  
            "hire_date": "2017-06-20T13:29:00Z"  
        },  
        {  
            "id": 79,  
            ...  
        }  
    ]  
}
```



POST http://localhost:8888/ords/peeps/thickdb/

Authorization Headers Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (6) Tests

Pretty Raw Preview JSON

```

1 [ {
2   "ret": [
3     {
4       "log_id": 92996,
5       "log_date": "2016-11-07T11:58:02.517Z",
6       "owner": "ORDS_METADATA",
7       "job_name": "CLEAN_OLD_ORDS_SESSIONS",
8       "job_subname": null,
9       "status": "SUCCEEDED",
10      "error#": 0,
11      "req_start_date": "2016-11-05T23:07:45.857",
12      "actual_start_date": "2016-11-06T02:06:21.506Z",
13      "run_duration": {
14        "DAYS": 1,
15        "HOURS": 9,
16        "MINUTES": 51,
17        "SECONDS": 36
18      },
19      "instance_id": 1,
20      "session_id": "89_57445",
21      "slave_pid": "4907",
22      "cpu_used": {
23        "NANOS": 18000000
24      },
25      "credential_owner": null,
26      "credential_name": null,
27      "destination_owner": null,
28      "destination": null,
29      "additional_info": null,
30      "errors": null,
31      "output": null,
32      "binary_errors": null,
33      "binary_output": null
34    }
35  ]
36 }
37 
```

# RETURN REFCURSOR

Auto PLSQL - Insert Record

POST http://localhost:8888/ords/peeps/insert\_thickdb/

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {"string": "good morning Montreal"}
```

Body Cookies Headers (6) Tests

Pretty Raw Preview

Status: 200 OK Time: 3398 ms

Save Response

Connections

Start Page | GET files | GET\_EMP\_JSON | GET\_BEERS | INSERT\_THICKDB | THICKDB

Code | Dependencies | Grants | Errors | Details | Profiles | References

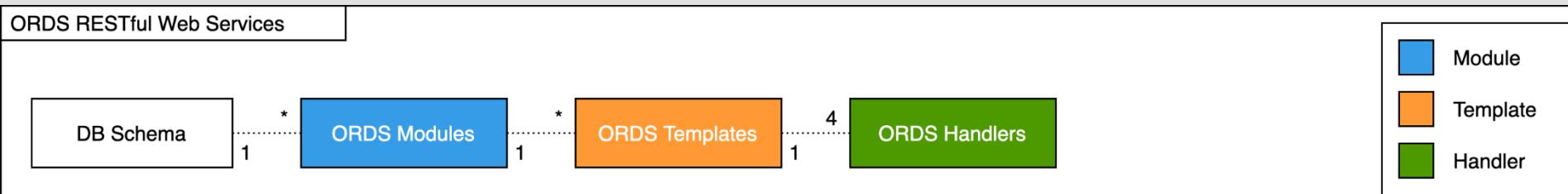
```

create or replace PROCEDURE INSERT_THICKDB
(
  STRING IN VARCHAR2
) AS
BEGIN
  INSERT INTO THICK_INSERT VALUES (STRING);
  commit;
END INSERT_THICKDB;
  
```

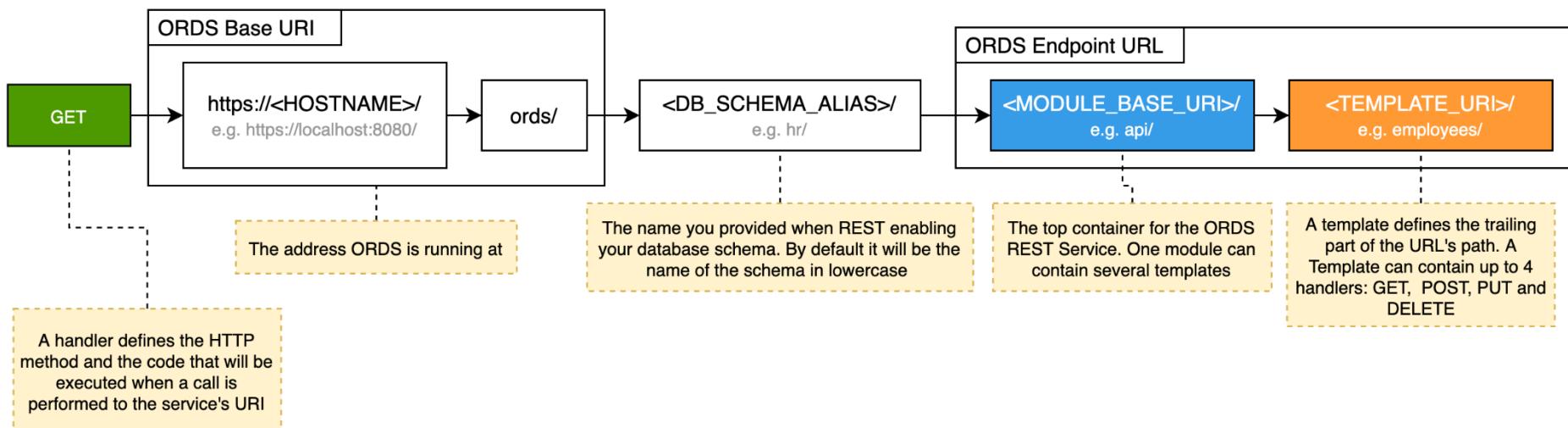
# PL/SQL TABLE API



# RESTful Services (Modules, Templates, & Handlers)

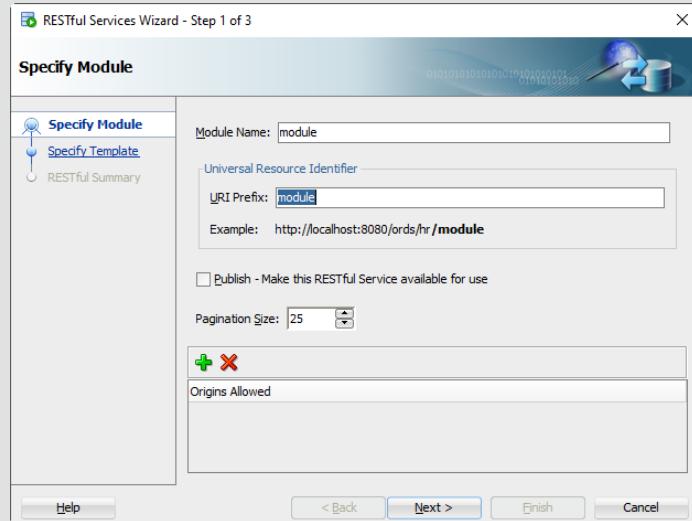


A GET request example for: `https://<HOSTNAME>/ords/<DB_SCHEMA_ALIAS>/<MODULE_BASE_URI>/<TEMPLATE_URI>/`

