Oracle ORDS 101 - Jumpstart your Development

Dietmar Aust
Opal-Consulting, Köln
www.opal-consulting.de

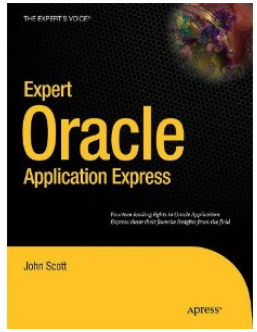
ORACLE ACE

OPAL Consulting

Dietmar Aust

- Dipl.-Inform. Dietmar Aust, Freelance Consultant
  - Master's Degree in Computer Science (MSCS)
- Building Oracle based Web Applications since 1997
  - Portal, Forms, Reports, OWA Toolkit, now APEX!
- 1997-2000: Consultant at Oracle Germany
- Since 09/2000: Freelance Consultant, Since 2006 – APEX only!
- Blog: http://daust.blogspot.com/
- Regular presenter at Oracle conferences (ODTUG, DOAG, OOW)
- Author of the JasperReportsIntegration toolkit
  - http://www.opal-consulting.de/tools

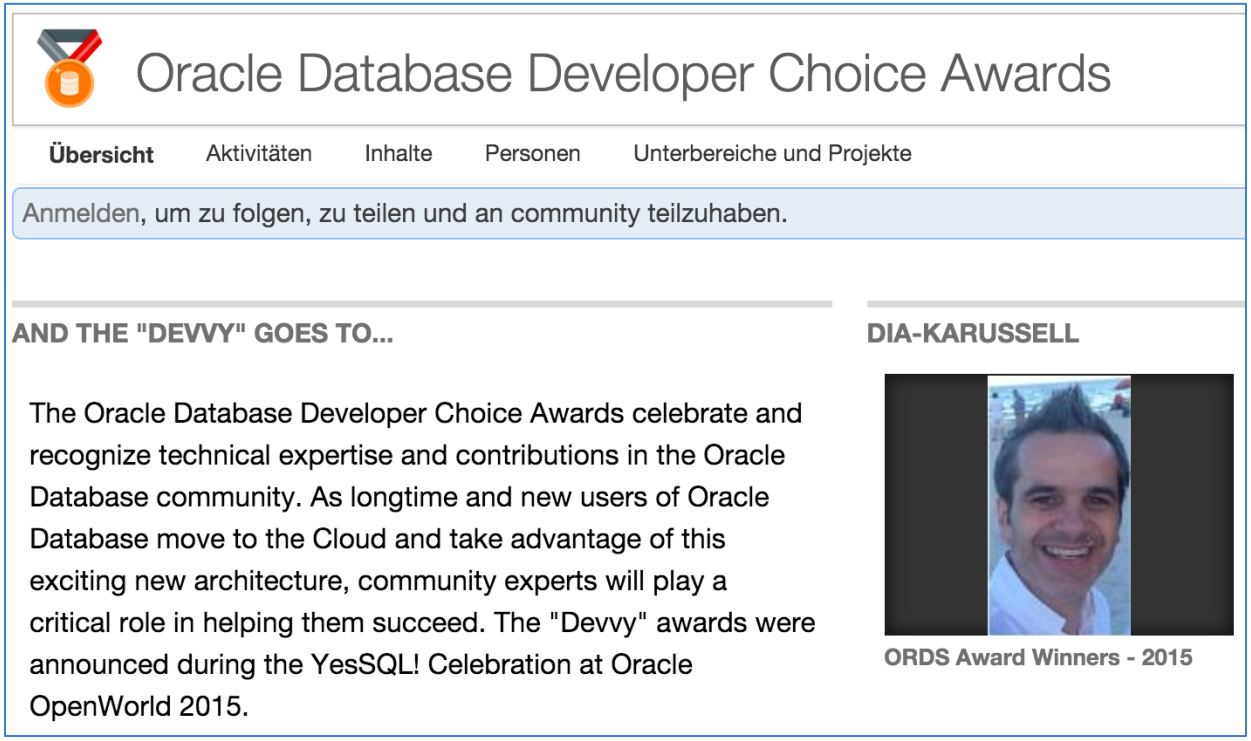
ORACLE ACE



OPAL Consulting

Dietmar Aust

- 2015 Database Developer of the year in the ORDS category
  - Primarily for my knowledge regarding ORDS as the platform technology for APEX
  - RESTful Services with ORDS are still relatively new – but they have great potential and will gain in importance



Agenda

Agenda

- What is REST?
- What is ORDS?
  - Components and Architecture
- Management of the REST definitions with SQL Developer and the API
- Use Cases
  - Navigation / Links / Filter / Sorting / Parameter (Input / Output)
- Security
  - Authentication and Authorization
- Additional Reading
- The slides and the demo script will be available from OGh.nl website and also on my blog: http://daust.blogspot.de

What is REST?

What is REST?
Definition

- It is an architectural style for applications, neither a protocol nor a W3C standard

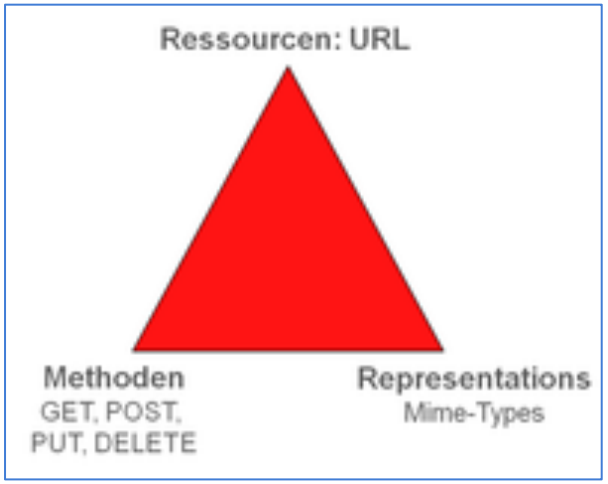
- REST := **Re**presentational **S**tate **T**ransfer term coined in 2000 by Roy Fielding
  - https://en.wikipedia.org/wiki/Representational_state_transfer

- Characteristics:
  - Stateless (100% of the application state is managed by the client)
  - Based on the http protocol
  - Highly scaleable
  - REST uses http methods (POST, PUT, GET, DELETE, …) to implement CRUD operations (Create / Read / Update / Delete)

- Why?
  - Lightweight alternative to RPC (Remote Procedure Calls) and other Web Services (SOAP, WSDL, …)
  - Increasingly popular through APIs provided by Google, Facebook, Twitter and others.

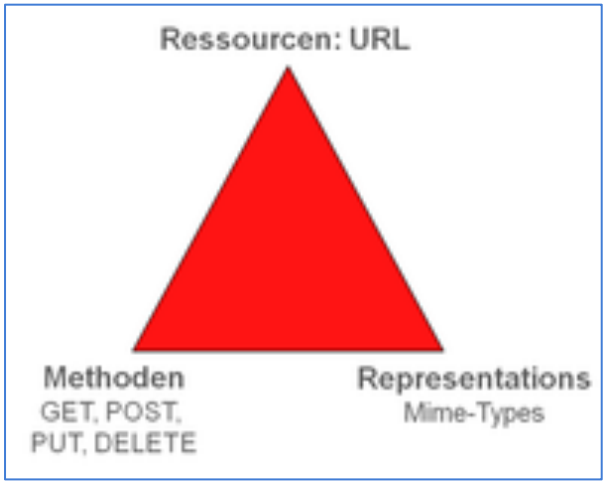
What is REST?
Ressources

- Ressources provide services and are uniquely identifyable
  - http://api.example.com/customers/
  - http://api.example.com/customers/1234
  - http://api.example.com/customers/1234/orders/
- Multiple URIs can point to the same ressource:
  - http://example.org/NewOrleans/traffic/I10
  - http://example.org/traffic/NewOrleans/I10
- We model the ressource, not the action!
  - Use of nouns in plural form
  - PUT http://example.com/accounts/12345
  - ~~PUT http://example.com/accounts/edit/12345~~
  - POST http://example.com/accounts/
  - ~~POST http://example.com/accounts/addaccount~~



What is REST?
Methods

- Methods implement a specific operation
  - Uniform operations for all ressources
  - GET, POST, PUT, DELETE, OPTIONS, HEAD
- We use very few verbs to operate on many different nouns.



RESTful API HTTP methods

Resource	GET	PUT	POST	DELETE
Collection URI, such as `http://api.example.com/resources/`	**List** the URIs and perhaps other details of the collection's members.	**Replace** the entire collection with another collection.	**Create** a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.[11]	**Delete** the entire collection.
Element URI, such as `http://api.example.com/resources/item17`	**Retrieve** a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	**Replace** the addressed member of the collection, or if it does not exist, **create** it.	Not generally used. Treat the addressed member as a collection in its own right and **create** a new entry in it.[11]	**Delete** the addressed member of the collection.

