ORDS

DatabaseRESTAPI
https://oracle.com/rest

Jeff Smith
Senior Principal Product Manager
Jeff.d.smith@oracle.com || @thatjeffsmith
Database Tools, Oracle Corp

ORACLE

Not Just THAT SQLDev Guy…

- Database Development Tools team
- Product manager/story teller (SQLDev, SQLcl, Data Modeler, ORDS)
- In the time before time…
  - DBA, Unix/Apache, Perl, software support, preSales
- I bother people online as @thatjeffsmith
- Contact me for a free remote presentation for your group/company

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Oracle REST Data Services

Formerly Known as the APEX Listener

ORACLE

Get audience sympathy/laughing early

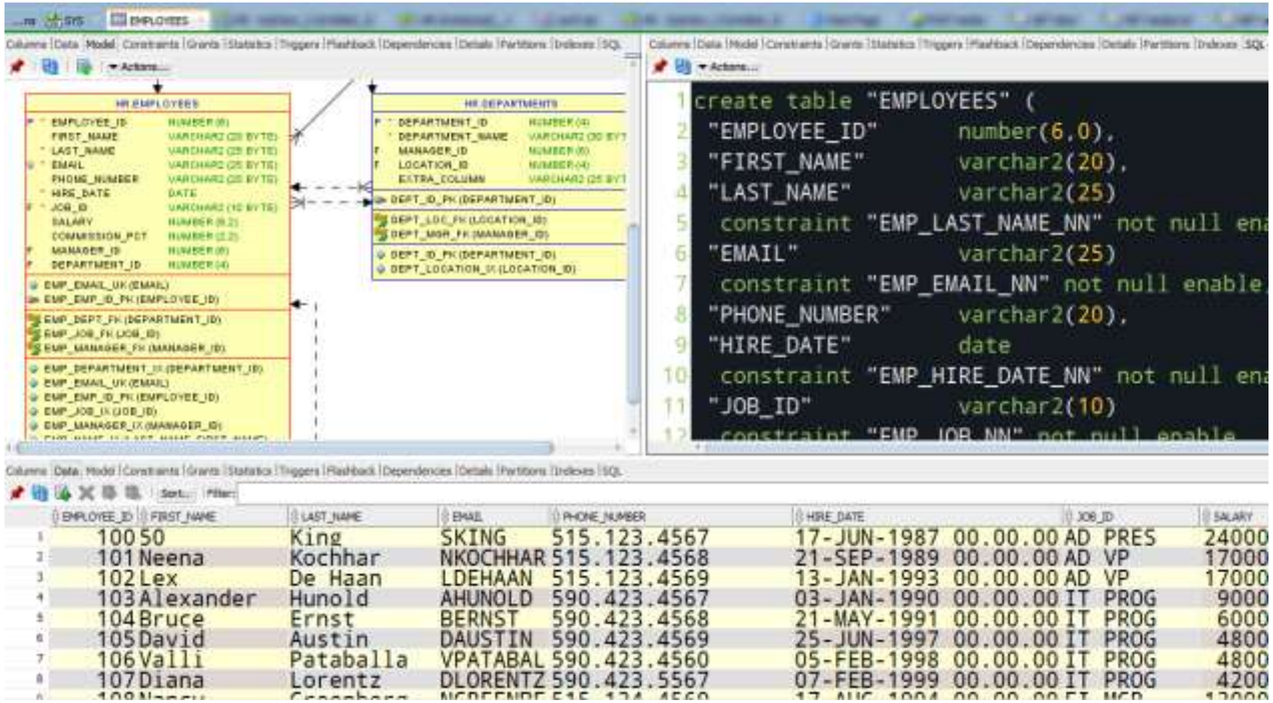
- I’m gonna need a REST API for all of our Oracle data…
- …on Monday

Your Requirements

- No connections to the db (directly at least)
- No DB authentication (let the app/web tier handle that)
- Link driven (stateless)
- Plays nice with the rest of the apps
  - Standard response/error codes
  - JSON
  - Well documented

Employee Data => /app123/hr/employees/ /app123/hr/employees/:id

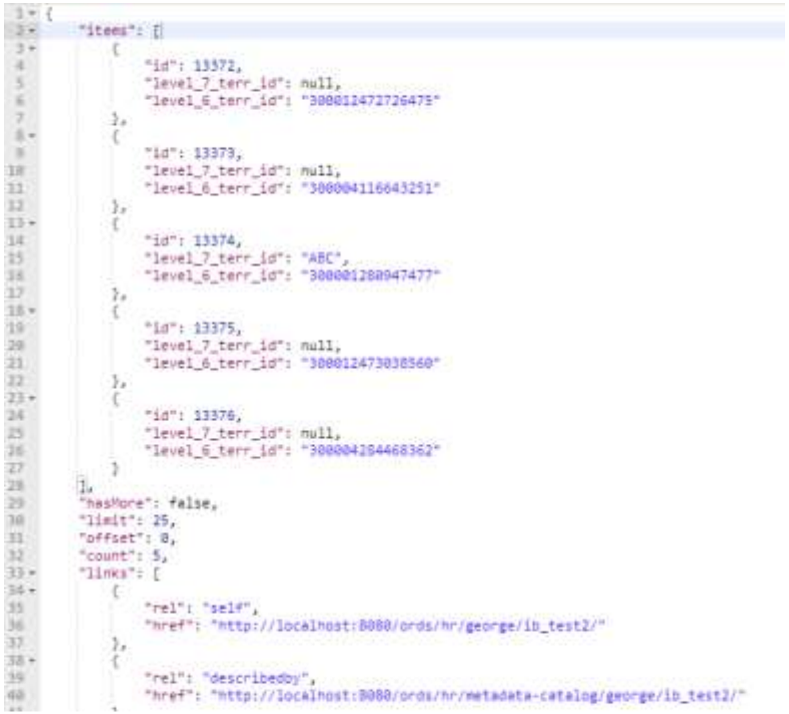
- GET
- PUT
- POST
- DELETE
- Metadata
- DOCS


HR.EMPLOYEES
EMPLOYEE_ID NUMBER (6)
FIRST_NAME VARCHAR2 (20 BYTE)
LAST_NAME VARCHAR2 (25 BYTE)
EMAIL VARCHAR2 (25 BYTE)
PHONE_NUMBER VARCHAR2 (20 BYTE)
HIRE_DATE DATE
JOB_ID VARCHAR2 (10 BYTE)
SALARY NUMBER (8,2)
COMMISSION_PCT NUMBER (2,2)
MANAGER_ID NUMBER (6)
DEPARTMENT_ID NUMBER (4)
EMP_EMAIL_UK (EMAIL)
EMP_EMP_ID_PK (EMPLOYEE_ID)
EMP_DEPT_FK (DEPARTMENT_ID)
EMP_JOB_FK (JOB_ID)
EMP_MANAGER_FK (MANAGER_ID)
EMP_DEPARTMENT_IX (DEPARTMENT_ID)
EMP_EMAIL_UK (EMAIL)
EMP_EMP_ID_PK (EMPLOYEE_ID)
EMP_JOB_IX (JOB_ID)
EMP_MANAGER_IX (MANAGER_ID)
HR.DEPARTMENTS
DEPARTMENT_ID NUMBER (4)
DEPARTMENT_NAME VARCHAR2 (30 BYT
MANAGER_ID NUMBER (6)
LOCATION_ID NUMBER (4)
EXTRA_COLUMN VARCHAR2 (25 BYT
DEPT_ID_PK (DEPARTMENT_ID)
DEPT_LOC_FK (LOCATION_ID)
DEPT_MGR_FK (MANAGER_ID)
DEPT_ID_PK (DEPARTMENT_ID)
DEPT_LOCATION_IX (LOCATION_ID)
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Actions...
Sort... Filter:

```
create table "EMPLOYEES" (
 "EMPLOYEE_ID"       number(6,0),
 "FIRST_NAME"        varchar2(20),
 "LAST_NAME"         varchar2(25)
  constraint "EMP_LAST_NAME_NN" not null ena
 "EMAIL"             varchar2(25)
  constraint "EMP_EMAIL_NN" not null enable
 "PHONE_NUMBER"      varchar2(20),
 "HIRE_DATE"         date
  constraint "EMP_HIRE_DATE_NN" not null ena
 "JOB_ID"            varchar2(10)
  constraint "EMP_JOB_NN" not null enable
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	100	50	King	SKING	515.123.4567	17-JUN-1987 00.00.00	AD_PRES	24000
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-1989 00.00.00	AD_VP	17000
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-1993 00.00.00	AD_VP	17000
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-1990 00.00.00	IT_PROG	9000
5	104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-1991 00.00.00	IT_PROG	6000
6	105	David	Austin	DAUSTIN	590.423.4569	25-JUN-1997 00.00.00	IT_PROG	4800
7	106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-1998 00.00.00	IT_PROG	4800
8	107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-1999 00.00.00	IT_PROG	4200

{JSON} or JavaScript Object Notation



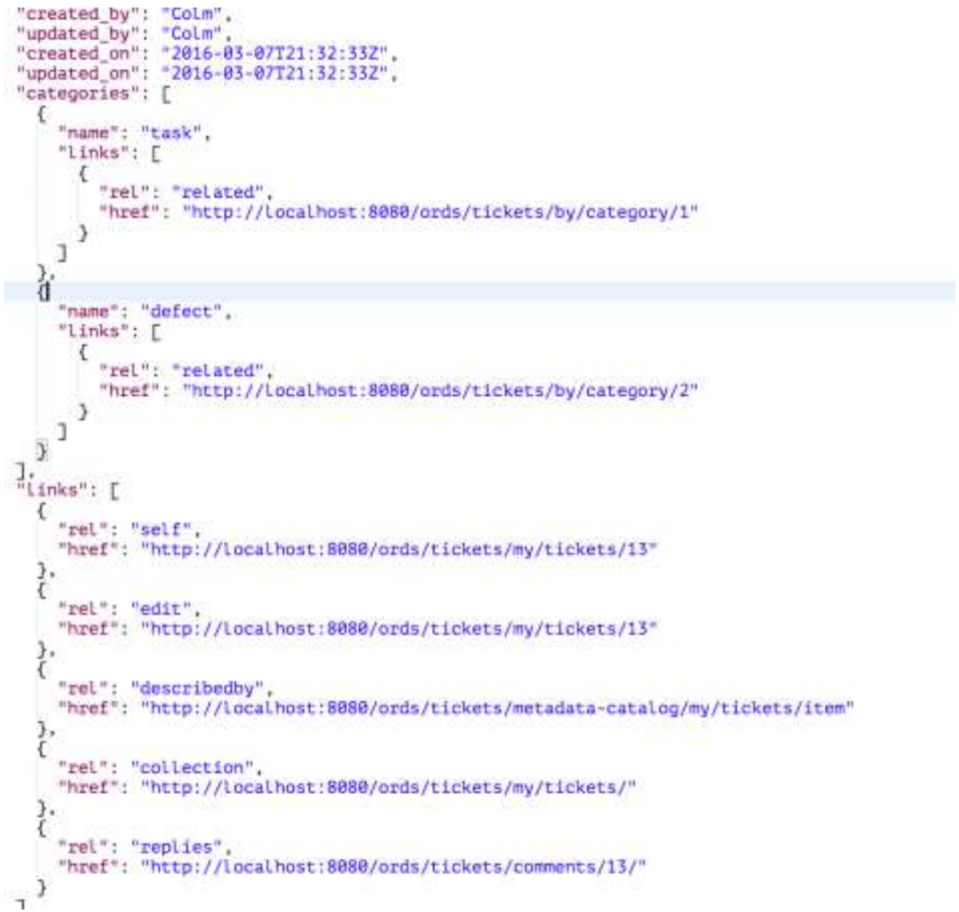
```
{
    "items": [
        {
            "id": 13372,
            "level_7_terr_id": null,
            "level_6_terr_id": "300012472726475"
        },
        {
            "id": 13373,
            "level_7_terr_id": null,
            "level_6_terr_id": "300004116643251"
        },
        {
            "id": 13374,
            "level_7_terr_id": "ABC",
            "level_6_terr_id": "300001280947477"
        },
        {
            "id": 13375,
            "level_7_terr_id": null,
            "level_6_terr_id": "300012473038560"
        },
        {
            "id": 13376,
            "level_7_terr_id": null,
            "level_6_terr_id": "300004284468362"
        }
    ],
    "hasMore": false,
    "limit": 25,
    "offset": 0,
    "count": 5,
    "links": [
        {
            "rel": "self",
            "href": "http://localhost:8080/ords/hr/george/ib_test2/"
        },
        {
            "rel": "describedby",
            "href": "http://localhost:8080/ords/hr/metadata-catalog/george/ib_test2/"
```

- JAY-sun
- Not just for JavaScript
- Skinnier than XML
- Flexible
  - Easily adapted to represent database objects & data
- Link-friendly

Tabular

Nested

Hyperlink