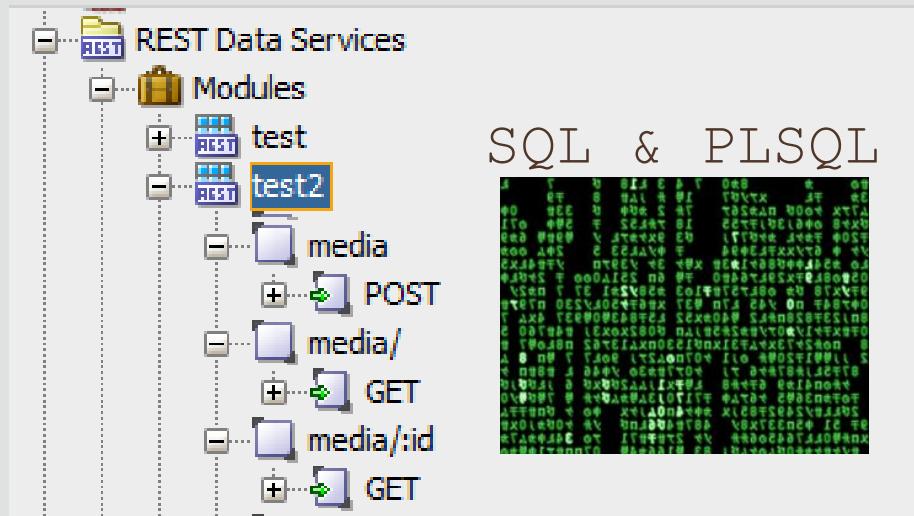


# Modules (“packages”)

- A collection of RESTful Services
- Secured as a group with a privilege
- Published or hidden as a group
- Default response pagination size



# URI Templates



/ords/hr/test2/employees  
/ords/hr/test2/employees/  
/ords/hr/test2/employees/:id

# Handlers – inputs, code, & response

The screenshot shows the Oracle SQL Developer interface with three main panes illustrating the configuration of a RESTful handler.

**Left Pane (Connections):** Displays the connection to the 'exa express JEFF' database, specifically the 'hr' schema. The 'Tables (Filtered)' section is expanded, showing tables like add\_emp, addGetEmp, other\_user, thirty, tweets, and others.

**Middle Pane (Worksheet):** Contains the PL/SQL code for the 'POST AddEmp' handler:begin
 add\_emp(
 p\_emp\_name =>:p\_emp\_name,
 p\_emp\_salary =>:p\_emp\_salary,
 p\_out\_id =>:new\_epid,
 p\_out\_total =>:total
 );
 COMMIT;
 :status := 302;
 :location := './simple\_emp/61';
end;

**Right Pane (Parameters):** Shows the configuration for the 'POST' method. The 'Method Handler' tab is selected, displaying the method type as 'POST'. The 'Source Type' is set to 'PL/SQL'. The 'Parameters' tab lists two output parameters: 'LOCATION' (Bind Parameter: 'location', Access Method: 'OUT', Source Type: 'HTTP HEADER', Data Type: 'STRING') and 'STATUS' (Bind Parameter: 'status', Access Method: 'OUT', Source Type: 'HTTP HEADER', Data Type: 'INTEGER'). Below this, the 'MIME Types' section specifies 'application/json'.

**Bottom Pane (Examples):** Provides examples of the handler's usage:

- URI Module: /rpc/
- URI Pattern: AddEmp
- Example URL: http://myhost:8080/ords/myschema/rpc/AddEmp

# Source Types - Shapes Responses

- Collection Query
  - Multiple records/paging
- Collection Query Item
  - Single record only/no paging

```
"employee_id": 124,  
"first_name": "Kevin",  
"last_name": "Mourgos",  
"email": "KMOURGOS",  
"phone_number": "650.123.5234",  
"hire_date": "1999-11-16T05:00:00Z",  
"job_id": "ST_MAN",  
"salary": 11919.54,  
"commission_pct": null,  
"manager_id": 100,  
"department_id": 50,  
"column1": null  
}  
],  
▼ "first": {  
| "$ref": "http://localhost:8080/ords/hr/source_type/query/"  
|},  
▼ "next": {  
| "$ref": "http://localhost:8080/ords/hr/source_type/query/?page=1"  
|}  
}
```

**No Metadata**

# Source Types – Determines Responses

- Feed
  - Multiple records/paging
  - First column auto-generates \$link
- PL/SQL
  - Executes PL/SQL Block
- Media Resource
  - First Column sets Mime Type
  - Second Column sets raw content (no json-ification)