What is REST? Methods

- Communication of success and error messages through standard HTTP Response codes 1xx, 2xx, 3xx, 4xx, 5xx
  - http://www.restapitutorial.com/httpstatuscodes.html#

HTTP Status Codes

This page is created from HTTP status code information found at ietf.org and Wikipedia. Click on the **category heading** or the **status**

1xx Informational

100 Continue
101 Switching Protocols
102 Processing (WebD

2xx Success

★ 200 OK
★ 201 Created
202 Accepted
203 Non-Authoritative Information
★ 204 No Content
205 Reset Content
206 Partial Content
207 Multi-Status (WebDAV)
208 Already Reported (
226 IM Used

3xx Redirection

300 Multiple Choices
301 Moved Permanently
302 Found
303 See Other
★ 304 Not Modified
305 Use Proxy
306 (Unused)
307 Temporary Redirect
308 Permanent Redirec

4xx Client Error

★ 400 Bad Request
★ 401 Unauthorized
402 Payment Required
★ 403 Forbidden
★ 404 Not Found
405 Method Not Allowe
406 Not Acceptable
407 Proxy Authentication Required
408 Request Timeout
★ 409 Conflict
410 Gone
411 Length Required
412 Precondition Failed
413 Request Entity Too Large
414 Request-URI Too L
415 Unsupported Media Type
416 Requested Range Not Satisfiable
417 Expectation Failed
418 I'm a teapot (RFC 2324)
420 Enhance Your Calm (Twitter)
422 Unprocessable Ent
423 Locked (WebDAV)
424 Failed Dependency (WebDAV)
425 Reserved for WebD
426 Upgrade Required
428 Precondition Required
429 Too Many Request
431 Request Header Fields Too Large
444 No Response (Nginx)
449 Retry With (Microso
450 Blocked by Windows Parental Controls (Microsoft)
499 Client Closed Request (Nginx)

5xx Server Error

★ 500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
505 HTTP Version Not
506 Variant Also Negotiates (Experimental)
507 Insufficient Storage (WebDAV)
508 Loop Detected (We
509 Bandwidth Limit Exceeded (Apache)
510 Not Extended
511 Network Authentication Required
598 Network read timeout error
599 Network connect timeout error

★ 401 Unauthorized

The request requires user authentication. The response MUST include a WWW-Authenticate header field (section 14.47) containing a challenge applicable to the requested resource. The client MAY repeat the request with a suitable Authorization header field (section 14.8). If the request already included Authorization credentials, then the 401 response indicates that authorization has been refused for those credentials. If the 401 response contains the same challenge as the prior response, and the user agent has already attempted authentication at least once, then the user SHOULD be presented the entity that was given in the response, since that entity might include relevant diagnostic information. HTTP access authentication is explained in "HTTP Authentication: Basic and Digest Access Authentication".

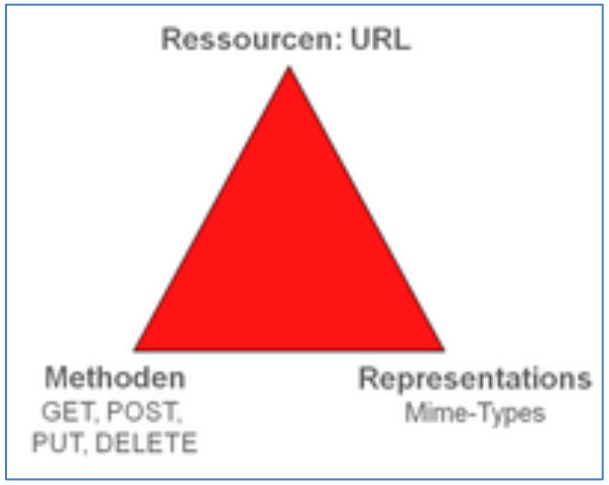
Wikipedia

Similar to 403 Forbidden, but specifically for use when authentication is possible but has failed or not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource. See Basic access authentication and Digest access authentication.

★ Error code response for missing or invalid authentication token.

What is REST?
Representations

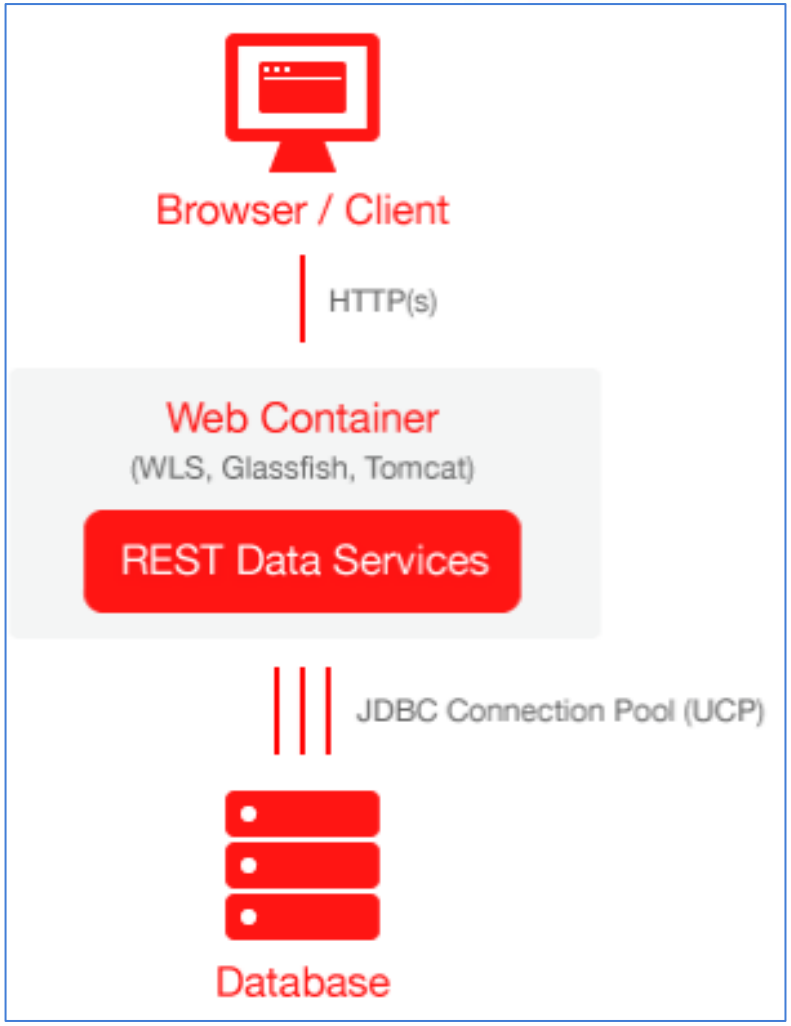
- Representations determine how the answer will be interpreted
  - XML representation using mime-type: text/xml
  - JSON representation using mime-type: application/json
- A single ressource can provide multiple different representations
  - JSON, XML, CSV …
  - The right representation is actively „negotiated“
  - The client sends a list of preferred mime-types – the server responds with the best answer and sends the chosen mime-type in the „Content-Type“ http header.



What is ORDS?

What is ORDS?

- Oracle Rest Data Services (ORDS)
  - Middleware J2EE component in the application server (WLS, Glassfish, Tomcat)
  - Translates URLs into a call in the database (either select or stored procedure call)
- Three major use cases
  - Support for OWA toolkit applications (will replace mod_plsql)
  - Oracle Application Express (APEX)
  - RESTful Webservices



What is ORDS?
The History

Version	Date	Description
1.0	2010	First release as Oracle APEX Listener with with support for OWA toolkit used by APEX
1.1	2011	First release with REST support for JSON, Microdata, CSV, Pagination. Also added FOP
2.0	2012	OAuth2 support, Integrated with APEX, Multi Database, SQL Developer integration
2.0.5	2013	Added support for Oracle Pluggable Databases (12c)
2.0.6	2014	Renamed to Oracle REST Data Services to emphasize REST commitment, integration with APEX 4.2 in SQL Workshop
2.0.8	2014	Added REST Filtering
3.0.0	2015	REST AutoTable, NoSQL, DB12 JSON, Bulk loading over REST,…

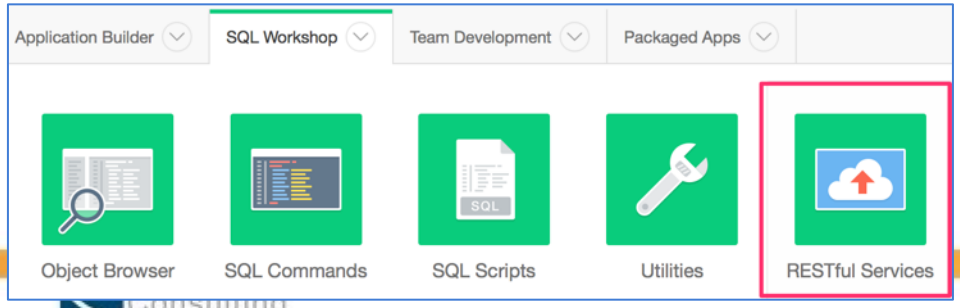
What is ORDS?
APEX REST vs. ORDS_METADATA REST Support