```
"created_by": "Colm",
"updated_by": "Colm",
"created_on": "2016-03-07T21:32:33Z",
"updated_on": "2016-03-07T21:32:33Z",
"categories": [
  {
    "name": "task",
    "links": [
      {
        "rel": "related",
        "href": "http://localhost:8080/ords/tickets/by/category/1"
      }
    ]
  },
  {
    "name": "defect",
    "links": [
      {
        "rel": "related",
        "href": "http://localhost:8080/ords/tickets/by/category/2"
      }
    ]
  }
],
"links": [
  {
    "rel": "self",
    "href": "http://localhost:8080/ords/tickets/my/tickets/13"
  },
  {
    "rel": "edit",
    "href": "http://localhost:8080/ords/tickets/my/tickets/13"
  },
  {
    "rel": "describedby",
    "href": "http://localhost:8080/ords/tickets/metadata-catalog/my/tickets/item"
  },
  {
    "rel": "collection",
    "href": "http://localhost:8080/ords/tickets/my/tickets/"
  },
  {
    "rel": "replies",
    "href": "http://localhost:8080/ords/tickets/comments/13/"
  }
]
```

But we never use SQL, only PL/SQL Table APIs!

- No worries!
- We auto-magically handle PL/SQL too
- RPC -> HTTPS POST
- Input parameters passed by POST header
- Responses & Results passed back, also in {JSON}

*** Bryn/THICKDB APPROVED ***

Why should a DBA or PL/SQL Dev care about REST?

- **RESTful** web services are a way of providing interoperability between computer systems on the Internet
- REST often treated as a Religion, BUT…
- …provides a predictable model for delivering services
- We aim for pure REST, but don't let that pursuit get in the way of practicality

REpresentational State Transfer (REST)

It relies on a ***stateless, client-server, cacheable communications protocol*** -- and in virtually all cases, the HTTP(S!) protocol is used.

REST is *an architecture style* for designing networked applications. The idea is that, rather than using complex mechanisms such as CORBA, RPC or SOAP to connect between machines, simple HTTP is used to make calls between machines. *(rest.elkstein.org)*

REST is Easy

- Small uniform set of operations: GET, POST, PUT, DELETE (CRUD!)
- Small set of uniform status codes URLs & hyperlinks encourage stateless behavior
- Text based protocol with simple request/response model

About those Codes

- 1xx Informational responses.
- 2xx Success.
- 3xx Redirection.
- 4xx Client errors.
- 5xx Server errors.

{REST} The Architectural Style of the Web

- Model resources, not actions:
  - GET /ords/hr/employees/ - GOOD
  - GET /ords/hr/delete_emp/ - BAD
  - DELETE /ords/hr/employees/97 - GOOD
- Uniform operations on all resources:
  - GET, POST, PUT, DELETE, OPTIONS, HEAD
- Stateless requests, state transitions communicated via hyper-links.

Resource Collection Pattern

- MASTER RESOURCE: called the Collection URI:

https://example.com/ords/hr/employees/

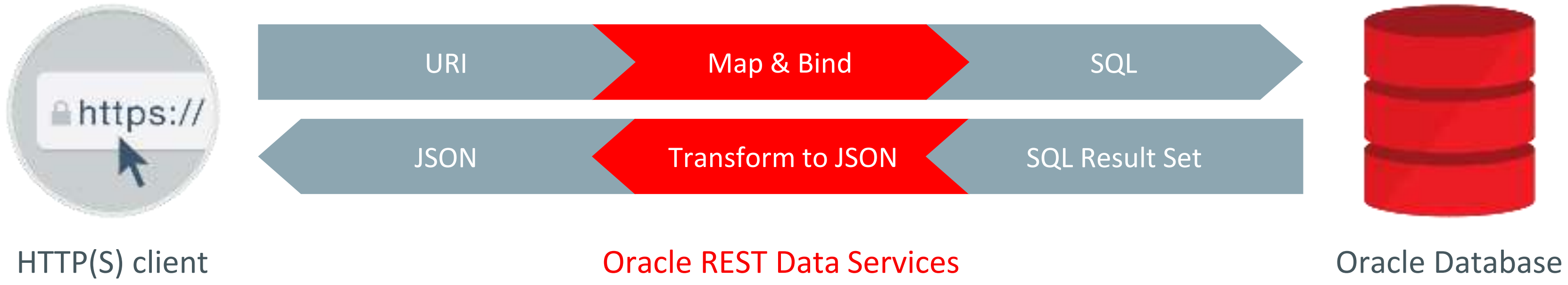
- DETAIL RESOURCE; called the Item URI:

https://example.com/ords/hr/employees/:id

The Verbs

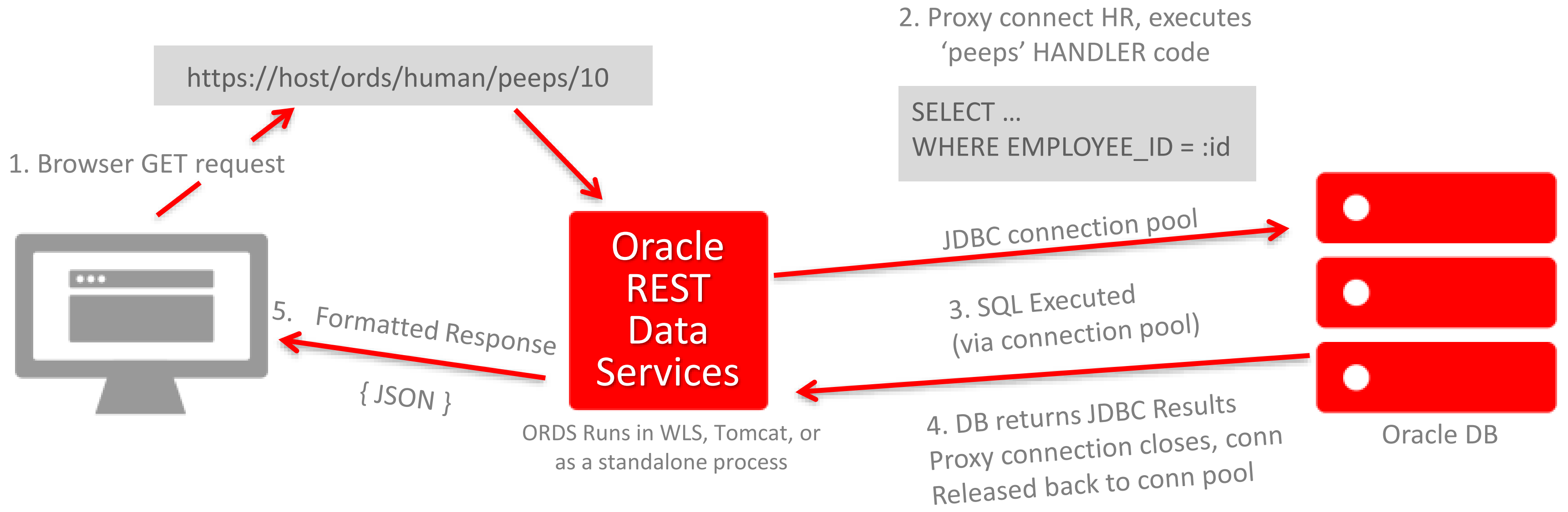
Method	Purpose	Classification	Database Operation
GET	Retrieve resource	Safe, Idempotent	SELECT
PUT	Create or replace resource	Idempotent	MERGE, UPDATE
DELETE	Delete resource	Idempotent	DELETE
POST	Anything. Normally create	Unsafe	INSERT

How do we Marry this Architectural Style to the DB?



- Java JEE mid tier application, e.g., WebLogic, Tomcat, Glassfish (deprecated)
  - Also supports "Standalone" mode for development
- For input, maps/binds URI to SQL and PL/SQL
- For output, transforms results to JSON and other formats

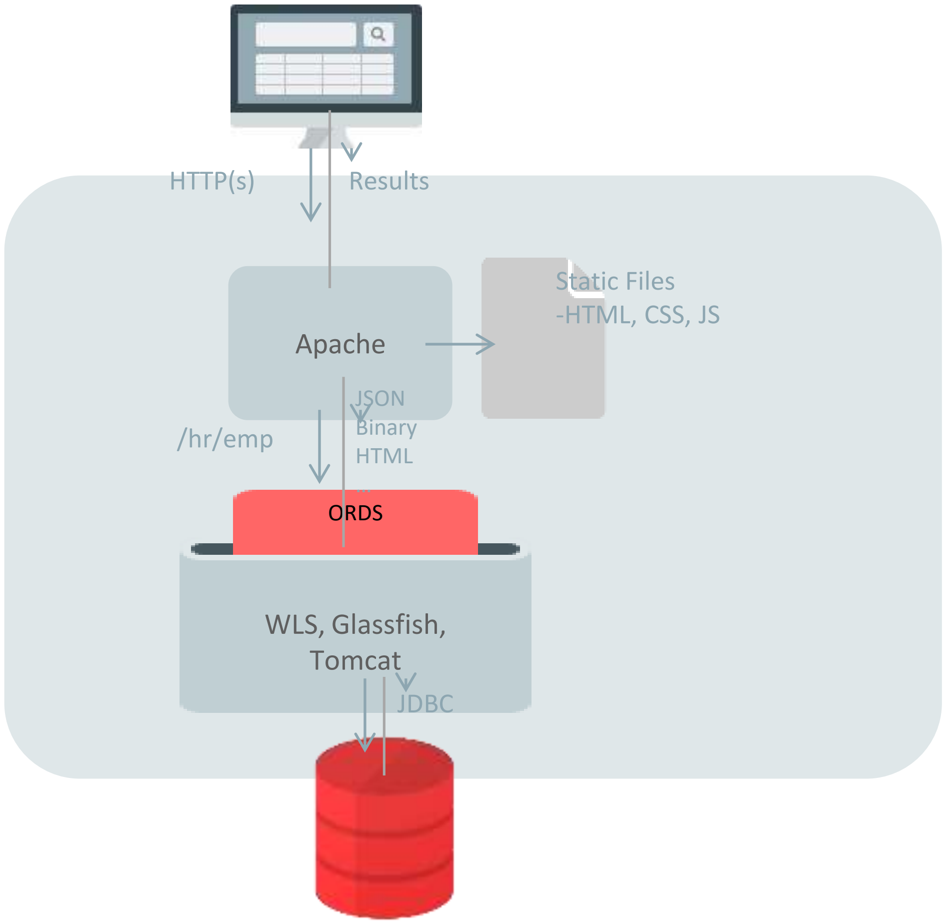
Anatomy of a RESTful Service Transaction



https://host/ords/human/peeps/10
1. Browser GET request
2. Proxy connect HR, executes 'peeps' HANDLER code
SELECT …
WHERE EMPLOYEE_ID = :id
Oracle REST Data Services
JDBC connection pool
3. SQL Executed (via connection pool)
5. Formatted Response
{ JSON }
ORDS Runs in WLS, Tomcat, or as a standalone process
4. DB returns JDBC Results Proxy connection closes, conn Released back to conn pool
Oracle DB

Typical Architecture

- Standard webserver layout
- Implements Java Servlet
- Deploys to WLS, Tomcat, Glassfish
- OR Embedded Jetty (standalone)



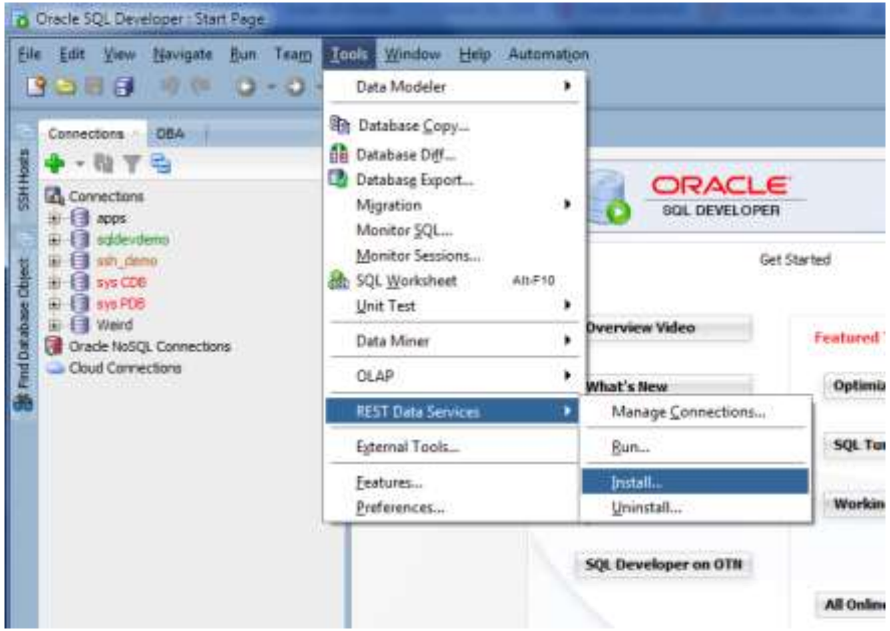
And it’s Easy!

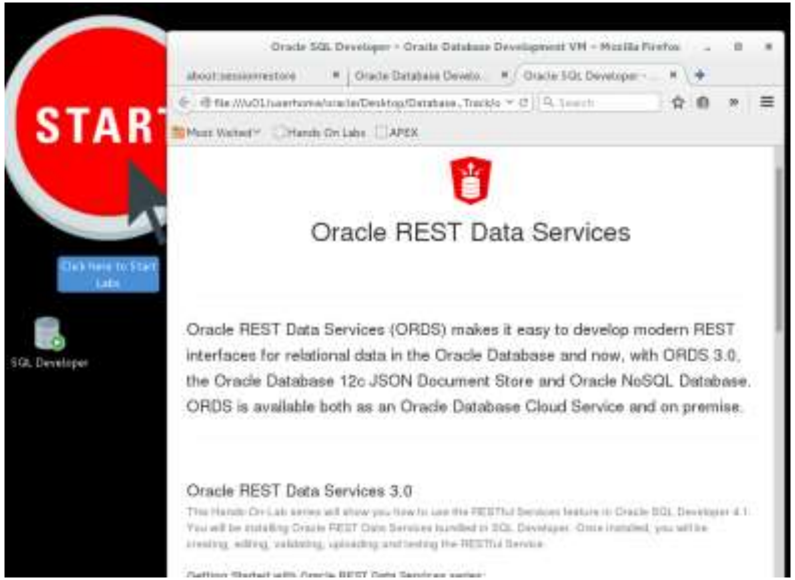
I had been looking for a chance to perform a POC, so I proposed I just provide him with a couple of web services. ***In under a day I had functioning web services in place for him to consume.*** *There’s been refinements to them and new ones developed since then. We are now exploring where else we could leverage this technology.*

- a REAL customer talking about ORDS

ORDS – How do I get started???

Use SQL Developer to install & run. Use Hands On Labs to learn.




START
Click here to Start Labs
SQL Developer
Oracle SQL Developer - Oracle Database Development VM - Mozilla Firefox
Most Visited
Hands On Labs
APEX

Oracle REST Data Services

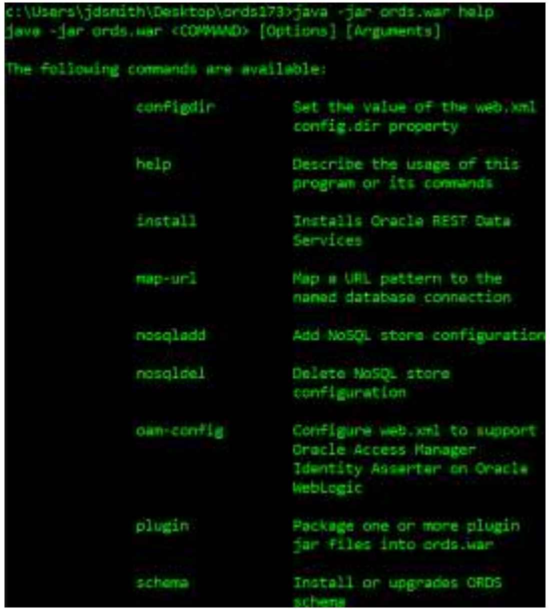
Oracle REST Data Services (ORDS) makes it easy to develop modern REST interfaces for relational data in the Oracle Database and now, with ORDS 3.0, the Oracle Database 12c JSON Document Store and Oracle NoSQL Database. ORDS is available both as an Oracle Database Cloud Service and on premise.

Oracle REST Data Services 3.0

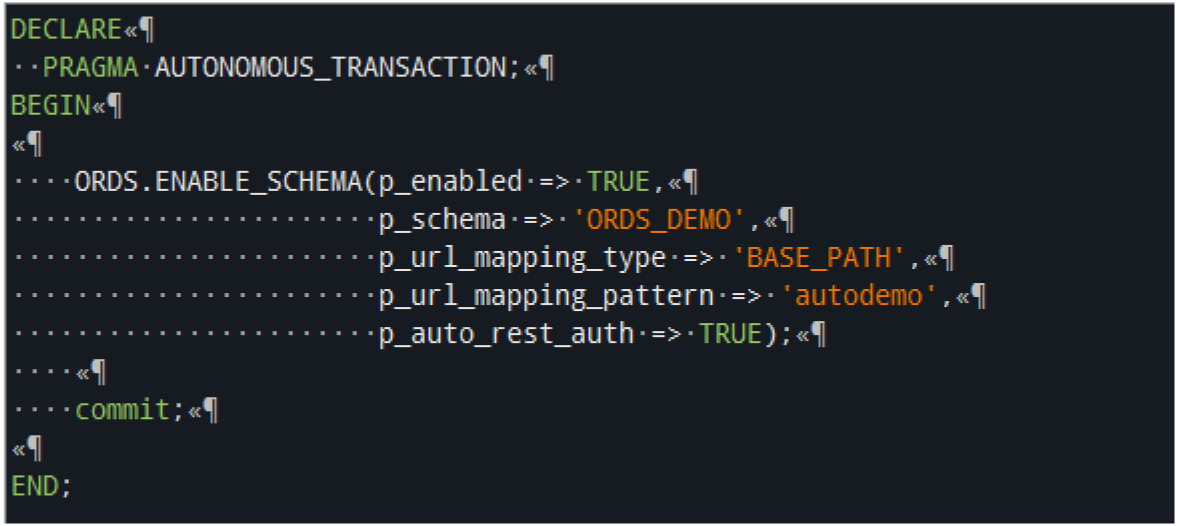
This Hands-On-Lab series will show you how to use the RESTful Services feature in Oracle SQL Developer 4.1. You will be installing Oracle REST Data Services bundled in SQL Developer. Once installed, you will be creating, editing, validating, uploading and testing the RESTful Service.

Getting Started with Oracle REST Data Services series:

Full Command-Line Interface & PL/SQL API