The screenshot shows two browser tabs side-by-side. Both tabs have the URL `localhost:8080/ords/hr/source_type/feed/100`.  
The left tab displays the JSON response for the 'Feed' source type. It contains an array of objects under the key 'items'. Each object has a 'url' field containing a self-link (`http://localhost:8080/ords/hr/source_type/feed/100`) and various employee details like employee\_id, first\_name, last\_name, etc.  
The right tab displays the JSON response for the 'Collection' source type. It contains an array of objects under the key 'items'. Each object has a 'url' field containing a self-link (`http://localhost:8080/ords/hr/source_type/feed/100`) and various employee details like employee\_id, first\_name, last\_name, etc. Additionally, it includes a 'link' field with a 'rel' value of 'collection' and a 'href' value of `http://localhost:8080/ords/hr/source_type/feed/*`.

```
[{"url": "http://localhost:8080/ords/hr/source_type/feed/100", "employee_id": 100, "first_name": "Support", "last_name": "King", "email": "support@hr.com", "phone_number": "515.123.4567", "hire_date": "2010-09-21T11:09:08Z", "job_id": "AD_PRES", "salary": 120000, "commission_pct": null, "manager_id": null, "department_id": 90, "column": null}, {"url": "http://localhost:8080/ords/hr/source_type/feed/101", "employee_id": 101, "first_name": "Kochhar", "last_name": "King", "email": "KOCO@HR.COM", "phone_number": "515.123.4568", "hire_date": "2010-09-21T04:00:00Z", "job_id": "AD_VP", "salary": 140000, "commission_pct": null, "manager_id": 100, "department_id": 90, "column": null}, {"url": "http://localhost:8080/ords/hr/source_type/feed/102", "employee_id": 102, "first_name": "Lex", "last_name": "Meyer", "email": "LMEYER@HR.COM", "phone_number": "515.123.4569", "hire_date": "2010-09-21T11:09:08Z", "job_id": "AD_ASST", "salary": 110000, "commission_pct": null, "manager_id": null, "department_id": 90, "column": null}], [{"rel": "collection", "href": "http://localhost:8080/ords/hr/source_type/feed/*"}]
```

Feed Source Type

# Manual JSONification: JSON DB API

CUSTOMERS

CUST_ID	CUST_LAST_NAME	CUST_CITY
25451	Everett	Heilbronn
25452	Odenwalld	Heilbronn
...	...	...



```
{  
  "custid":25451,  
  "name":“Everett”,  
  "City":“Heilbronn“  
}
```

# Generating JSON – easy with SQL

- JSON\_OBJECT creates single JSON objects

```
select json_object('custid' is CUSTID,
                   'name' is LAST_NAME,
                   'City' is CITY) output
from src_customer, src_city
where src_customer.city_id = src_city.city_id;
```

AUSGABE

```
{"custid":103075,"name":"Haus","City":"Tucumcari"}
 {"custid":104203,"name":"Seto","City":"Oran"}
 {"custid":102374,"name":"Hornick","City":"Mackville"}
 {"custid":102440,"name":"Spivak","City":"Atwood"}
 {"custid":102943,"name":"Kelly","City":"Rock Creek"}
 {"custid":101789,"name":"Krishnan","City":"Loundesville"}
 {"custid":100115,"name":"Carbery","City":"Frederick"}
```

- JSON\_OBJECTAGG aggregates rows into a single column

```
select city,
       json_objectagg('name' is last_name) cust_per_city
  from src_customer, src_city
 where city like 'Co%'
   and src_customer.city_id like src_city.city_id
 group by city;
```

# Result

```
CUST_CITY
```

```
-----  
CUST_PER_CITY
```

```
-----  
Coburg
```

```
{"name": "Rice", "name": "Tillman", "name": "Barnett", "name": "Zimmer", "name": "Klessner",  
", "name": "Ingersoll", "name": "Bane", "name": "Vankirk", "name": "Lefevre", "name": "Tri  
mmer", "name": "Smeed", "name": "Lipp", "name": "Klemm", "name": "Obrien", "name": "Kelley  
", "name": "Titus", "name": "Felt", "name": "Joseph", "name": "Colven", "name": "Snodgrass  
", "name": "Rosenblum", "name": "Burgess", "name": "Kraft", "name": "Krebs", "name": "Zhao  
", "name": "Carr", "name": "Kish", "name": "Dwyer", "name": "Eastwood", "name": "Haole", "n  
ame": "Lamar", "name": "Lengel", "name": "Knox", "name": "Mannings", "name": "Curr", "name  
": "Jeffreys", "name": "Lickey", "name": "Ingold", "name": "Goode", "name": "Kelleher", "n  
ame": "Ladd", "name": "Greenley", "name": "Rush", "name": "Gilmour", "name": "Baley"}
```

```
Cochem
```

```
{"name": "Eden", "name": "Pack", "name": "Hale", "name": "Newsome", "name": "Calahan", "na  
me": "Lucy", "name": "Newkirk", "name": "Jewell", "name": "Parks", "name": "Rust", "name":  
"Sampler", "name": "Stone", "name": "Poindexter", "name": "Geralt", "name": "Wiser", "nam  
e": "Valentino", "name": "Bakker"}
```

# Generating JSON – easy with SQL

- `JSON_ARRAYAGG` is an aggregate function returning a JSON Array

```
select json_object('country_name' is r.country,
                  'ctinfo' is
                    (select json_arrayagg(
                        json_object('name' is last_name,
                                   'phone' is phone,
                                   'city' is city,
                                   'street' is address absent on null)
                        absent on null)
                     from src_customer co , src_city c, src_region r
                     where co.city_id = c.city_id
                     and c.region_id = r.region_id
                     and age < 40)
                     absent on null) as ct1
from src_region r;
```

# The result – an excerpt

◊ CT1

```
1 {"country_name": "USA", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "San Francis"},  
2 {"country_name": "USA", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "San Francis"},  
3 {"country_name": "USA", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "San Francis"},  
4 {"country_name": "USA", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "San Francis"},  
5 {"country_name": "France", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "San Fr"},  
6 {"country_name": "France", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "San Fr"},  
7 {"country_name": "France", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "San Fr"},  
8 {"country_name": "France", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "San Fr"},  
9 {"country_name": "Great Britain", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "S"},  
10 {"country_name": "Great Britain", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "S"},  
11 {"country_name": "Great Britain", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "S"},  
12 {"country_name": "Great Britain", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "S"},  
13 {"country_name": "Nederland", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "San F"},  
14 {"country name": "Nederland", "ctinfo": [{"name": "McCarthy", "phone": "(214) 555 3075", "city": "Dallas", "street": "27 Pasadena Drive"}, {"name": "Travis", "phone": "(510) 555 4448", "city": "San F"}]
```