ORDS is currently transitioning away from the dependency on APEX

- ORDS requires a repository to store the webservice definitions

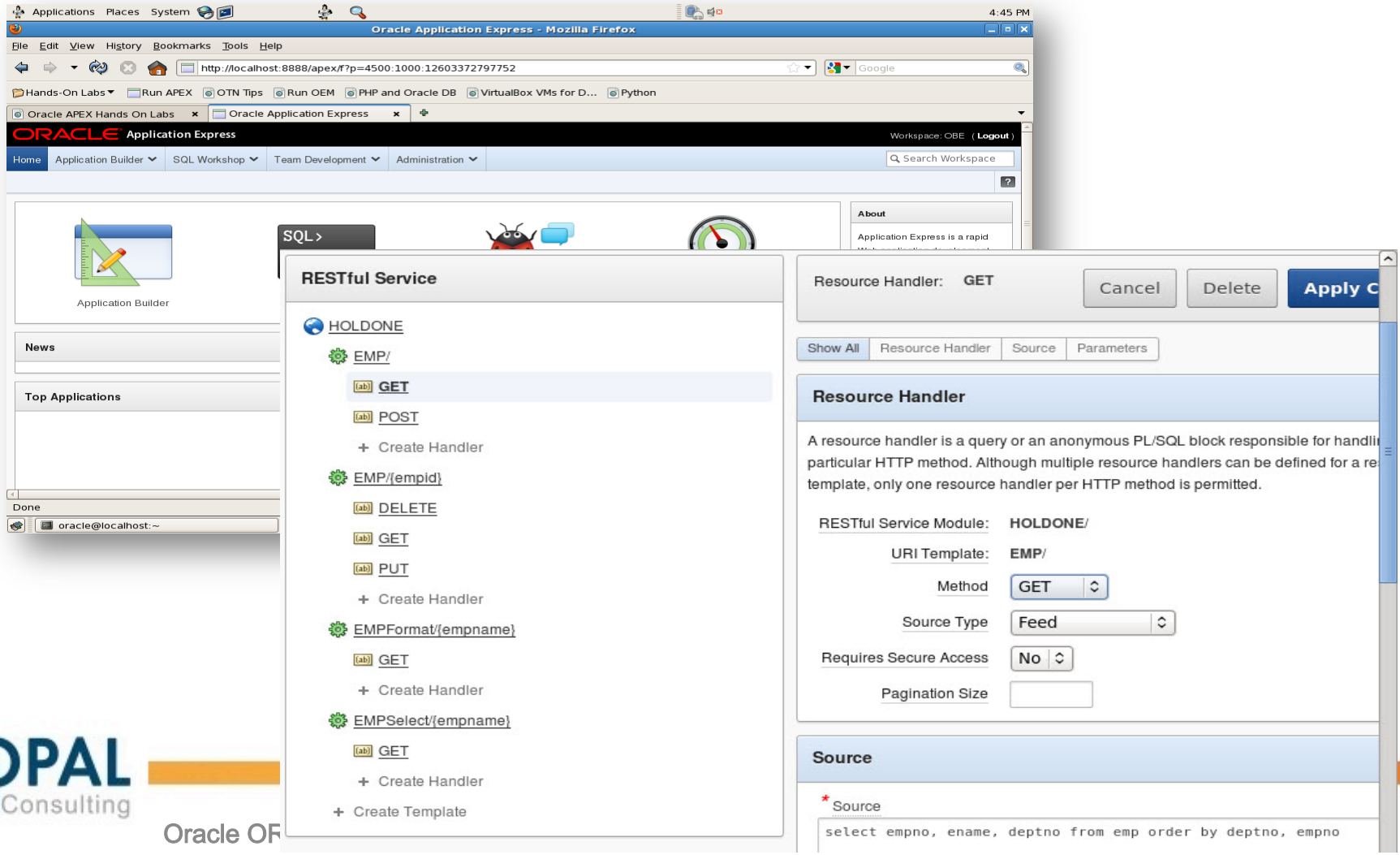
- ORDS 2.0
- Schemas
  - APEX_040200/APEX_050000
  - APEX_LISTENER
  - APEX_REST_PUBLIC_USER
- Configuration using
  - APEX SQL Workshop

- ORDS 3.0
- Schemas
  - ORDS_METADATA
  - ORDS_PUBLIC_USER
- Configuration using
  - SQL Developer
  - PL/SQL API



What is ORDS?
APEX REST vs. ORDS_METADATA REST Support

- APEX REST support in the APEX SQL Workshop



What is ORDS?
APEX REST vs. ORDS_METADATA REST Support

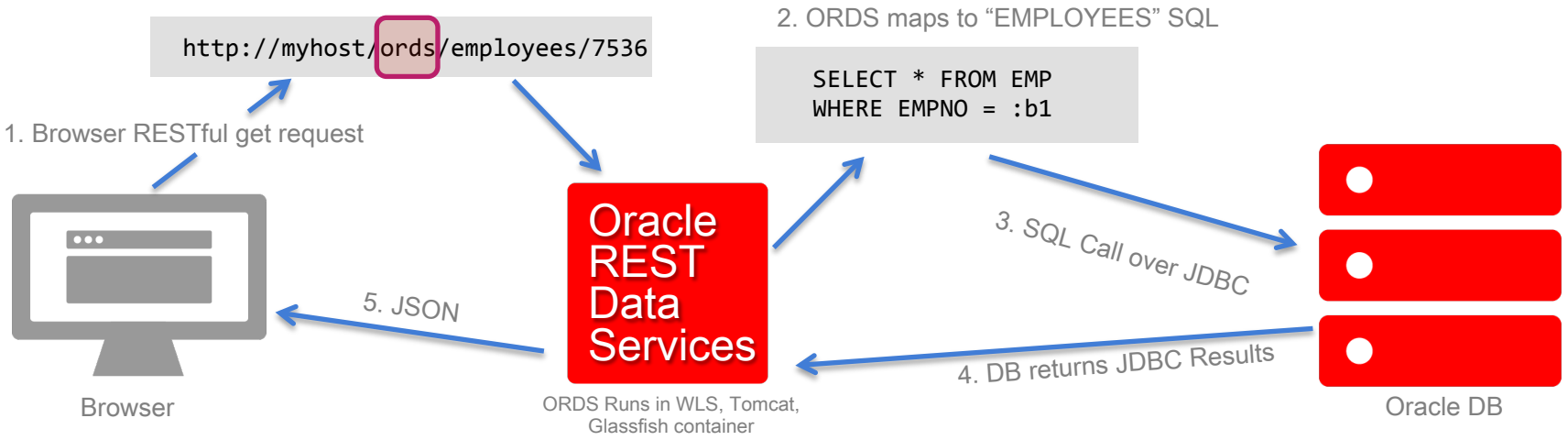
- Two different repositories : APEX REST and ORDS_METADATA REST
  - Typically both are installed when using APEX 5
  - APEX 5 requires that you run apex_rest_config.sql which creates APEX_LISTENER and APEX_REST_PUBLIC_USER
- In which repository do I create the webservice?
- APEX REST
  - Integration with APEX Session
- ORDS_METADATA REST
  - The new REST functionality based on the new metadata repository
  - PL/SQL APIs (define and oauth)
- The Future?
  - New features will only be added to ORDS_METADATA REST

What is ORDS?
APEX REST vs. ORDS_METADATA REST Support

- Currently the documentation doesn't always clearly distinguish between APEX REST and ORDS_METADATA REST.
- Thus the samples in the documentation and on the internet are mixed …. but there are small functional differences!
- In this presentation we will focus on the features available in **ORDS_METADATA REST Support with ORDS 3.0**

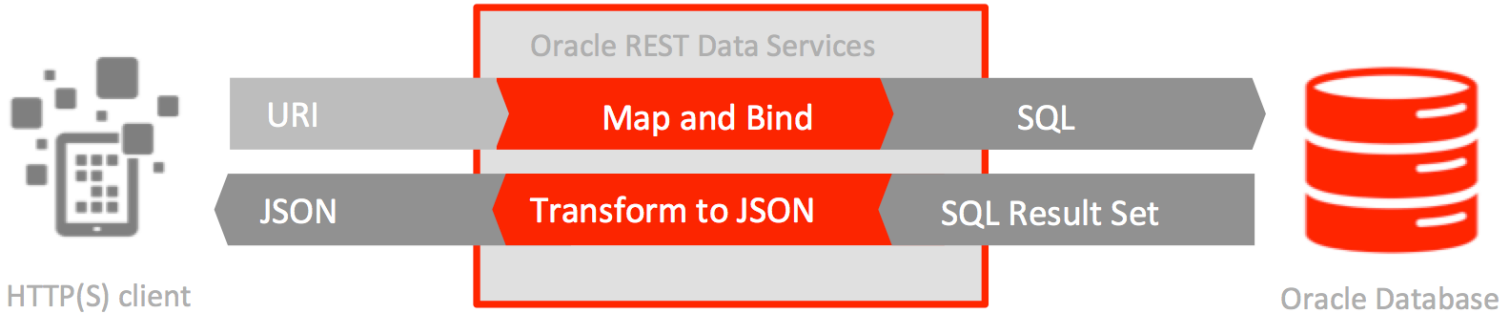
What is ORDS? Architecture

- How is a REST webservice call actually processed?