```
c:\Users\jdsmith\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
```



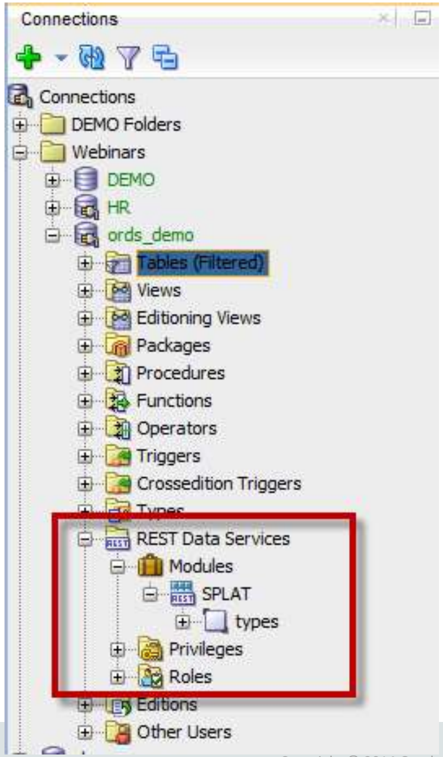
```
DECLARE«¶
··PRAGMA·AUTONOMOUS_TRANSACTION;«¶
BEGIN«¶
«¶
····ORDS.ENABLE_SCHEMA(p_enabled·=>·TRUE,«¶
·······················p_schema·=>·'ORDS_DEMO',«¶
·······················p_url_mapping_type·=>·'BASE_PATH',«¶
·······················p_url_mapping_pattern·=>·'autodemo',«¶
·······················p_auto_rest_auth·=>·TRUE);«¶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