# Generating JSON – easy with SQL

- Special values in JSON

Boolean, e.g. true/false NOT as string

```
select json_object('is_Boolean' is 'true' format json) from dual;
```

```
JSON_OBJECT('IS_BOOLEAN' IS 'TRUE' FORMATJSON)
```

```
1 {"is_Boolean":true}
```

Null values as “null” or leave them out ?

```
select json_object ('nullField' is null ) from dual;  
select json_object ('nullField' is null absent on null) from dual;
```

```
JSON_OBJECT('NULLFIELD' IS NULL)
```

```
1 {"nullField":null}
```

VS

```
JSON_OBJECT('NULLFIELD' IS NULL ABSENT ON NULL)
```

```
1 {}
```

# Helpers for your Code, Automatic :binds

- POST request body
  - :body (BLOB) or :body\_text (CLOB)
- Who is accessing the end point
  - Authenticated user is mapped to :current\_user
- HTTP Response Status :status\_code & Redirects :forward\_location
- :content\_type, :fetch\_size, :fetch\_offset, :row\_count, :page\_offset

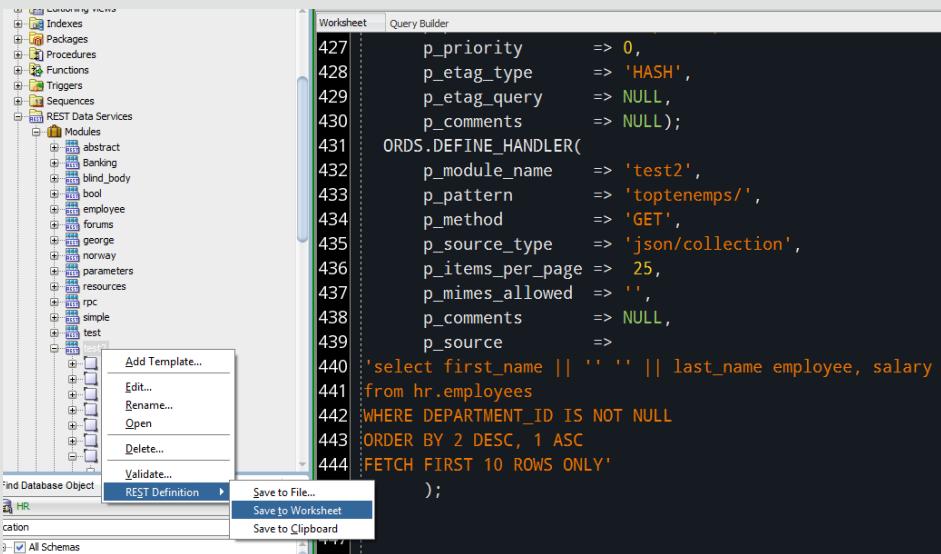
# RESTful Service Source

```
I am HR on orcl > rest export test2
-- Generated by SQLcl REST Data Services 18.1.0.0
-- Exported REST Definitions from ORDS Schema Version 17.4.1.353.06.48
-- Schema: HR   Date: Mon Jun 11 13:52:12 EDT 2018
--
BEGIN
ORDS.ENABLE_SCHEMA(
    p_enabled      => TRUE,
    p_schema       => 'HR',
    p_url_mapping_type  => 'BASE_PATH',
    p_url_mapping_pattern => 'hr',
    p_auto_rest_auth  => FALSE);

ORDS.DEFINE_MODULE(
    p_module_name   => 'test2',
    p_base_path     => '/test2/',
    p_items_per_page => 25,
    p_status        => 'PUBLISHED',
    p_comments      => NULL);

ORDS.DEFINE_TEMPLATE(
    p_module_name   => 'test2',
    p_pattern       => 'case/',
    p_priority      => 0,
    p_etag_type     => 'HASH',
    p_etag_query    => NULL,
    p_comments      => NULL);

ORDS.DEFINE_HANDLER(
    p_module_name   => 'test2',
    p_pattern       => 'case/',
    p_method        => 'GET',
    p_source_type   => 'json/query',
    p_items_per_page => 25,
    p_mimes_allowed => '',
    p_comments      => NULL,
    p_source        =>
```



# OpenAPI/Swagger

Swagger Editor   File   Edit   Generate Server   Generate Client

```
139 },
140     "/media": {
141         "post": {
142             "responses": {
143                 "200": {
144                     "description": "output of the endpoint",
145                     "schema": {
146                         "type": "object",
147                         "properties": {
148                             ...
149                         }
150                     }
151                 }
152             },
153             "parameters": [
154                 {
155                     "name": "content_type",
156                     "in": "header",
157                     "type": "string",
158                     "required": true
159                 },
160                 {
161                     "name": "title",
162                     "in": "header",
163                     "type": "string",
164                     "required": true
165                 },
166                 {
167                     "name": "payload",
168                     "in": "body",
169                     "required": true,
170                     "schema": {
171                         "$ref": "#/definitions/PAYLOAD1"
172                     }
173                 }
174             ]
175         }
176     },
177     "/media/{id)": {
178         "get": {
179             "produces": [
180                 "content/unknown"
181             ],
182             "responses": {
183                 "200": {
184                     "description": "output of the endpoint",
185                     "schema": {
186                         "type": "object",
187                         "properties": {
188                             ...
189                         }
190                     }
191                 }
192             },
193             "parameters": [
194                 {
195                     "name": "id",
196                     "in": "path",
197                 }
198             ]
199         }
200     }
201 }
```

POST /media

Parameters

Name Description

content\_type \* required  
string  
(header)

title \* required  
string  
(header)

payload \* required  
(body)

Example Value Model

```
{
    "content_type": "string",
    "title": "string",
    "body": "string"
}
```