What is ORDS?
Architecture

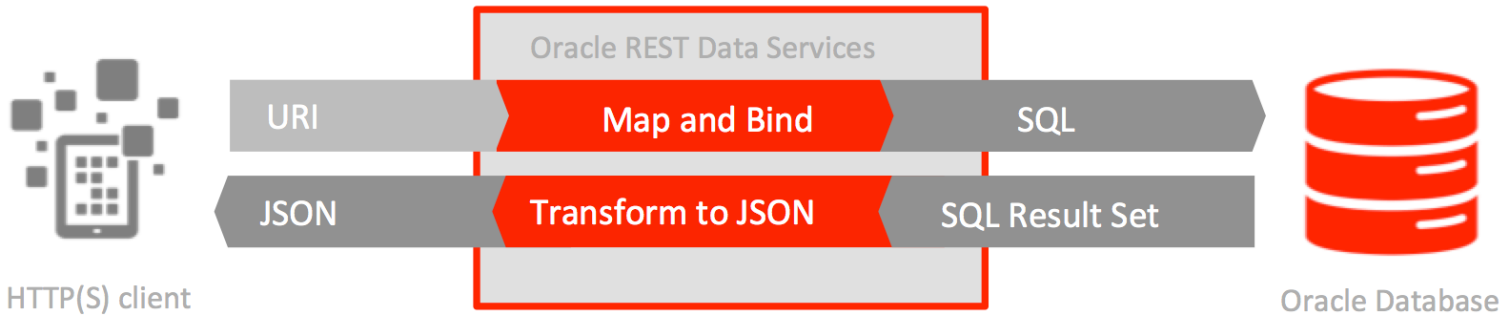
- Map and Bind:



  - Implicitly access all URI parameters in the URL or in the body (e.g. POST request)
    - Happens automatically , even JSON Parameters (using Content-Type: application/json)
  - Explicit parameters possible
  - Access header variables

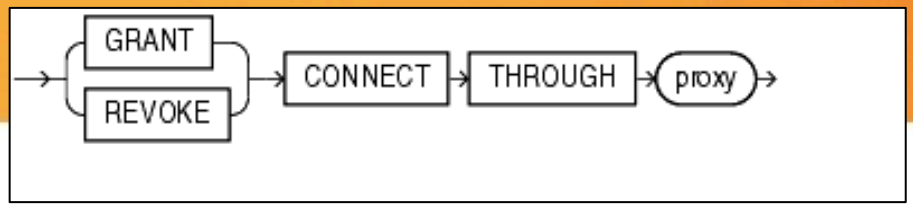
What is ORDS? Architecture

▶ Transform to JSON

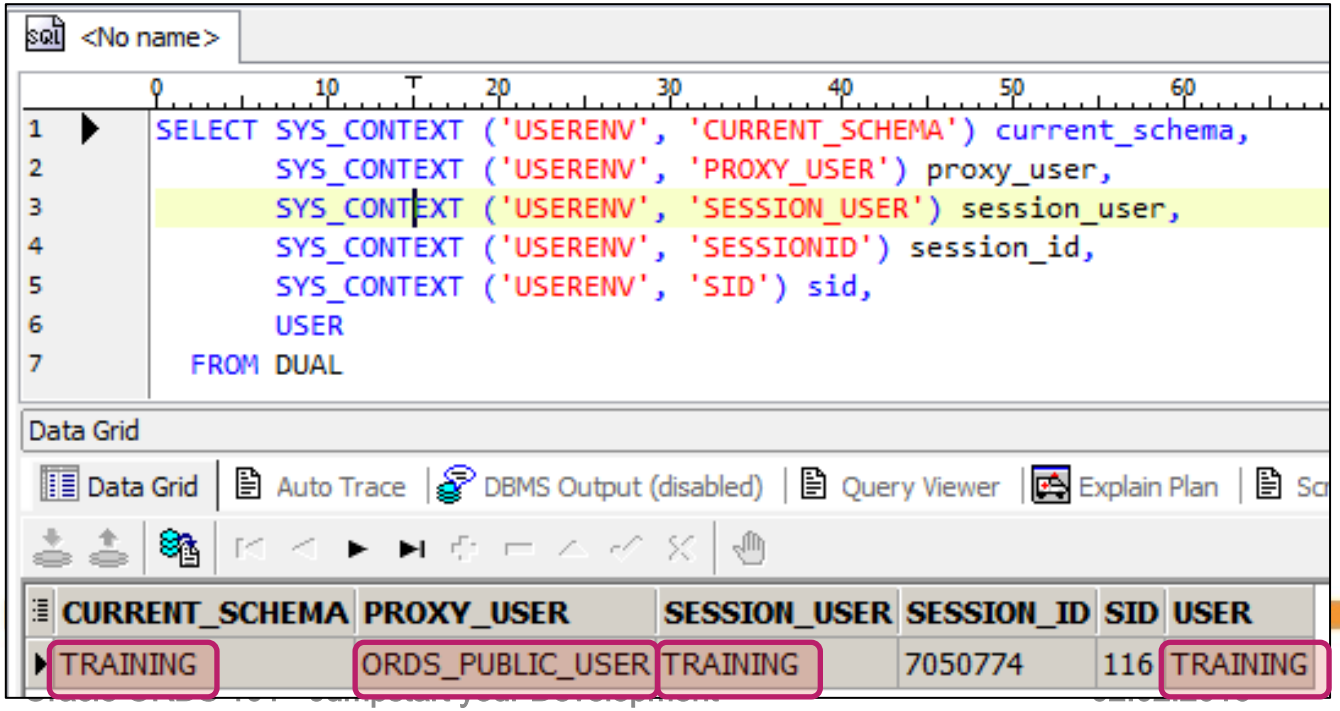


- Return JSON by using bind variables (declaratively) or create the JSON manually yourself
- Declarative Formats: JSON or CSV, manually you can create anything
- Can change the http return code or set http header variables

What is ORDS? Architecture



- Connection Pooling
  - The target Oracle user (schema) is activated using a Proxy Connect
  - The user ORDS_PUBLIC_USER connect to the database and then switches its identity to the target Oracle user
  - Thus we need fewer connection pools and each connection pool becomes smaller since multiple Oracle users can be served with the same connection pool
  - Each SQL and PL/SQL statement is executed using the the original user session



What is ORDS?
Architecture

- Connection Pooling
  - Easier to deal with than with APEX, since with APEX the session user will be APEX_PUBLIC_USER, NOT the parsing schema