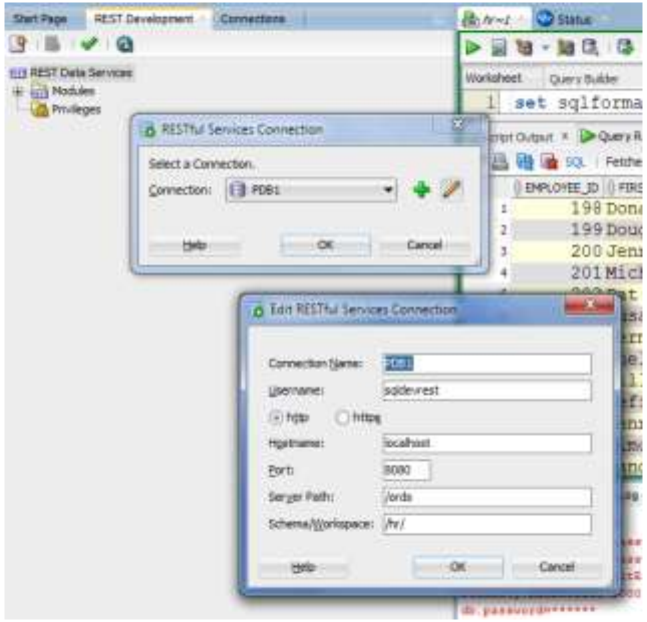
Enable a Schema

- Services are EXECUTED as the REST enabled schema USER …
- … via ORDS_PUBLIC_USER Proxy Connect
- What your session can see & do = straight forward & predictable
- Secured/Authorization outside the database

Develop RESTFul Services: PL/SQL, GUI, or even REST

```
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       => FALSE);

  ORDS.DEFINE_MODULE(
      p_module_name    => 'SPLAT',
      p_base_path      => '/splat/',
      p_items_per_page =>  25,
      p_status         => 'PUBLISHED',
      p_comments       => NULL);
  ORDS.DEFINE_TEMPLATE(
      p_module_name    => 'SPLAT',
      p_pattern        => 'types',
      p_priority       => 0,
      p_etag_type      => 'HASH',
      p_etag_query     => NULL,
      p_comments       => NULL);
  ORDS.DEFINE_HANDLER(
      p_module_name    => 'SPLAT',
      p_pattern        => 'types',
      p_method         => 'GET',
      p_source_type    => 'json/query',
      p_items_per_page =>  25,
      p_mimes_allowed  => '',
      p_comments       => NULL,
      p_source         =>
'select * from d_types'
      );
  COMMIT;
END;
```





set sqlforma

RESTful Services Connection

Select a Connection.

Connection: PDB1

Edit RESTful Services Connection

Connection Name: PDB1
Username: sqldevrest
[✓ http] https
Hostname: localhost
Port: 8080
Server Path: /ords
Schema/Workspace: /hr/

	EMPLOYEE_ID	FIRS
1	198	Dona
2	199	Doug
3	200	Jenn
4	201	Mich

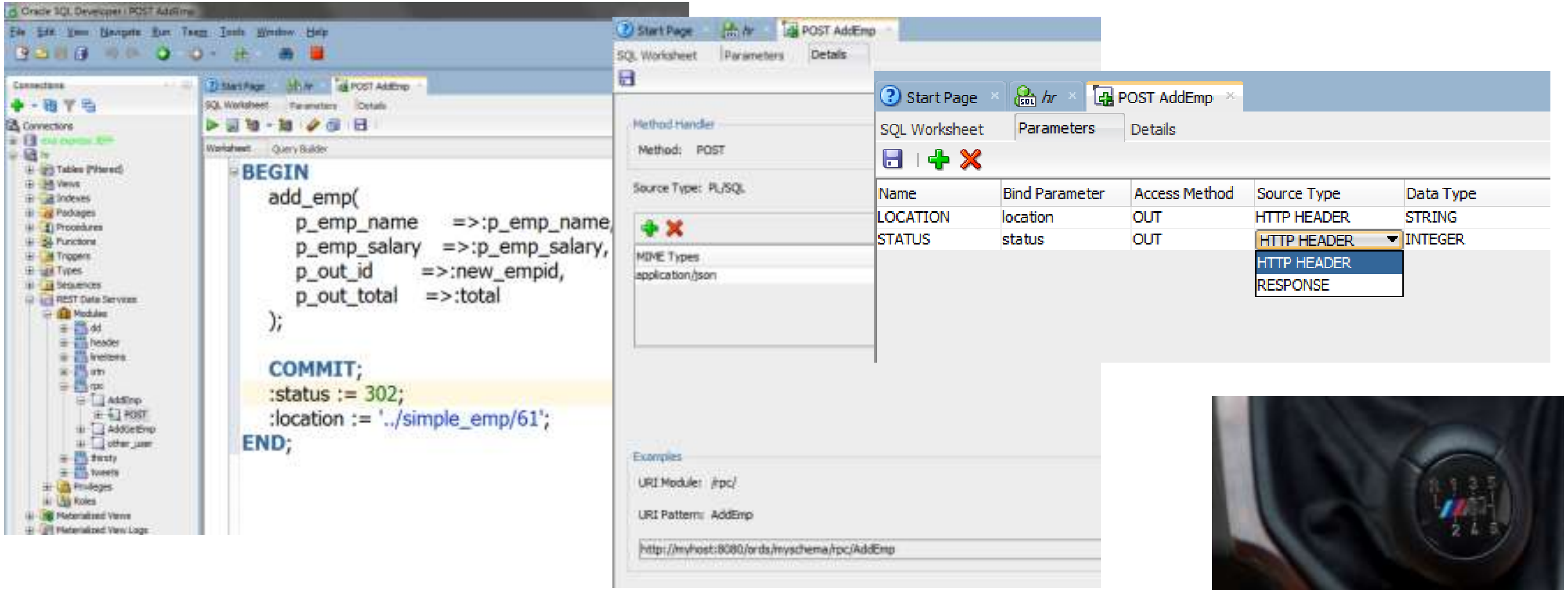
ORACLE

Code and No/Low Code RESTful Service Options

- No need to know Java
- Database developers (PLSQL & SQL) get started quickly
- PL/SQL API
- GUI/IDE Support (SQL Developer!)