Parameter content type application/json

Responses Response content type application/json

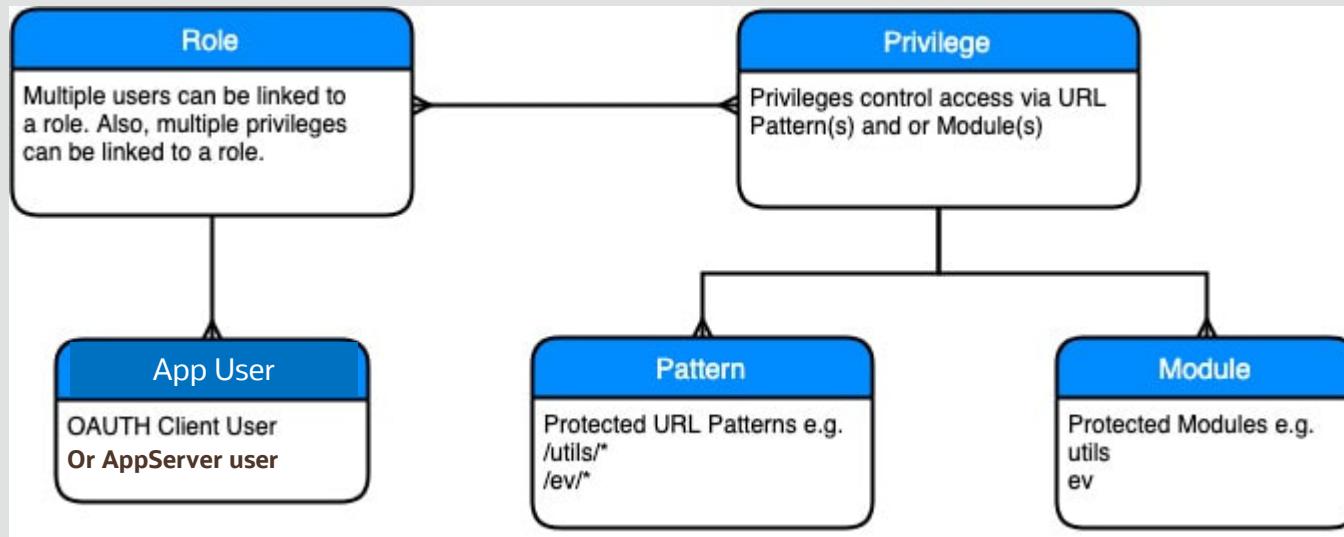
Code Description

200 output of the endpoint

Example Value Model

```
{}  
O
```

# Access Control



# The ‘Extras’

# Other things you GET out-of-the-box

- REST Enabled SQL & Type II REST JDBC Driver
- DB API
- SQL Developer Web
- SODA

# REST Enabled SQL

- SQL as a Service?
- Application/SQL
- Query on the Post Body
- Single statement or SCRIPT

POST ▾ http://localhost:8080/ords/hr/\_sql Send **200 OK** TIME 1.08 s SIZE 2.5 KB

Other ▾ Basic ▾ Query Header 1 Docs Preview ▾ Header 4 Cookie Timeline

```
1  select * from employees fetch first 3 rows only
  
```

112 ],
113 "items": [
114 {
115 "employee\_id": 100,
116 "first\_name": "Suppoørt",
117 "last\_name": "King",
118 "email": "Suppoørt",
119 "phone\_number": "515.123.4567",
120 "hire\_date": "2018-08-21T11:09:58Z",
121 "job\_id": "AD\_PRES",
122 "salary": 49243.75,
123 "commission\_pct": null,
124 "manager\_id": null,
125 "department\_id": 90,
126 "column1": null
127 },
128 {
129 "employee\_id": 101,
130 "first\_name": "Neena",
131 "last\_name": "Kochhar",
132 "email": "NKOCHHAR",
133 "phone\_number": "515.123.5368",
134 "hire\_date": "1989-09-21T04:00:00Z",
135 "job\_id": "AD\_VP",
136 "salary": 34888.29,
137 "commission\_pct": null,
138 "manager\_id": 100,
139 "department\_id": 90,
140 "column1": null
141 },
142 {
143 "employee\_id": 102,
144 "first\_name": "Lex",
145 "last\_name": "De Haan",
146 "email": "LDEHAAN",
147 "phone\_number": "515.123.4569",
148 "hire\_date": "1993-01-13T05:00:00Z",
149 "job\_id": "AD\_VP",
150 "salary": 34888.29,
151 "commission\_pct": null

```
<entry key="restEnabledSql.active">true</entry>
```

# No SQLNet Access? Try our REST Driver

```
I am on > connect dev/oracle@jdbc:oracle:orest:@http://localhost:8888/ords/ords_demo/
Connected.

Session altered.

DDL Option PRETTY on
DDL Option CONSTRAINTS off
DDL Option REF_CONSTRAINTS off
DDL Option PARTITIONING on
DDL Option TABLESPACE on
DDL Option SEGMENT_ATTRIBUTES off
DDL Option STORAGE off
'HELLOJEFF'
hello jeff

I am ORDS DEMO on jdbc:oracle:orest:@http://localhost:8888/ords/ords_demo/ >
```