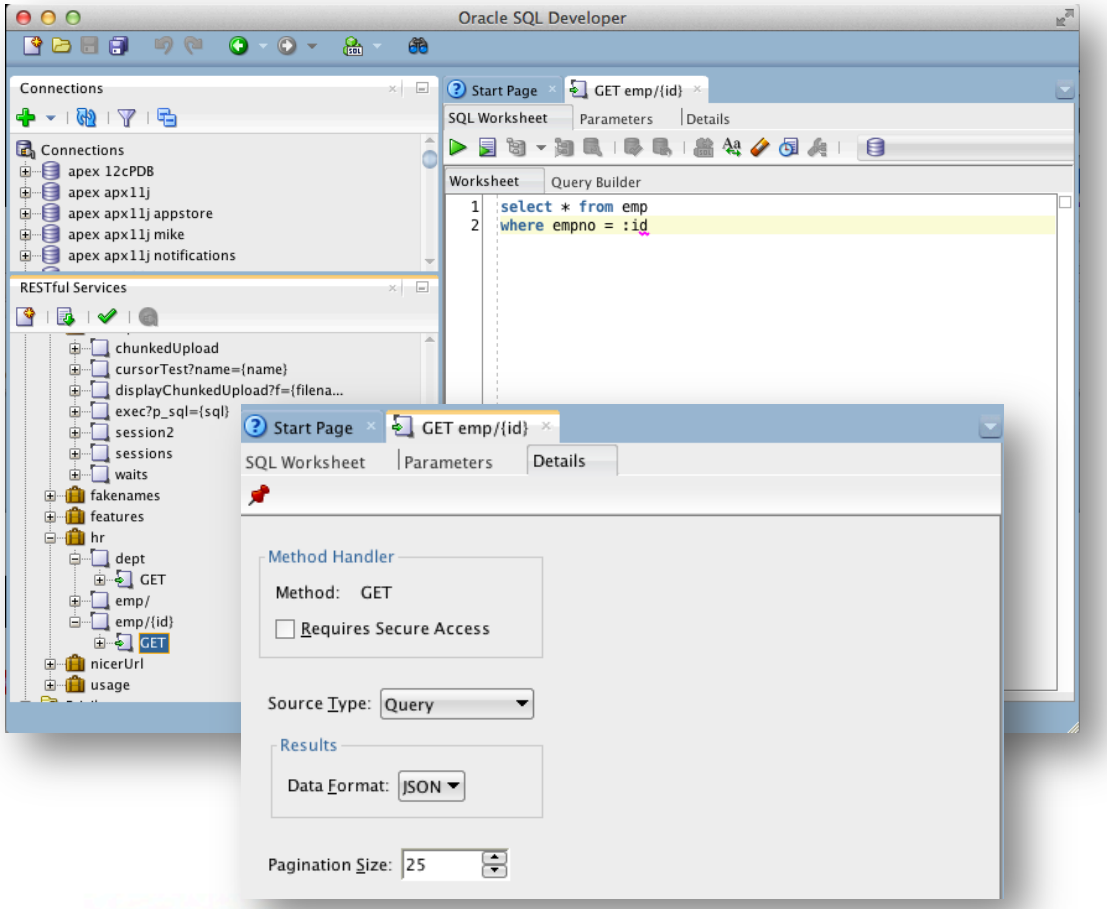
```
SELECT SYS_CONTEXT ('userenv', 'session_user') session_user,
       SYS_CONTEXT ('userenv', 'current_user') parsing_schema,
       v ('APP_USER') application_user
  FROM DUAL
```

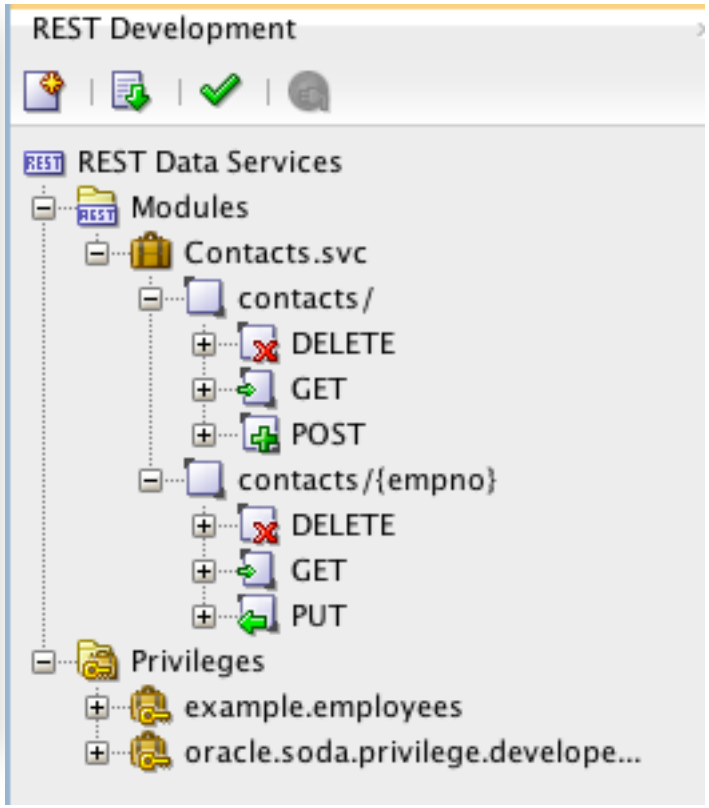
SESSION_USER	PARSING_SCHEMA	APPLICATION_USER
APEX_PUBLIC_USER	TRAINING	DIETMAR.AUST

Management of the REST definitions with SQL Developer and the API

REST Definitions
Management with SQL Developer

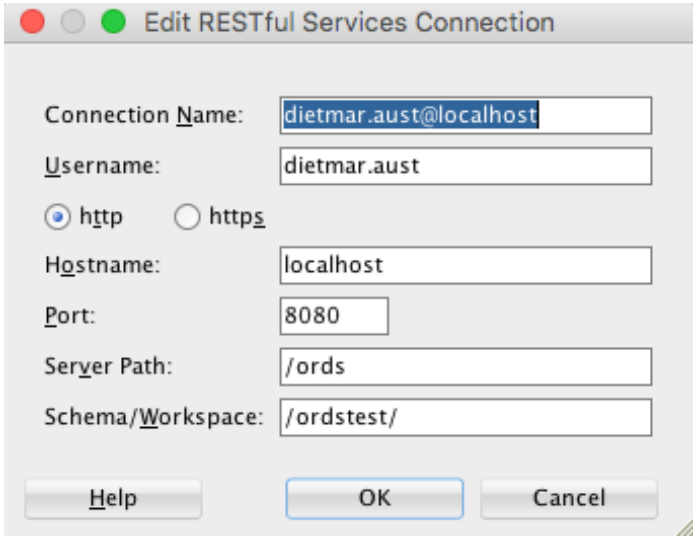
- Management of the REST definitions with SQL Developer





REST Definitions
Management with SQL Developer

- Management of the REST definitions with SQL Developer



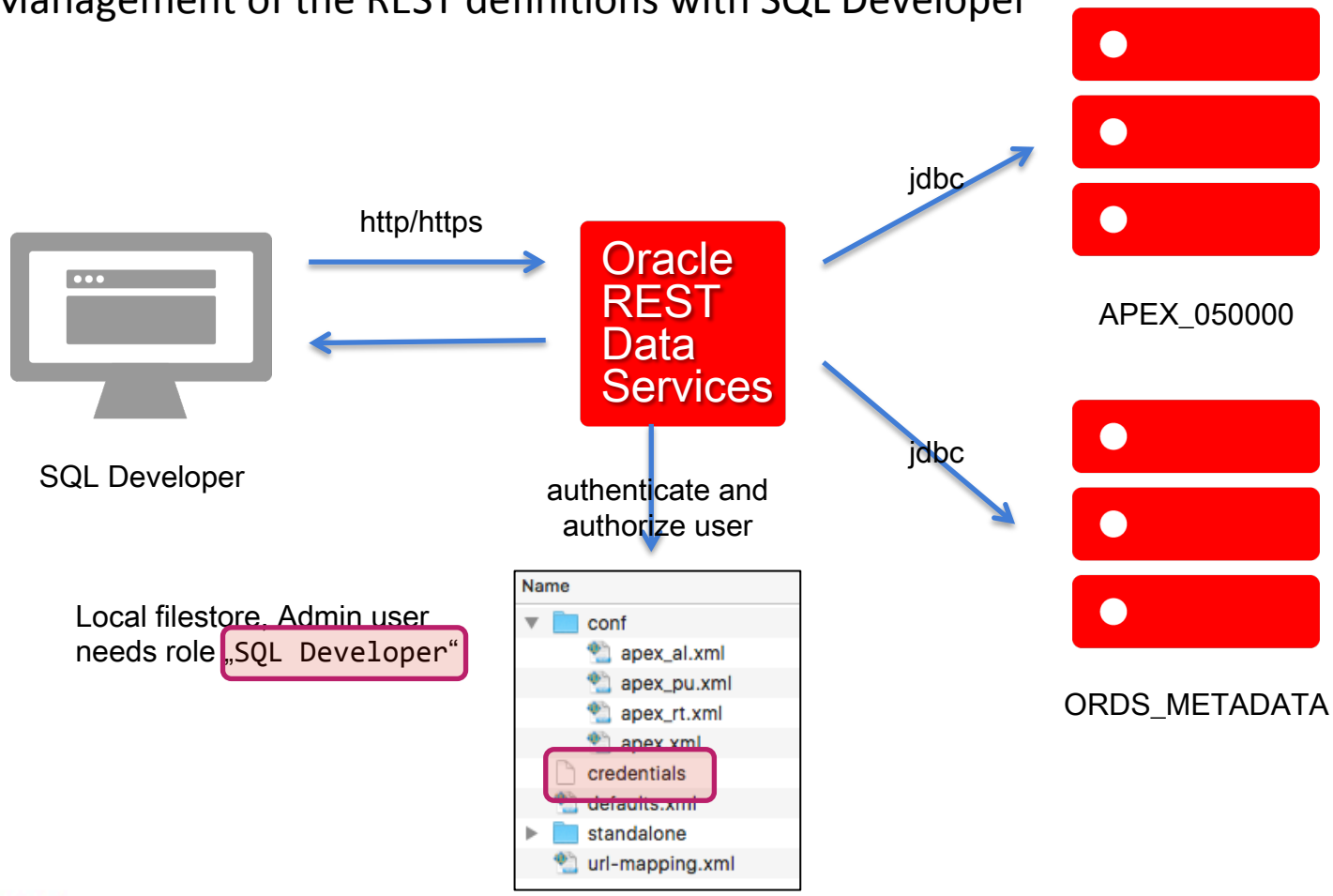
Schema/Workspace will decide between APEX REST and ORDS REST