VS

Manual – You Define Modules/URIs/Handlers/the Code



```
BEGIN
   add_emp(
      p_emp_name     =>:p_emp_name,
      p_emp_salary   =>:p_emp_salary,
      p_out_id       =>:new_empid,
      p_out_total    =>:total
   );

   COMMIT;
   :status := 302;
   :location := '../simple_emp/61';
END;
```

Method Handler

Method: POST

Source Type: PL/SQL

MIME Types
application/json

Examples

URI Module: /rpc/

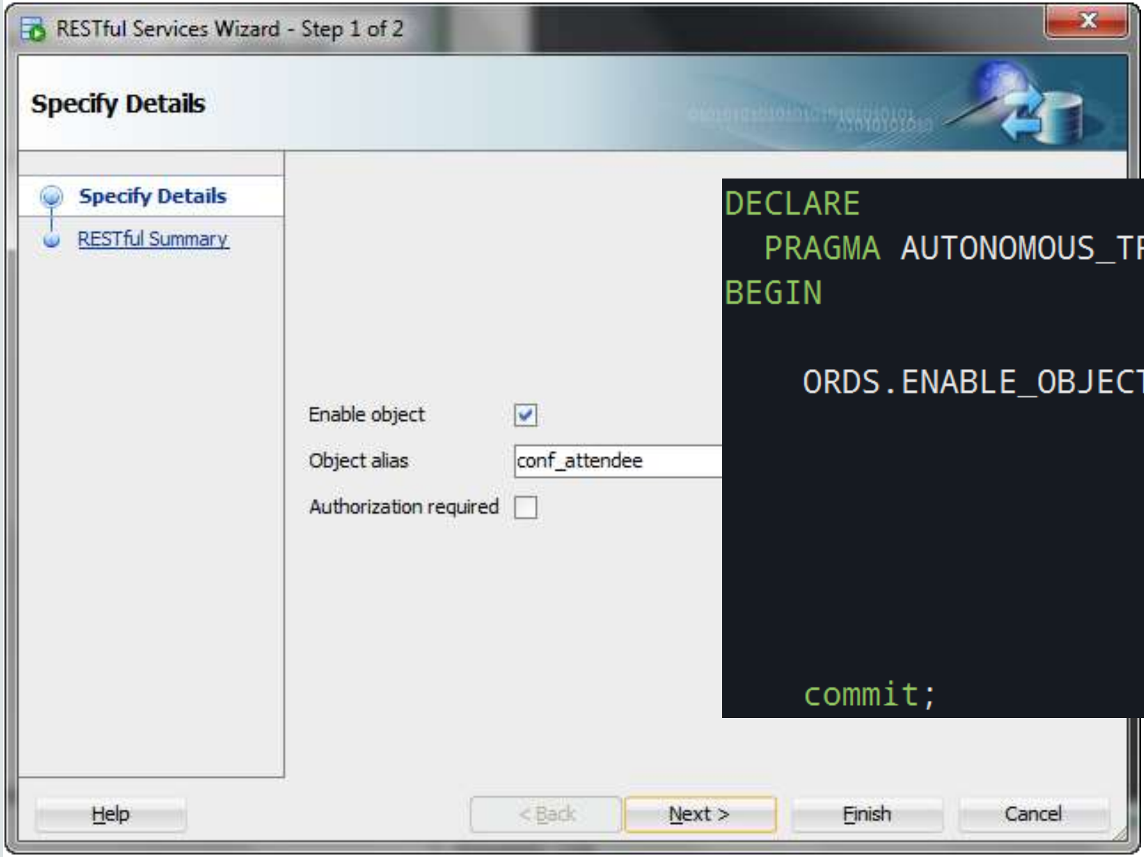
URI Pattern: AddEmp

http://myhost:8080/ords/myschema/rpc/AddEmp

Name	Bind Parameter	Access Method	Source Type	Data Type
LOCATION	location	OUT	HTTP HEADER	STRING
STATUS	status	OUT	HTTP HEADER	INTEGER

Automatic

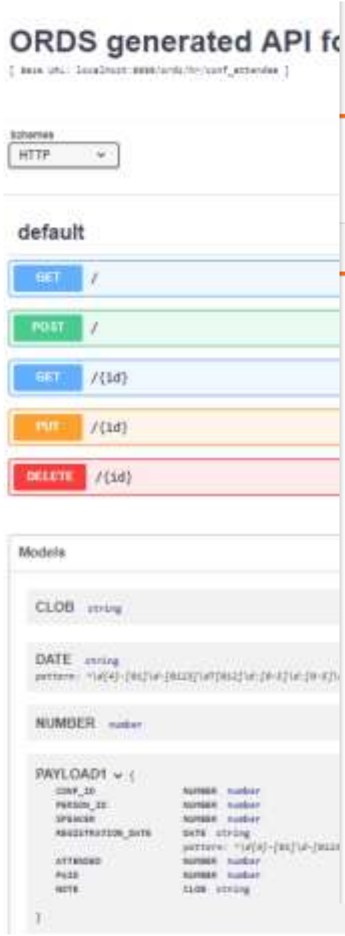
- Pick the Database objects to PUBLISH
- TABLEs and VIEWs
    - GET, POST, PUT, DELETE handlers avail for CRUD
- Stored Procedures, Functions, Packages (PL/SQL)
    - POST handler avail for RPC


RESTful Services Wizard - Step 1 of 2

Specify Details

Enable object ✓
Object alias: conf_attendee
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;
```


ORDS generated API fo

[Base URL: localhost:8080/ords/hr/conf_attendee]

Schemes
HTTP

default

GET	/
POST	/
GET	/{id}
PUT	/{id}
DELETE	/{id}

Models

CLOB string

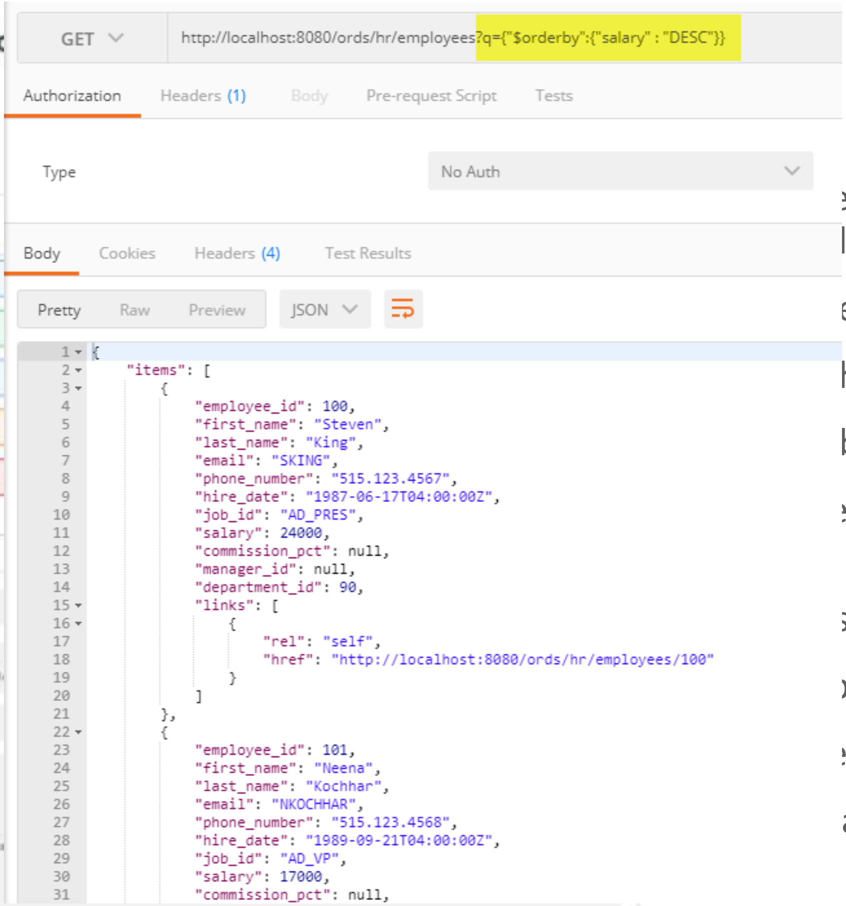
DATE string
pattern: ^\d{4}-[01]\d-[0123]\dT[012]\d:[0-5]\d:[0-5]\d

NUMBER number

PAYLOAD1 {

CONF_ID	NUMBER number
PERSON_ID	NUMBER number
SPEAKER	NUMBER number
REGISTRATION_DATE	DATE string
	pattern: ^\d{4}-[01]\d-[012
ATTENDED	NUMBER number
PAID	NUMBER number
NOTE	CLOB string

}


GET http://localhost:8080/ords/hr/employees?q={"$orderby":{"salary" : "DESC"}}

Authorization | Headers (1) | Body | Pre-request Script | Tests

Type: No Auth

Body | Cookies | Headers (4) | Test Results

Pretty | Raw | Preview | JSON