- `jdbc:oracle:orest:`
- Single JAR on SQLcl downloads page

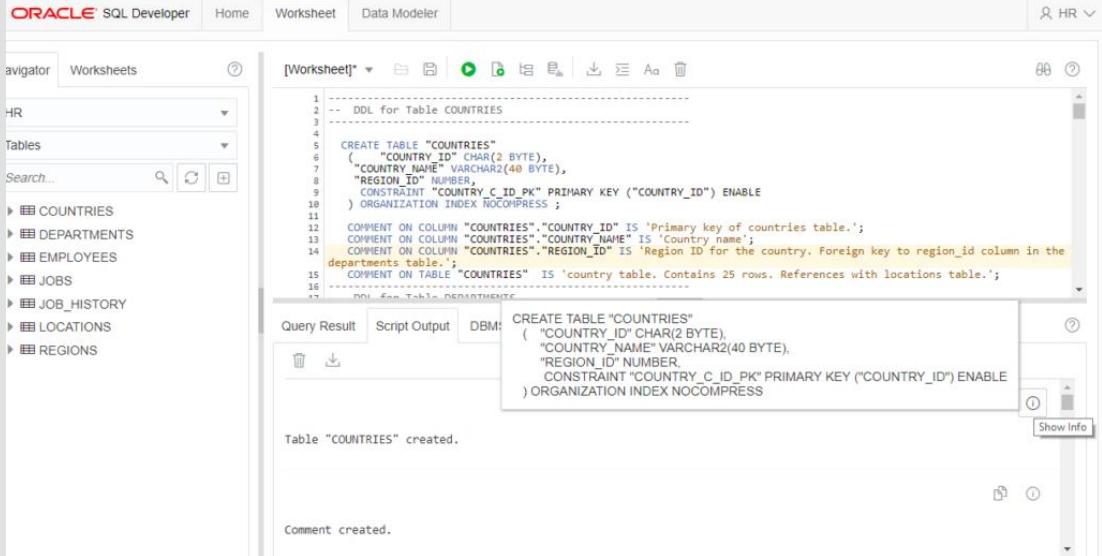
# New for ORDS 19.4 – SQL Developer Web

- Requires the DB API & REST Enabled SQL
- Database User Authentication/REST Enabled Schemas Only
- HTML5/js (Oracle JET)
- 11gR2 and higher

# Features

- SQL Worksheet
- Create/Edit dialogs
- Excel/CSV Imports
- Schema Diagramming
- REST IDE
- JSON Workshop
- Performance Hub

ASH Analytics  
Real Time SQL Monitoring



The screenshot shows the Oracle SQL Developer interface with the 'Worksheet' tab selected. The central area displays the following DDL code for creating the 'COUNTRIES' table:

```
1 -- DDL for Table COUNTRIES
2
3
4
5 CREATE TABLE "COUNTRIES"
6   (
7     "COUNTRY_ID" CHAR(2 BYTE),
8     "COUNTRY_NAME" VARCHAR2(40 BYTE),
9     "REGION_ID" NUMBER,
10    CONSTRAINT "COUNTRY_C_ID_PK" PRIMARY KEY ("COUNTRY_ID") ENABLE
11 ) ORGANIZATION INDEX NOCOMPRESS ;
12
13 COMMENT ON COLUMN "COUNTRIES"."COUNTRY_ID" IS 'Primary key of countries table.';
14 COMMENT ON COLUMN "COUNTRIES"."COUNTRY_NAME" IS 'Country name';
15 COMMENT ON COLUMN "COUNTRIES"."REGION_ID" IS 'Region ID for the country. Foreign key to region_id column in the
16 departments table.';
17 COMMENT ON TABLE "COUNTRIES" IS 'country table. Contains 25 rows. References with locations table.';
```

The 'Script Output' tab at the bottom shows the result of running the DDL:

```
CREATE TABLE "COUNTRIES"
(
  "COUNTRY_ID" CHAR(2 BYTE),
  "COUNTRY_NAME" VARCHAR2(40 BYTE),
  "REGION_ID" NUMBER,
  CONSTRAINT "COUNTRY_C_ID_PK" PRIMARY KEY ("COUNTRY_ID") ENABLE
) ORGANIZATION INDEX NOCOMPRESS
```

Below the output, a message indicates the table was created:

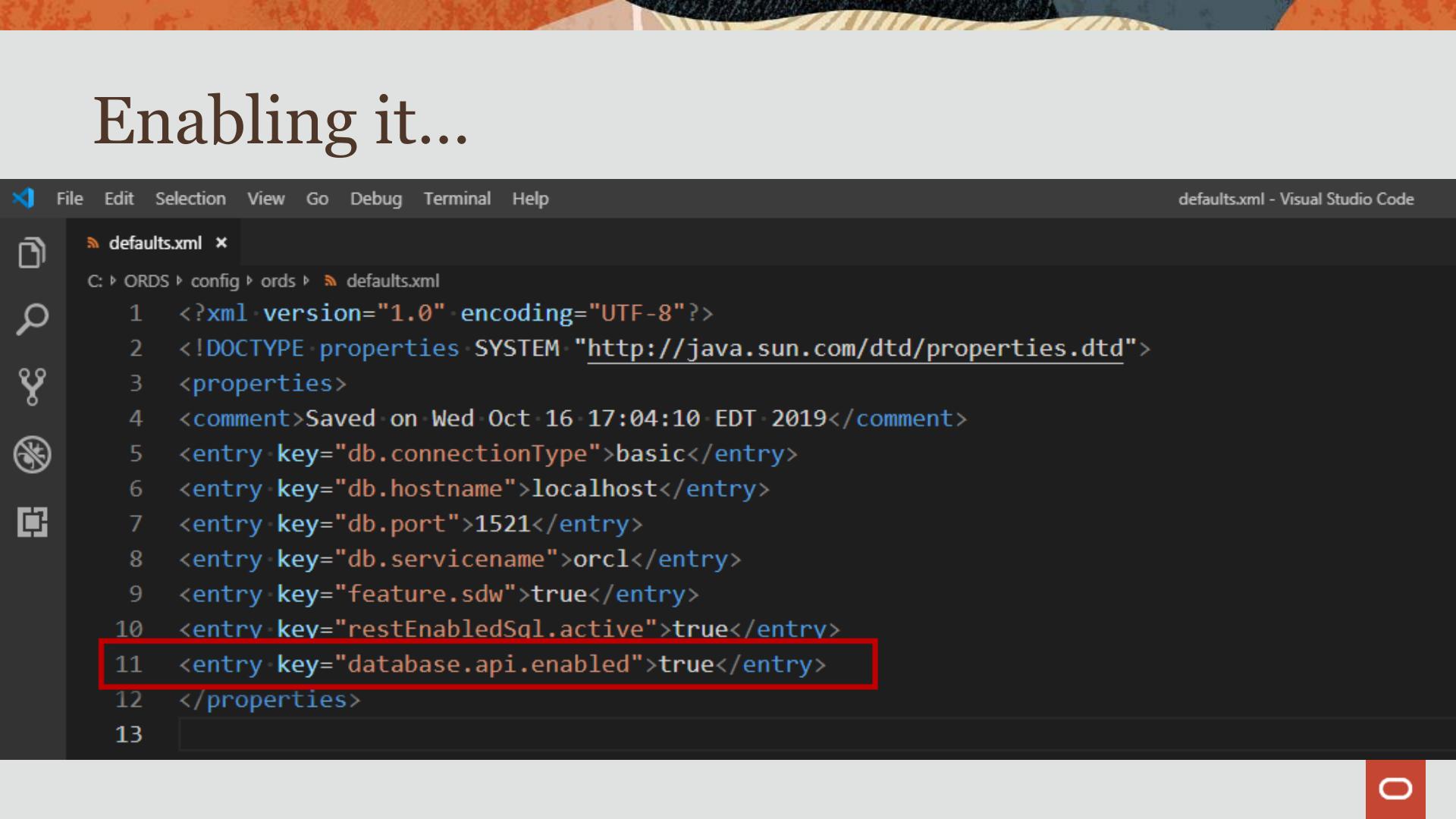
Table "COUNTRIES" created.