- Create user on command line

```
## User to manage REST definitions in SQL Developer
java -jar ords.war user dietmar.aust "SQL Developer”
```

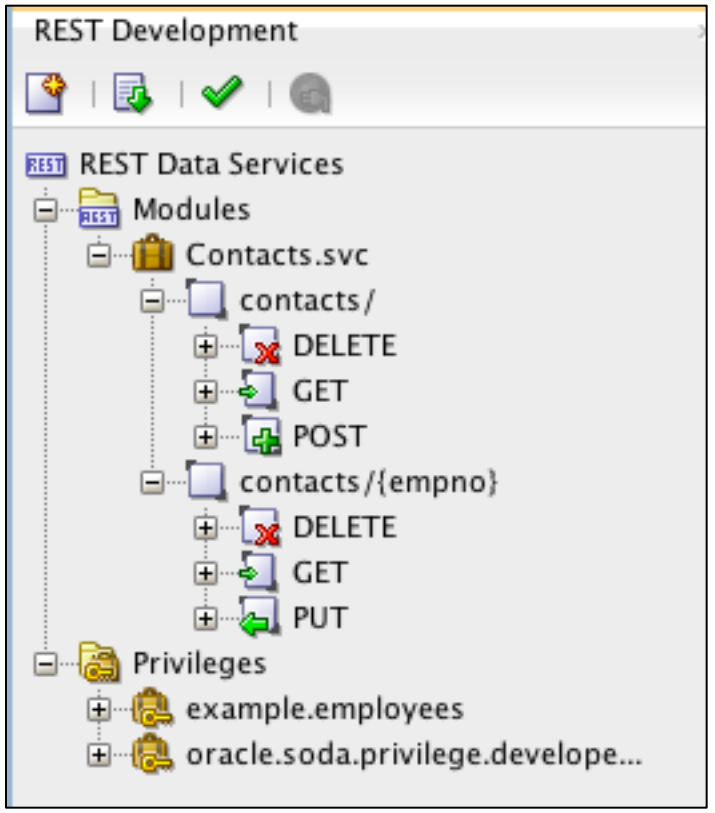
REST Definitions
Management with SQL Developer

- Management of the REST definitions with SQL Developer



REST Definitions
Modules, Ressource Templates and Handlers

- REST components
  - Modules
    - Resource Templates
      - Methods / Handlers (GET, PUT, POST, DELETE)



Demo

REST Definitions
Management through the API

- Management through the PL/SQL API
- Simple file … contains all resource templates and methods for a module in a single place
- First we delete the existing definition and then we recreate it from scratch
- Very well suited for script based deployment
- API reference (Package ORDS): http://docs.oracle.com/cd/E56351_01/doc.30/e56293/ords_ref.htm#AELIG90180
- **Use Package ORDS instead of ORDS_SERVICES in the future!!!**

```
set sqlblanklines on

declare
  l_moduleid   number;
  l_templateid number;
  l_handlerid  number;
  l_module_name user_ords_modules.name%type := 'demo';
  l_pattern user_ords_templates.uri_template%type;
begin
  -- delete module if exists
  ords.delete_module( p_module_name => l_module_name);

  -- create module
  ords.define_module(
    p_module_name       => l_module_name
   ,p_base_path => 'demo/'
   ,p_status     => 'PUBLISHED'
  );

  ------------------------------------------------------------
  -- HANDLER departments/
  ------------------------------------------------------------
  l_pattern := 'departments/';
  ords.define_template(
    p_module_name    => l_module_name
   ,p_pattern => l_pattern
  );

  ------------------------------------
  -- GET departments/
  ------------------------------------
  ords.define_handler(
    p_module_name       => l_module_name
   ,p_pattern => l_pattern
   ,p_source_type  => ords.source_type_collection_feed
   ,p_source       => q'[

    select deptno "$self",
           dept.*
      from dept
```

REST Definitions
Handler - Types

- Handler – types
  - *SQL Query (legacy) (source_type_query)*
  - *SQL Query (eine Zeile) (legacy) (source_type_query_one_row)*
  - Collection (source_type_collection_feed)
  - Collection Item (source_type_collection_item)
  - Feed (source_type_feed)
  - PL/SQL (source_type_plsql)
    - Generate everything manually myself
  - Media (source_type_media)
    - Binary representations

REST Definitions
Handler - Typen

- Handler – Typ: SQL Query (legacy) (source_type_query)
  - Contains a link to itself

```
select emp.*
  from emp
```


```
{
  - items: [
      - {
            empno: 7839,
            ename: "KING",
            job: "PRESIDENT",
            mgr: null,
            hiredate: "1981-11-16T23:00:00Z",
            sal: 1000,
            comm: 66,
            deptno: 10
        },
      - {
            empno: 7698,
            ename: "BLAKE",
            job: "MANAGER",
            mgr: 7839,
            hiredate: "1981-04-30T22:00:00Z",
            sal: 1000,
            comm: 111,
            deptno: 30
        }
    ],
  - first: {
        $ref: "http://localhost:8080/ords/ordstest/handler-test/test"
    }
}
```

REST Definitions
Handler - Typen

- Handler – Typ SQL: SQL Query (one row) (legacy) (source_type_query_one_row)

```
select emp.*
  from emp
 where empno=:empno
```


```
{
    empno: 7839,
    ename: "KING",
    job: "PRESIDENT",
    mgr: null,
    hiredate: "1981-11-16T23:00:00Z",
    sal: 1000,
    comm: 66,
    deptno: 10
}
```

REST Definitions
Handler - Typen

- Handler – Typ SQL: Collection (source_type_collection_feed)

```
select emp.*
  from emp
```

- Complete incl. navigation links:
  - Self
  - Describedby
  - First (only by pagination or limit)
  - Next (only by pagination or limit)
  - Previous (only by pagination or limit)


```
{
  - items: [
    - {
        empno: 7839,
        ename: "KING",
        job: "PRESIDENT",
        mgr: null,
        hiredate: "1981-11-16T23:00:00Z",
        sal: 1000,
        comm: 66,
        deptno: 10
      },
    - {
        empno: 7698,
        ename: "BLAKE",
        job: "MANAGER",
        mgr: 7839,
        hiredate: "1981-04-30T22:00:00Z",
        sal: 1000,
        comm: 111,
        deptno: 30
      }
  ],
  hasMore: true,
  limit: 2,
  offset: 0,
  count: 2,
  - links: [
    - {
        rel: "self",
        href:
        "http://localhost:8080/ords/ordstest/handler-test/emps-collection/"
      },
    - {
        rel: "describedby",
        href:
        "http://localhost:8080/ords/ordstest/metadata-catalog/handler-test/emps-collection/"
      },
    - {
        rel: "first",
        href:
        "http://localhost:8080/ords/ordstest/handler-test/emps-collection/?limit=2"
      },
    - {
        rel: "next",
        href:
        "http://localhost:8080/ords/ordstest/handler-test/emps-collection/?offset=2&limit=2"
      }
  ]
}
```

REST Definitions
Handler - Typen

- Handler – Typ SQL: Collection Item (source_type_collection_item)
  - Contains a link to the collection itself

```
select emp.*
  from emp
where empno=:empno
```

REST Definitions
Handler - Typen

- Handler – Typ SQL: Feed (source_type_feed)

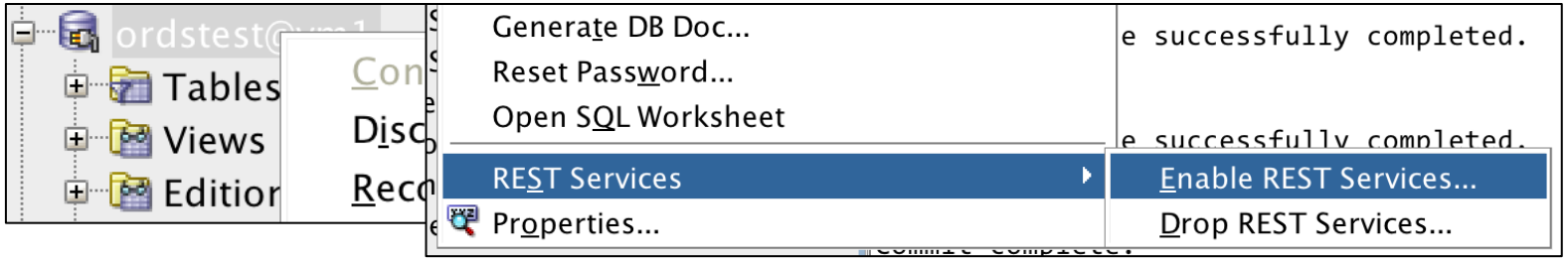
```
select emp.*
  from emp
```

Use Cases

Use Cases
Enable REST in Schema

First step: Enable REST capabilities for a schema in the database

- Using the GUI (right-click on the connection)