```
{
    "items": [
        {
            "employee_id": 100,
            "first_name": "Steven",
            "last_name": "King",
            "email": "SKING",
            "phone_number": "515.123.4567",
            "hire_date": "1987-06-17T04:00:00Z",
            "job_id": "AD_PRES",
            "salary": 24000,
            "commission_pct": null,
            "manager_id": null,
            "department_id": 90,
            "links": [
                {
                    "rel": "self",
                    "href": "http://localhost:8080/ords/hr/employees/100"
                }
            ]
        },
        {
            "employee_id": 101,
            "first_name": "Neena",
            "last_name": "Kochhar",
            "email": "NKOCHHAR",
            "phone_number": "515.123.4568",
            "hire_date": "1989-09-21T04:00:00Z",
            "job_id": "AD_VP",
            "salary": 17000,
            "commission_pct": null,
```

ɛnablement of RESTful request
les – User Interface

es these operations

hema level Metadata

ble Metadata

ɛt (Select)

Query (Filtering/Order/ASOF)

ɜert

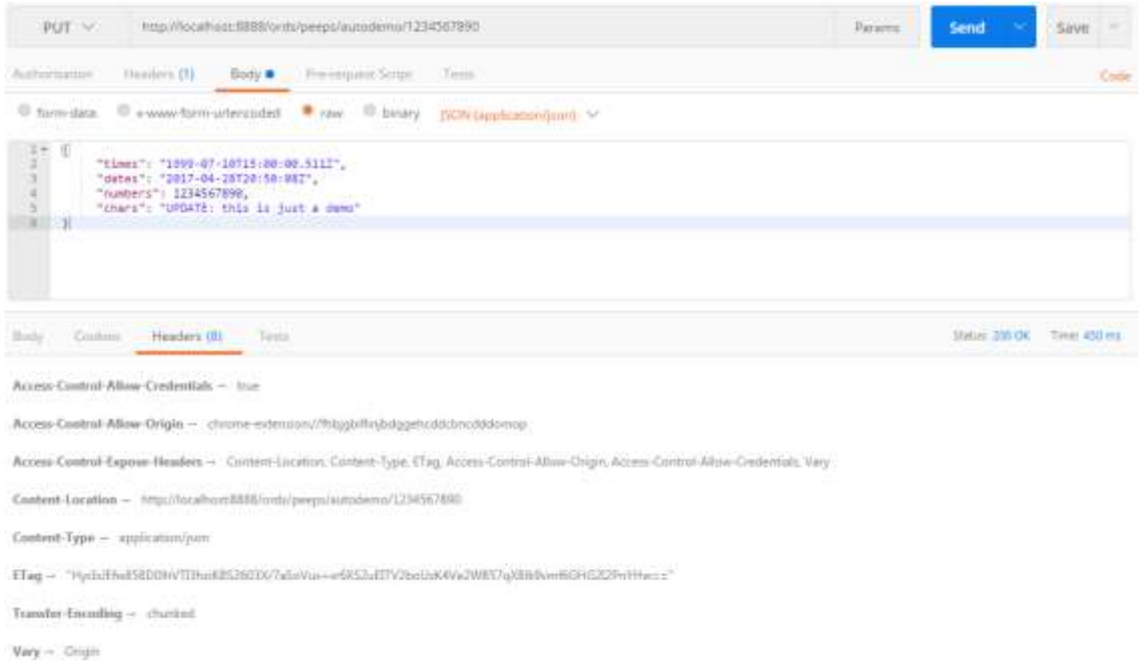
ɔdate

ɛlete

ad CSV

ORACLE

Update a Row



http://localhost:8888/ords/peeps/autodemo/1234567890

```
{
    "times": "1999-07-10T15:00:00.511Z",
    "dates": "2017-04-28T20:50:08Z",
    "numbers": 1234567890,
    "chars": "UPDATE: this is just a demo"
}
```

Status: 200 OK Time: 450 ms

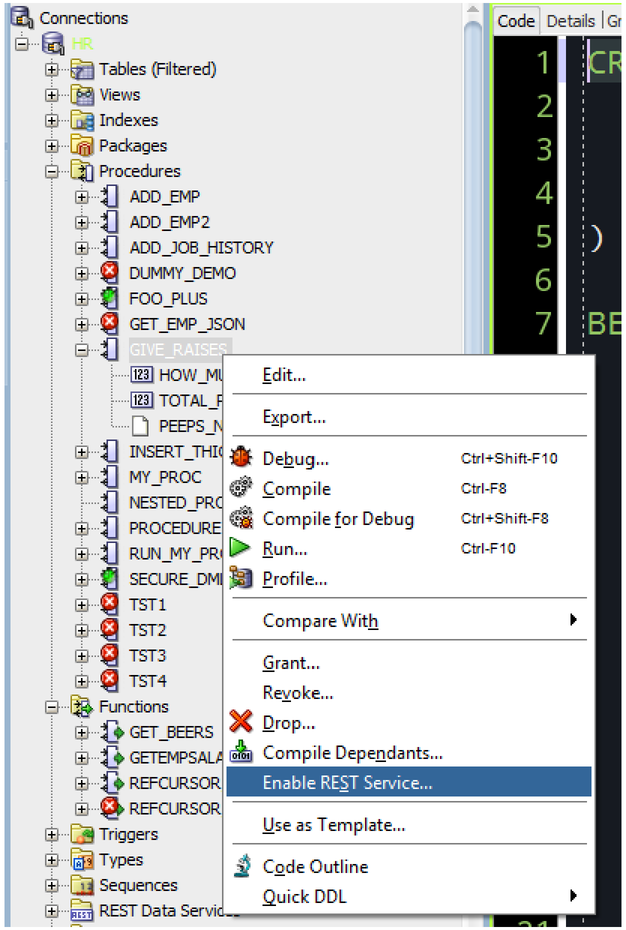
```
Access-Control-Allow-Credentials → true
Access-Control-Allow-Origin → chrome-extension://fhbjgbiflinjbdggehcddcbncdddomop
Access-Control-Expose-Headers → Content-Location, Content-Type, ETag, Access-Control-Allow-Origin, Access-Control-Allow-Credentials, Vary
Content-Location → http://localhost:8888/ords/peeps/autodemo/1234567890
Content-Type → application/json
ETag → "HycbJFfw858D0HVTI3huKBS2607X/7aSaVus+e6XS2uEITV2bdJsK4Va2W8Y7qX8BNmf6GHG2Z2PnYHacz="
Transfer-Encoding → chunked
Vary → Origin
```

METHOD : PUT /:PK

REQUEST BODY : JSON

RESPONSE: 200 OK

- Location (Header)
- JSON (Body)



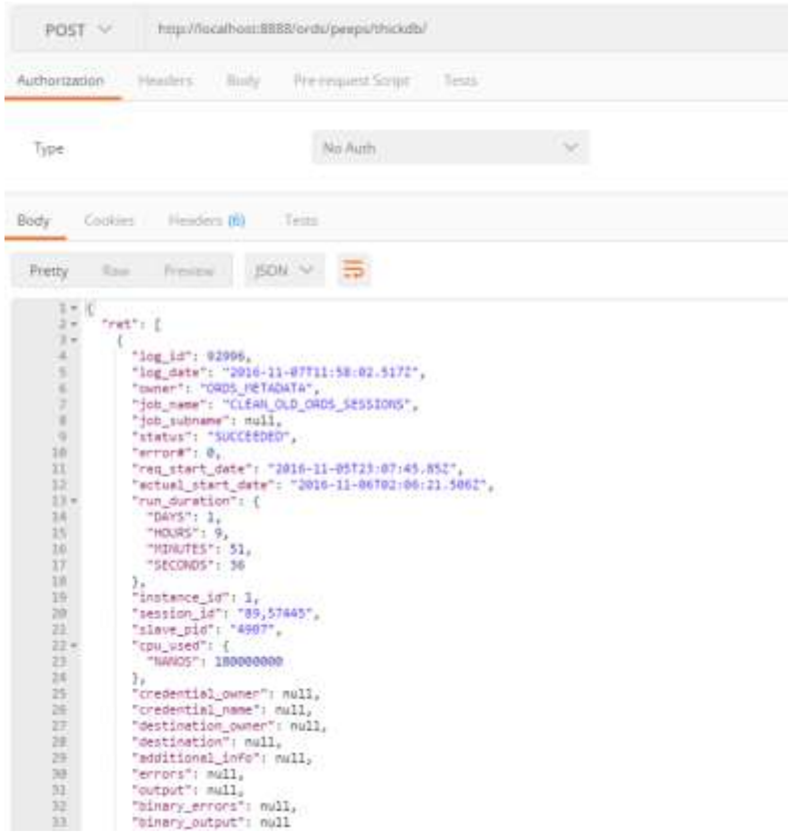
Remote Procedure Call over HTTP(S) via POST

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,
      …
    }
```

ORACLE



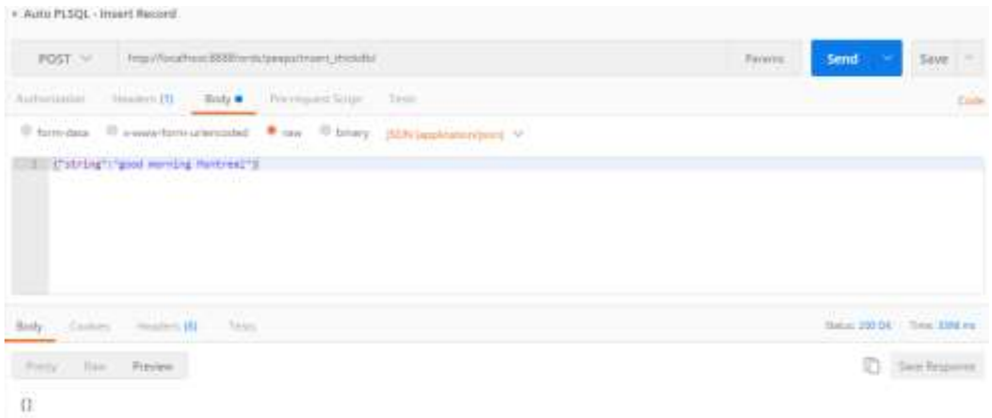
POST http://localhost:8888/ords/peeps/thickdb/

Authorization | Type: No Auth

Body | Pretty | JSON

```
{
  "ret": [
    {
      "log_id": 92996,
      "log_date": "2016-11-07T11:58:02.517Z",
      "owner": "ORDS_METADATA",
      "job_name": "CLEAN_OLD_ORDS_SESSIONS",
      "job_subname": null,
      "status": "SUCCEEDED",
      "error#": 0,
      "req_start_date": "2016-11-05T23:07:45.85Z",
      "actual_start_date": "2016-11-06T02:06:21.506Z",
      "run_duration": {
        "DAYS": 1,
        "HOURS": 9,
        "MINUTES": 51,
        "SECONDS": 36
      },
      "instance_id": 1,
      "session_id": "89,57445",
      "slave_pid": "4907",
      "cpu_used": {
        "NANOS": 180000000
      },
      "credential_owner": null,
      "credential_name": null,
      "destination_owner": null,
      "destination": null,
      "additional_info": null,
      "errors": null,
      "output": null,
      "binary_errors": null,
      "binary_output": null
```