# Database Management APIs

*Use HTTPS to manage and monitor your Oracle Database*

- Supports 11gR2, 12, 18, & 19c
- General, Data Dictionary, Monitoring, Performance, & PDB Lifecycle Management
- Automatic – no code to write, just enable and use it

# Enabling it...



File Edit Selection View Go Debug Terminal Help

defaults.xml - Visual Studio Code

defaults.xml

C:\ ORDS\config\ords\defaults.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
3  <properties>
4  <comment>Saved on Wed Oct 16 17:04:10 EDT 2019</comment>
5  <entry key="db.connectionType">basic</entry>
6  <entry key="db.hostname">localhost</entry>
7  <entry key="db.port">1521</entry>
8  <entry key="db.servicename">orcl</entry>
9  <entry key="feature.sdw">true</entry>
10 <entry key="restEnabledSql.active">true</entry>
11 <entry key="database.api.enabled">true</entry>
12 </properties>
13 
```

O

# Auth

- Authenticate using REST Enabled Oracle USERS

GET `/active_sessions_history/?q={"$eq":{"session_type":"FOREGROUND"}}` Send

Body

401 Unauthorized TIME 94 ms SIZE 16.1 KB

Preview Header 2 Cookie Timeline

USERNAME hr

PASSWORD

ENABLED

 ORACLE REST Data Services

**401** Unauthorized

2019-10-30T14:18:15.284Z / WFnGkf7lz5ZYFxvlhHv2rQ

Access to this resource is protected. Please [sign in](#) to access this resource.

# User Setup

## Option #1, Mid Tier User:

```
c:\ORDS\19.1-GA>java -jar ords.war user jeff "SQL Administrator"
Enter a password for user jeff:
Confirm password for user jeff:
Apr 23, 2019 9:38:54 AM oracle.dbtools.standalone.ModifyUser execute
INFO: Created user: jeff in file: C:\ORDS\config\ords\credentials
```

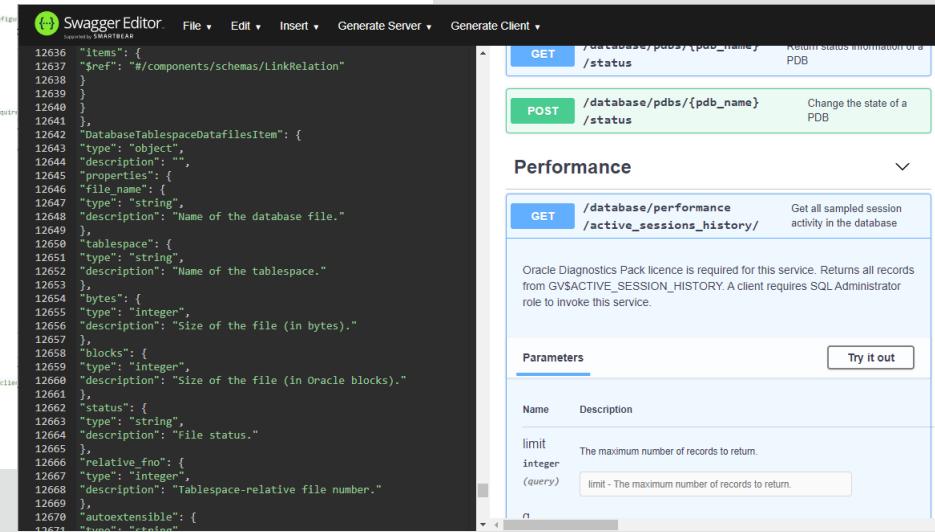
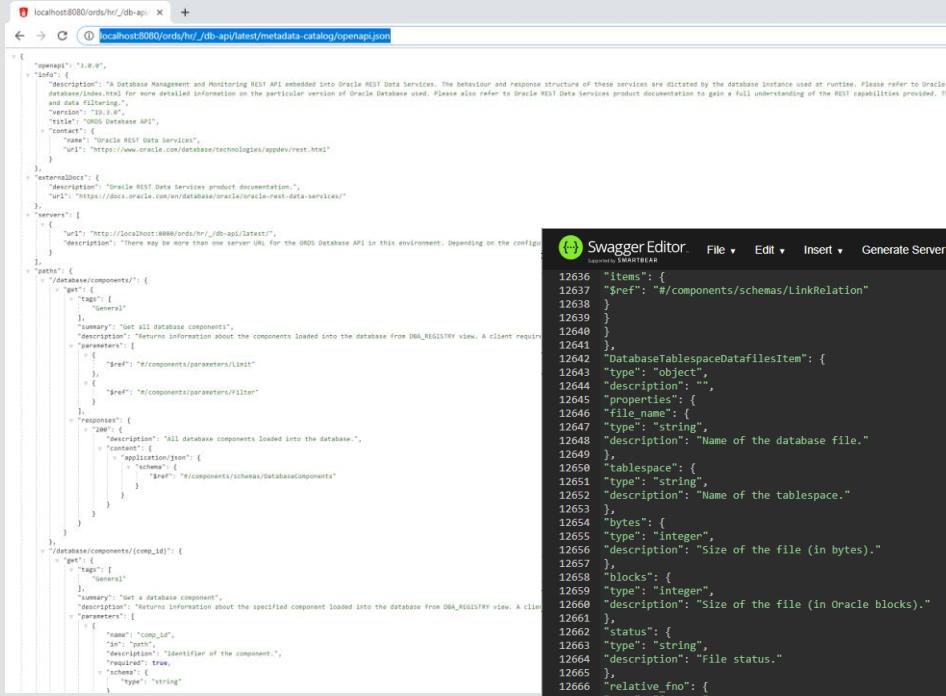
## Option #2, DB User:

```
1 BEGIN -- block to be executed as HR
2   ords.enable_schema (
3     p_enabled          => true,
4     p_schema            => 'HR',
5     p_url_mapping_type => 'BASE_PATH',
6     p_url_mapping_pattern => 'hr',
7     p_auto_rest_auth    => true
8   );
9
10  COMMIT;
11 END;
12 /
13
14 GRANT dba TO hr; -- or PDB_DBA for 12c+
```

# API Docs

- Generate it yourself
- Go to our copy on [docs.oracle.com](https://docs.oracle.com)

# GET the Open API Docs {Swagger 3}



# DB User Privilages?

Swagger Editor. Supported by SMARTBEAR

```
12636 "items": {  
12637   "$ref": "#/components/schemas/LinkRelation"  
12638 }  
12639 }  
12640 }  
12641 },  
12642 "DatabaseTablespaceDatafilesItem": {  
12643   "type": "object",  
12644   "description": "",  
12645   "properties": {  
12646     "file_name": {  
12647       "type": "string",  
12648       "description": "Name of the database file."  
12649     },  
12650     "tablespace": {  
12651       "type": "string",  
12652       "description": "Name of the tablespace."  
12653     },  
12654     "bytes": {  
12655       "type": "integer",  
12656       "description": "Size of the file (in bytes)."  
12657     },  
12658     "blocks": {  
12659       "type": "integer",  
12660       "description": "Size of the file (in Oracle blocks)."  
12661     },  
12662     "status": {  
12663       "type": "string",  
12664       "description": "File status."  
12665     },  
12666     "relative_fno": {  
12667       "type": "integer",  
12668       "description": "Tablespace-relative file number."  
12669     },  
12670     "autoextensible": {  
12671       "type": "string",  
12672     }  
12673   }  
12674 }
```

**GET /database/pdb/{pdb\_name}/status** Return status information of a PDB

**POST /database/pdb/{pdb\_name}/status** Change the state of a PDB

## Performance

**GET /database/performance/active\_sessions\_history/** Get all sampled session activity in the database

Oracle Diagnostics Pack licence is required for this service. Returns all records from GV\$ACTIVE\_SESSION\_HISTORY. A client requires SQL Administrator role to invoke this service.

**Parameters**

**Try it out**

Name	Description
limit	The maximum number of records to return.
integer	
(query)	limit - The maximum number of records to return.
o	

# Oracle Docs

A screenshot of a web browser showing the Oracle REST Data Services documentation. The URL in the address bar is `ocs.oracle.com/en/database/oracle/oracle-rest-data-services/index.html`. The page title is "Help Center". On the left sidebar, under "Related Products", "REST APIs for Oracle Database" is highlighted with a red box. Below it, "Oracle SQL Developer" is listed. The main content area features a section titled "Oracle REST Data Services" with a sub-section "Oracle REST Data Services Releases". A dropdown menu shows "Oracle REST Data Services 19.2" is selected. A "Get Started" button is visible at the bottom.

## ORDS Docs

A screenshot of the Oracle Database 19c Help Center. The URL in the address bar is `docs.oracle.com/en/database/Oracle/Database/Release-19`. The page title is "Help Center". The left sidebar has a "Get Started" section with "What's New" and a list of topics: "Install and Upgrade", "Administration", "Development", "Security", "Performance", "Clustering", "High Availability", "Data Warehousing", "Spatial and Graph", and "Distributed Data". Under "Books", "REST API Reference" is highlighted with a red box. The main content area features a section titled "Oracle Database 19c" with a "Get Started" button. It includes four cards: "Learn About Oracle Database" (with sub-links for introduction, SQL, and what's new), "Oracle Multitenant" (with sub-links for about, managing environment, and managing privileges), "Development" (with sub-links for applications, Java, JSON, and JSON-based applications), and "Oracle Database In-Memory" (with sub-links for introduction, column store, and populating).

## Database Docs

# ORDS on YouTube

## Other Resources

- SlideShare
- Blogs
- GitHub
- Articles

UKOUG Scene [Why REST, and What's in it or Me?](#)

Oracle Mag [AUTO REST & REST Enabled SQL](#)

- And don't forget [Oracle-Base!](#)



Oracle REST Data Services

4 videos • 1,553 views • Updated 5 days ago



Jeff Smith

A screenshot of the ThatJeffSmith YouTube channel page. It shows a list of four video thumbnails under the heading "Oracle REST Data Services Product Walk-through and Demonstration".

Video Title	Length	Uploader
Oracle REST Data Services Development with Oracle SQL Developer	16:59	Jeff Smith
Generating Swagger Doc for Your Oracle Database RESTful Services	15:07	Jeff Smith
Building a Web Service for up and downloading files to Oracle Database	25:11	Jeff Smith