- Using the command line / API

```
BEGIN
   ORDS.ENABLE_SCHEMA (p_enabled             => TRUE,
                       p_schema              => 'ORDSTEST',
                       p_url_mapping_type    => 'BASE_PATH',
                       p_url_mapping_pattern => 'ordstest',
                       p_auto_rest_auth      => FALSE);

   COMMIT;
END;
```

Use Cases
Navigation and Links

Implement navigation links to navigate between the different ressources

- Links used for:
  - Link to the current row
  - Link to an image or an embedded list (ressource orders can contain a list to the related order items)
  - Link to the parent
  - Link to other “siblings” using relative paths, e.g. ../..

```
select deptno "$self",
           dept.* ,
           deptno || '/employees/' "$employees"
      from dept
     where deptno=to_number(:deptno)
```

Demo

Use Cases
Modify ressources using POST, PUT and DELETE

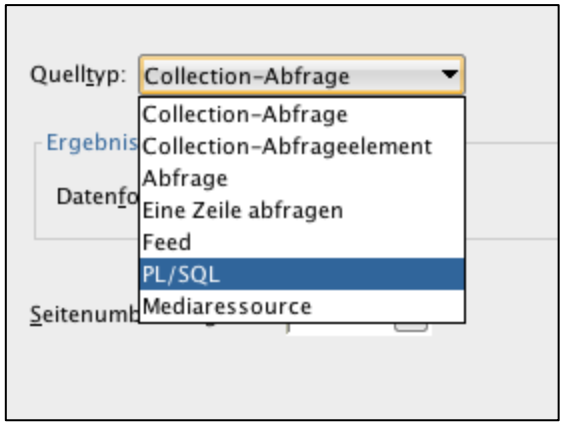
Modify ressources using POST, PUT and DELETE

- Create a new ressource (POST)
- Update a ressource (PUT)
- Delete a ressource (DELETE)

Demo

Use Cases
PL/SQL Handler – implement everything yourself

- Render everything manually with PL/SQL yourself
  - GET with Typ PL/SQL
  - Use OWA Toolkit to write it out
    - APEX_JSON, PL/JSON
    - 12c JSON Funktionen



Arbeitsblatt | Query Builder

```
1  begin
2    htp.p(q'[
3
4  {
5    "id": 10,
6    "name": "Administration",
7    "location": {
8      "id": 1700,
9      "streetAddress": "2004 Charade Rd",
10     "postalCode": "98199",
11     "country": {
12       "id": "US",
13       "name": "United States of America",
14       "regionId": 2
15     }
16   }
17 }
18
19
20   ]');
21
22 end;
```

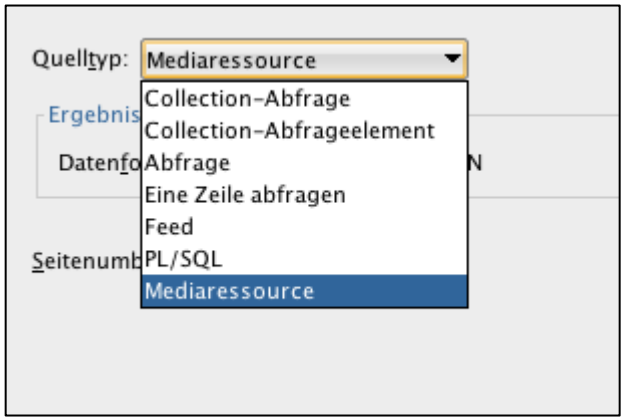
localhost:8080/ords/ordstest/demo/plsql-get-json

Apps | Google Kalender - M | Gesamtplanung-Diet | ORDS | R

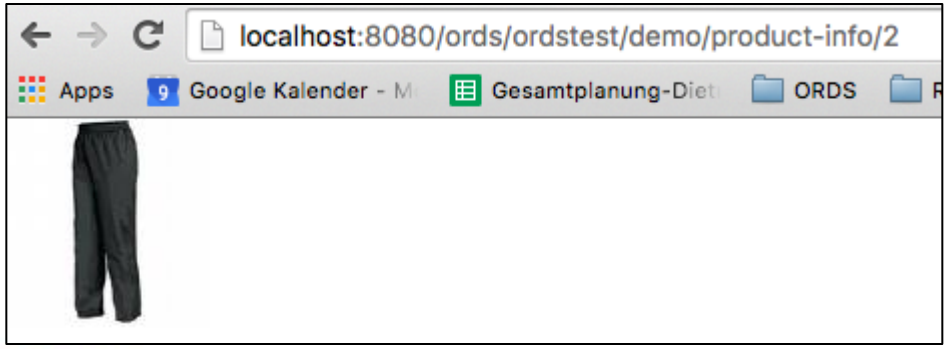
```
{
    id: 10,
    name: "Administration",
  - location: {
        id: 1700,
        streetAddress: "2004 Charade Rd",
        postalCode: "98199",
      - country: {
            id: "US",
            name: "United States of America",
            regionId: 2
        }
    }
}
```

Use Cases
Media-Ressourcen

- Display an image
  - GET handler (type Mediaressource)



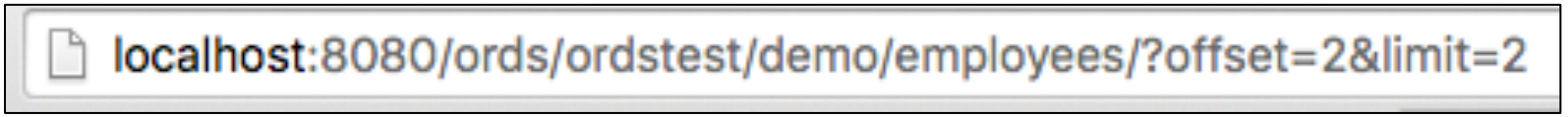

```
select mimetype, product_image
  from demo_product_info
 where product_id=to_number(:product_id)
```



Use Cases
Pagination

- Pagination
  - Allows to paginate through the result set
  - Only applicable for handler type collection (source_type_collection_feed)
  - http://docs.oracle.com/cd/E56351_01/doc.30/e56293/develop.htm#BABIHBDH
  - Pattern: `GET http://<HOST>:<PORT>/ords/<SchemaAlias>/<ObjectAlias>/?offset=<Offset>&limit=<Limit>`
  - Also creates the links "NEXT", "PREVIOUS" und "FIRST" mit

- Example:




```
- {
      rel: "first",
      href: "http://localhost:8080/ords/ordstest/demo/employees/?limit=2"
  },
- {
      rel: "next",
      href: "http://localhost:8080/ords/ordstest/demo/employees/?offset=4&limit=2"
  },
- {
      rel: "prev",
      href: "http://localhost:8080/ords/ordstest/demo/employees/?limit=2"
  }
```

Use Cases
Filter and Sort

- Result Set Filtering
  - Query Syntax to filter a collection
  - Only applicable for handler type collection (source_type_collection_feed)
  - http://docs.oracle.com/cd/E56351_01/doc.30/e56293/develop.htm#AELIG90104

Pattern: `GET http://<HOST>:<PORT>/ords/<SchemaAlias>/<ObjectAlias>/?q=<FilterClause>`

Example: `GET http://localhost:8080/ords/ordstest/emp/?q={"deptno":{"$lte":20}}`

- Sorting / Order By
  - Query Syntax to sort a collection
  - Only applicable for handler type collection (source_type_collection_feed)
  - http://docs.oracle.com/cd/E56351_01/doc.30/e56293/develop.htm#AELIG90104

Use Cases
Parameter

- Input parameters
  - Implicit
    - All variables that are passed in the URL or in the content body
    - `:content_type` (varchar2, z.B. application/json)
    - `:body` (als BLOB)
  - Explicit (using declatrative parameters)
    - All regular http header variables
    - All variables that are passed in the URL or in the content body

SQL-Arbeitsblatt | Parameter | Details

Name	Bind-Parameter	Zugriffsmethode	Quelltyp	Datentyp
empno	empno	IN	URI	INTEGER
User-Agent	user_agent	IN	HTTP HEADER	STRING
msg	msg	IN	URI	STRING
X-APEX-STATUS-CODE	status	OUT	HTTP HEADER	STRING