Execute, REFCURSOR RETURN



Auto PLSQL - Insert Record

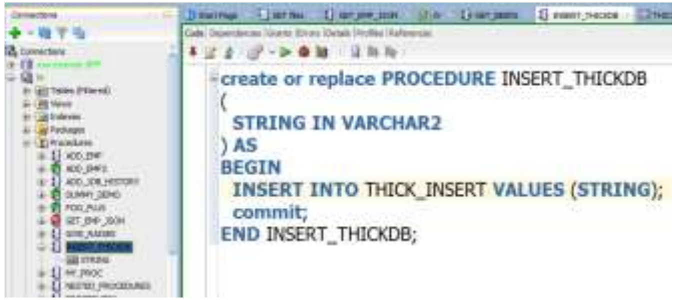
POST http://localhost:8888/ords/peeps/insert_thickdb/

Body | raw | JSON (application/json)

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

Body | Preview

```
{}
```



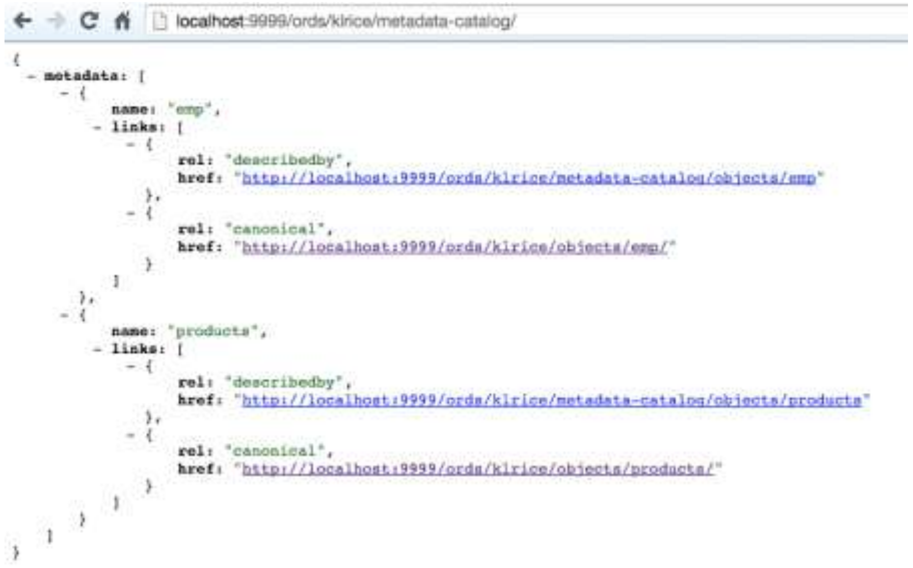
```
create or replace PROCEDURE INSERT_THICKDB
(
  STRING IN VARCHAR2
) AS
BEGIN
  INSERT INTO THICK_INSERT VALUES (STRING);
  commit;
END INSERT_THICKDB;
```

Execute PL/SQL TABLE API

/metadata-catalog/

Show me what's available for SCHEMA 'X'

- Describe & Inventory Services
- /ords/<schema>/metadata-catalog/

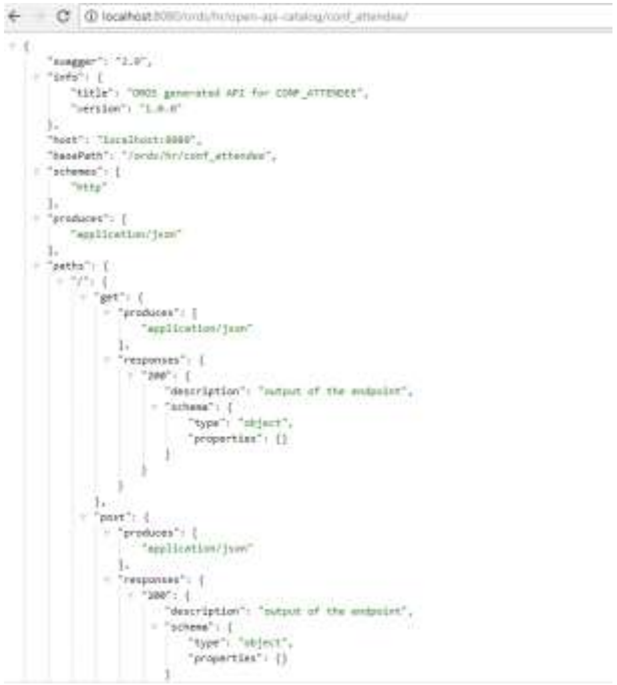


localhost:9999/ords/klrice/metadata-catalog/

```
{
 - metadata: [
    - {
          name: "emp",
        - links: [
            - {
                  rel: "describedby",
                  href: "http://localhost:9999/ords/klrice/metadata-catalog/objects/emp"
              },
            - {
                  rel: "canonical",
                  href: "http://localhost:9999/ords/klrice/objects/emp/"
              }
          ]
      },
    - {
          name: "products",
        - links: [
            - {
                  rel: "describedby",
                  href: "http://localhost:9999/ords/klrice/metadata-catalog/objects/products"
              },
            - {
                  rel: "canonical",
                  href: "http://localhost:9999/ords/klrice/objects/products/"
              }
          ]
      }
  ]
}
```

New for 17.4 - /open-api-catalog/

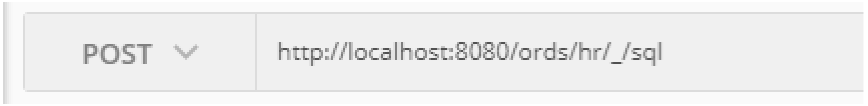
{Swagger}



localhost:8080/ords/hr/open-api-catalog/conf_attendee/

```
{
    "swagger": "2.0",
    "info": {
        "title": "ORDS generated API for CONF_ATTENDEE",
        "version": "1.0.0"
    },
    "host": "localhost:8080",
    "basePath": "/ords/hr/conf_attendee",
    "schemes": [
        "http"
    ],
    "produces": [
        "application/json"
    ],
    "paths": {
        "/": {
            "get": {
                "produces": [
                    "application/json"
                ],
                "responses": {
                    "200": {
                        "description": "output of the endpoint",
                        "schema": {
                            "type": "object",
                            "properties": {}
                        }
                    }
                }
            },
            "post": {
                "produces": [
                    "application/json"
                ],
                "responses": {
                    "200": {
                        "description": "output of the endpoint",
                        "schema": {
                            "type": "object",
                            "properties": {}
                        }
```

New for 17.4 - _/sql/

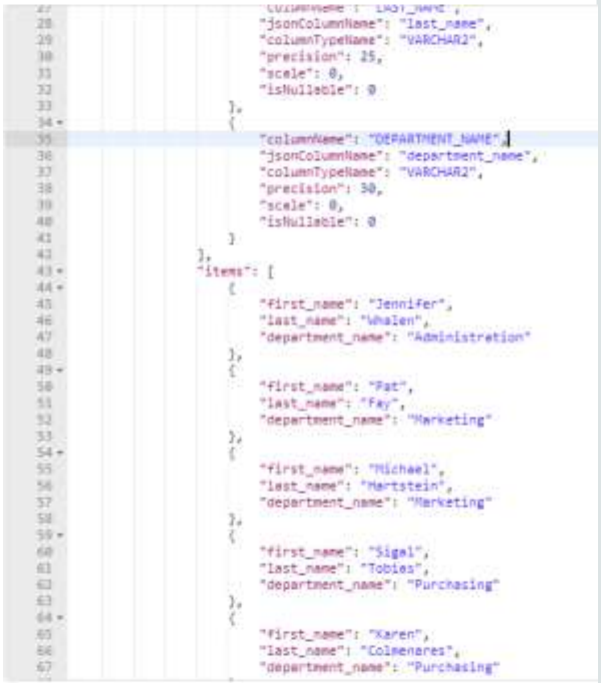


```
POST    http://localhost:8080/ords/hr/_/sql
```

Execute SQL via POST

- Disabled by default
- AUTH by user with sql dev priv or via DB

```
curl -X POST \
  http://localhost:8080/ords/hr/_/sql \
  -H 'authorization: Basic SFI6b3JhY2xl’ \
  -H 'cache-control: no-cache’ \
  -H 'content-type: application/sql’ \
  -H 'postman-token: 23a49622-a195-cb76-0606-358f3e371cdd’ \
  -d 'SELECT first_name, last_name, department_name
FROM hr.employees, hr.departments
where employees.department_id = departments.department_id'
```



```
                "columnName": "LAST_NAME",
                "jsonColumnName": "last_name",
                "columnTypeName": "VARCHAR2",
                "precision": 25,
                "scale": 0,
                "isNullable": 0
            },
            {
                "columnName": "DEPARTMENT_NAME",
                "jsonColumnName": "department_name",
                "columnTypeName": "VARCHAR2",
                "precision": 30,
                "scale": 0,
                "isNullable": 0
            }
        ],
        "items": [
            {
                "first_name": "Jennifer",
                "last_name": "Whalen",
                "department_name": "Administration"
            },
            {
                "first_name": "Pat",
                "last_name": "Fay",
                "department_name": "Marketing"
            },
            {
                "first_name": "Michael",
                "last_name": "Hartstein",
                "department_name": "Marketing"
            },
            {
                "first_name": "Sigal",
                "last_name": "Tobias",
                "department_name": "Purchasing"
            },
            {
                "first_name": "Karen",
                "last_name": "Colmenares",
                "department_name": "Purchasing"
```

Securing REST APIs

ORDS is Flexible - Security

Caveats

- Almost all dev/demo/blog is done with security off & with HTTP
- Always, always, always secure REST services and run with HTTPS

See Scott Spendolini's talk/slides this week on Security your REST APIs – lots of great info there

First Party Authentication

- Oracle REST Data Services specific solution
- Only available to the **author** of the API, application **must be deployed** on **same origin** as API
  - **https://example.com/api & https://example.com/app** ✓
  - **https://api.example.com & https://app.example.com** ✗
- User enters credentials in sign-in form, ORDS issues **cookie**, cookie is only validated by ORDS if the request is determined to originate from the from the **same origin** as the REST Service.

About OAuth 2.0

- IETF standard for securing access to REST APIs
- Comes in two forms:
  - **Two Legged** - For **Business to Business**, server to server applications
    - Example: Sync HR data between internal applications
  - **Three Legged** - For **Business to Consumer**, app to end-user applications
    - Example: Share subset of HR data with external benefits provider after employee approves access.
  - Third party **registers** client, issued **credentials**, uses credentials to acquire **access token**, uses **access token with request** to prove authorization

External Authentication

- Comes in many flavors, for example:
  - **Oracle Access Manager** - SSO cookie at Oracle OHS server level authenticates users stored in Oracle Identity Manager
- ORDS does not perform authentication, just authorization.
- Usually relies on HTTP cookies, need to restrict CORS allowed Origins to avoid CSRF

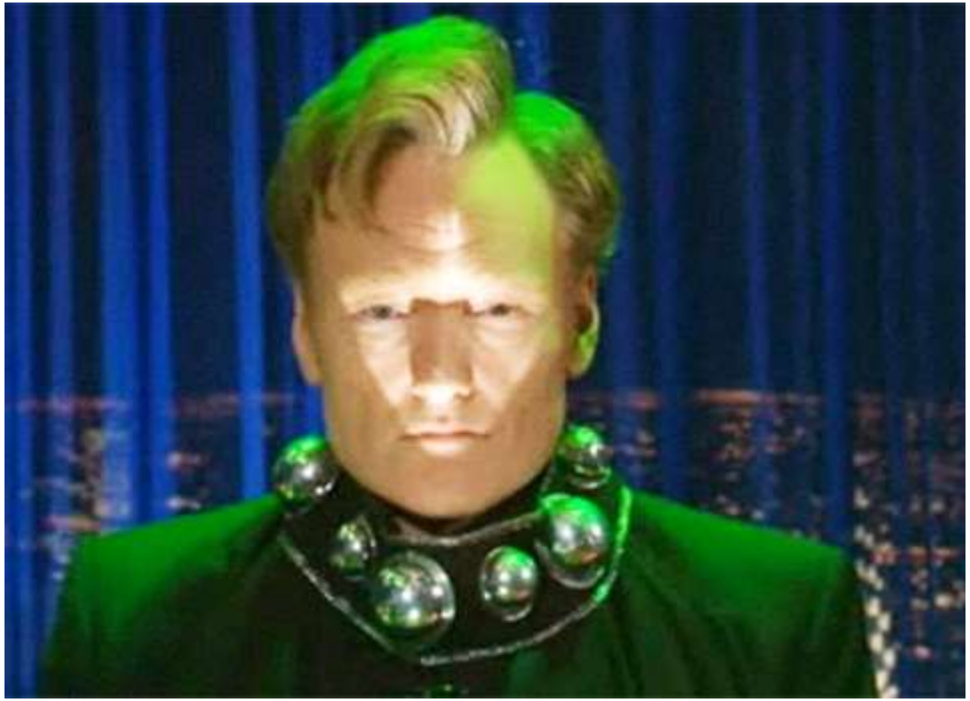
Coming Later This Year

REST API for managing your Oracle Database

- Available for On-Premises and Oracle Cloud DB Services
- Supports 11gR2 and higher
- Supports ‘Classic’ & Multitenant Architectures
- Supports RAC & Exadata
- Optional

In the year 2018…REST APIs for the Database!

COMING SOON



Listener – Start, stop, status

Database Ops

- Start, stop, alerts, INIT params, rotate TDE keys

PDB Ops

- Start, stop, create, clone, drop, plug, unplug

OS Stats

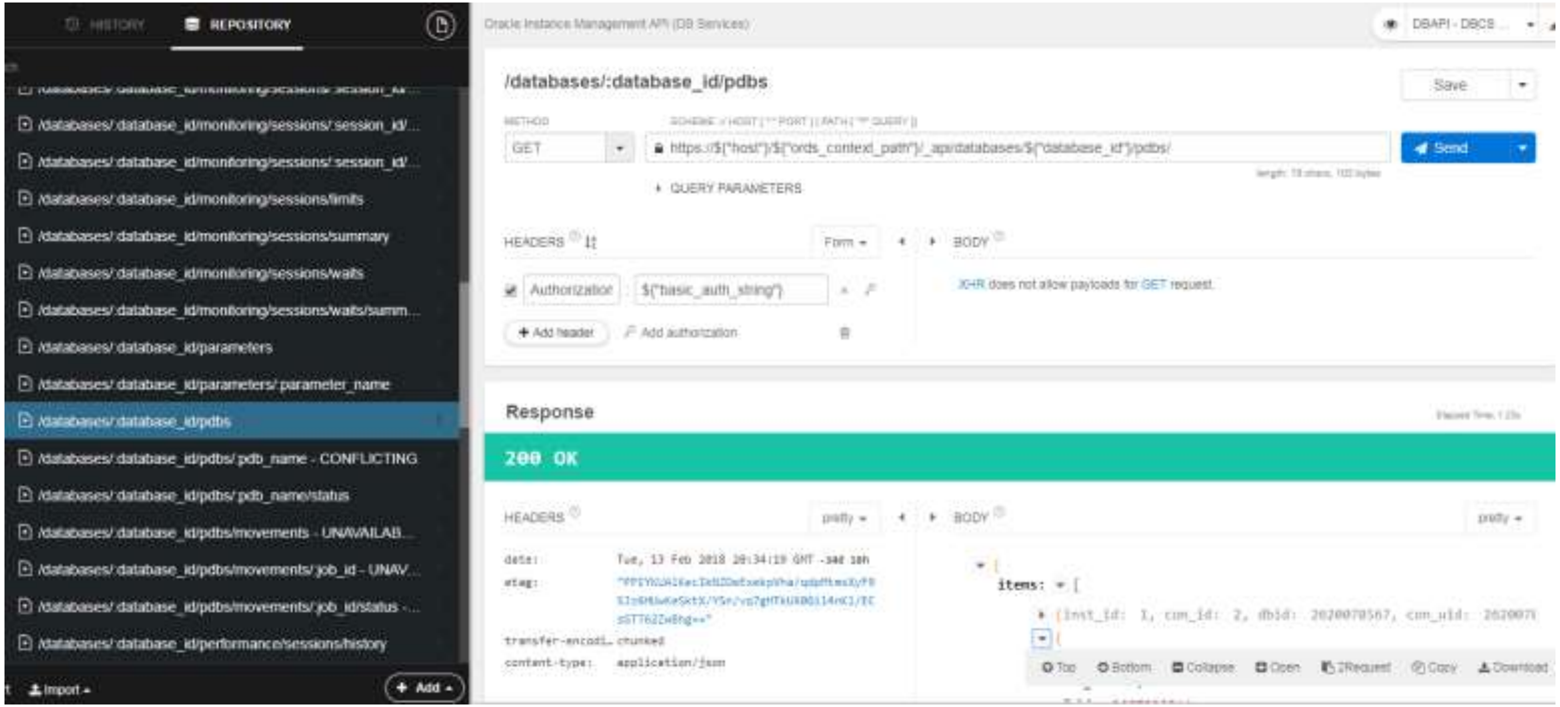
- Memory, cpu, processes

Reporting

- Backups, sessions, waits, ASH, AWR, locks, V$LONG_OPS, RTSM

Data Guard (all broker ops)

Clone a PDB, Get a list of Wait Events, Read the Alert Log…



/databases/:database_id/monitoring/sessions/:session_id/...
/databases/:database_id/monitoring/sessions/:session_id/...
/databases/:database_id/monitoring/sessions/limits
/databases/:database_id/monitoring/sessions/summary
/databases/:database_id/monitoring/sessions/waits
/databases/:database_id/monitoring/sessions/waits/summ...
/databases/:database_id/parameters
/databases/:database_id/parameters/:parameter_name
/databases/:database_id/pdbs
/databases/:database_id/pdbs/:pdb_name - CONFLICTING
/databases/:database_id/pdbs/:pdb_name/status
/databases/:database_id/pdbs/movements - UNAVAILAB...
/databases/:database_id/pdbs/movements/:job_id - UNAV...
/databases/:database_id/pdbs/movements/:job_id/status - ...
/databases/:database_id/performance/sessions/history

/databases/:database_id/pdbs

METHOD: GET

```
https://${"host"}/${"ords_context_path"}/_api/databases/${"database_id"}/pdbs/
```

✓ Authorization	${"basic_auth_string"}

XHR does not allow payloads for GET request.

Response

200 OK

```
date:              Tue, 13 Feb 2018 20:34:19 GMT -34d 10h
etag:              "[illegible]"
transfer-encodi…   chunked
content-type:      application/json
```

```
{
  items: [
    {inst_id: 1, con_id: 2, dbid: 2020078567, con_uid: 2020078
    {
```

Plus SQL Developer Web

Cloud First, On-Premises Later this Year

- Optional!
- 11gR2 and higher
- DBA screens
- SQL Worksheet
  - Run SQL, scripts, explain plan, autotrace, SQL history, formatter, insight
  - Create & Edit TABLE dialogs
- RE Schemas to a Relational Diagram/DD Reports

ORACLE

Thanks! Questions?

Resources

- Blogs
- Videos
- GitHub Examples
- Articles
  - UKOUG Scene Why REST, and What's in it or Me?
  - Oracle Mag AUTO REST & REST Enabled SQL