Use Cases
Parameter

- Output parameters
  - Explicit (using declatrative parameters)
    - Return a response by setting a simple bind variable which is then converted automatically to JSON by ORDS
    - Set an http Response Code (Pseudo-Header: `X-APEX-STATUS-CODE`), e.g. 201, 403
    - Redirect to a different URL (Pseudo-Header: `X-APEX-FORWARD`)
    - Set a http header variable

SQL-Arbeitsblatt | Parameter | Details

Name	Bind-Parameter	Zugriffsmethode	Quelltyp	Datentyp
empno	empno	IN	URI	INTEGER
User-Agent	user_agent	IN	HTTP HEADER	STRING
msg	msg	IN	URI	STRING
X-APEX-STATUS-CODE	status	OUT	HTTP HEADER	STRING

Demo

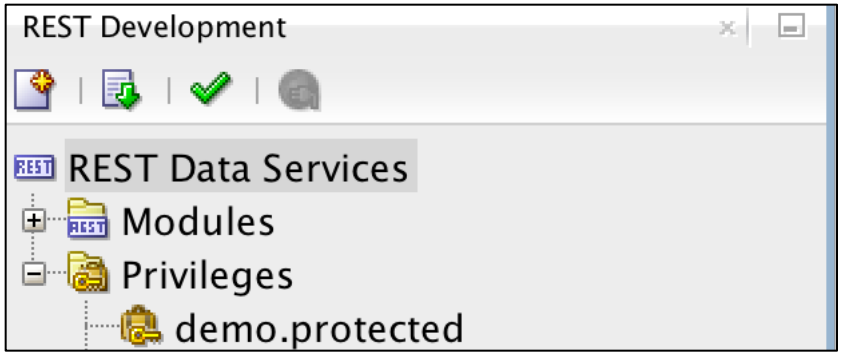
Security

Security

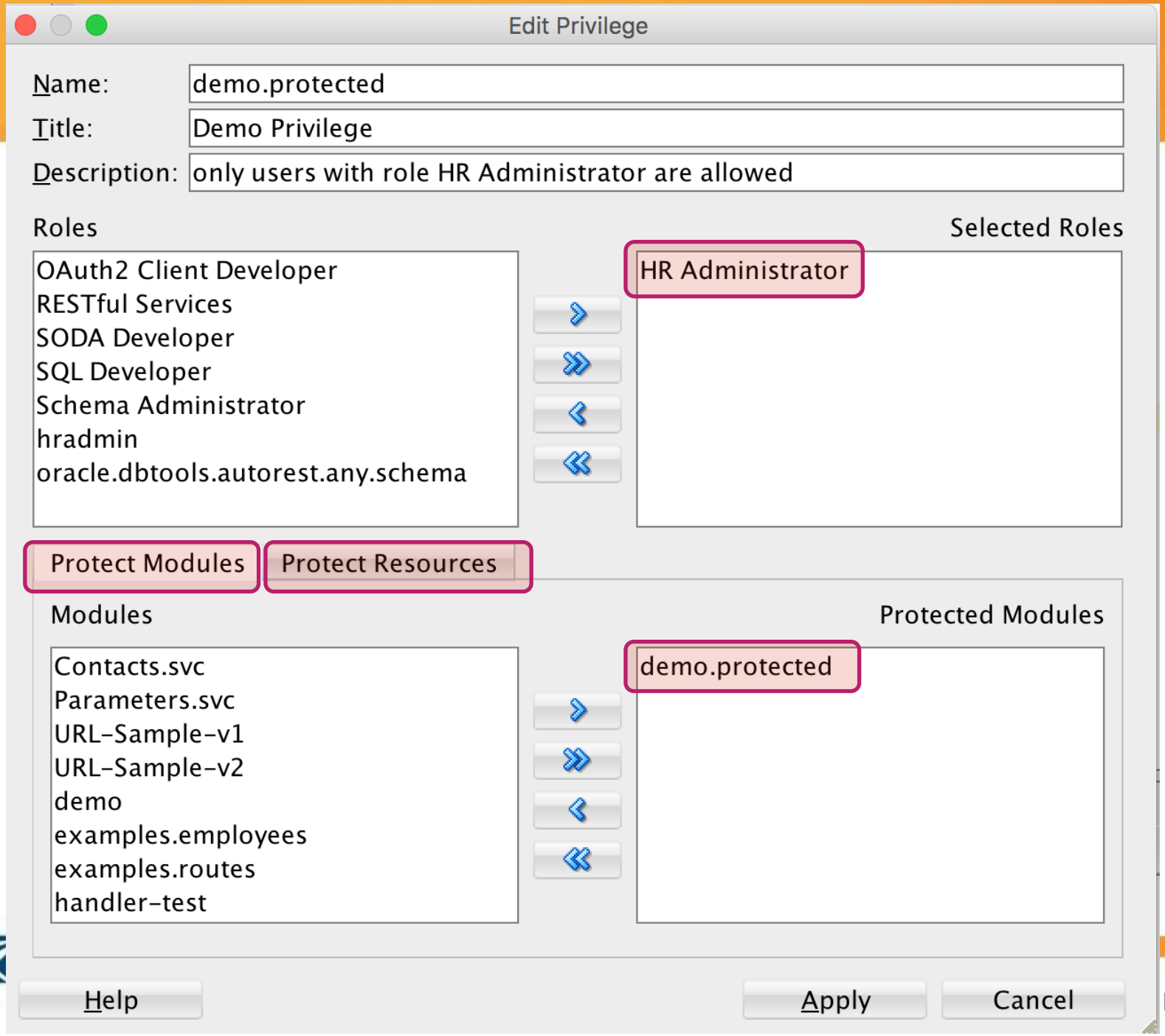
- Different ways of authenticating the current user
  - Authentication using the integrated password store ("credentials" file – just recommended for development and test environments)
  - Authentication using the application server (authentication is delegated, e.g. to Glassfish)
- Authentication using OAUTH2
  - Established standard – used widely
  - Basically controls a "session" between client/server and you still need to authenticate with the appserver
- More details to using OAUTH2 with ORDS: Articles from Carsten Czarski (in German, but can be translated using Google Translator)
  - http://json-rest-oracledb.blogspot.de/2015/12/vorher-anmelden-bitte-authentifizierung.html
  - http://json-rest-oracledb.blogspot.de/2016/01/ords-und-3-legged-oauth-so-gehts.html

Security

- Authorization := Protect access to ressources for certain user roles
- Create a ROLE first (only possible through the API)
- Create a privilege to protect a full module or just a URI pattern



- Cannot require protection just for a specific method, e.g. limit access to PUT, POST, DELETE and allow GET for everybody.
  - Perhaps using two modules:
    - `/public/departments/` (implement GET handler)
    - `/protected/departments/` (implement POST, PUT, DELETE handler)


Edit Privilege
Name:
demo.protected
Title:
Demo Privilege
Description:
only users with role HR Administrator are allowed
Roles
Selected Roles
OAuth2 Client Developer
RESTful Services
SODA Developer
SQL Developer
Schema Administrator
hradmin
oracle.dbtools.autorest.any.schema
HR Administrator
Protect Modules
Protect Resources
Modules
Protected Modules
Contacts.svc
Parameters.svc
URL-Sample-v1
URL-Sample-v2
demo
examples.employees
examples.routes
handler-test
demo.protected
Help
Apply
Cancel

Demo

Auto-REST

Auto-REST

Quickly Auto-REST enable a database table or view

- Pros:
  - Fast and easy
  - Can do some clever things using INSTEAD_OF triggers on the view
- Cons:
  - Can't use the authenticated :current_user variable to figure out the user identity which is required for logging purposes

Use Cases
Enable REST in Schema

Enable REST capabilities for a table or view

- Using the GUI (right-click on the table/view)
  - „Enable REST Service“

- Using the command line / API

```
BEGIN
    ORDS.ENABLE_OBJECT(p_enabled => TRUE,
                       p_schema => 'ORDSTEST',
                       p_object => 'DEPT',
                       p_object_type => 'TABLE',
                       p_object_alias => 'dept',
                       p_auto_rest_auth => FALSE);
   COMMIT;
END;
```

Demo

Debugging / Troubleshooting

Debugging / Troubleshooting

- Display error messages directly in the browser (only use on development / test environments, not production!)
  - Modify `default.xml`

```
<entry key="debug.debugger">true</entry>
<entry key="debug.printDebugToScreen">true</entry>
```

- Full logging with all details using java.util.logging
  - https://cdivilly.wordpress.com/2013/03/08/configuring-logging-in-oracle-application-express-listener-2-0-1/

Tools

Tools

- Command line tool: curl - https://curl.haxx.se/
- Advanced REST Client (for Google Chrome)
  - https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjeloo

- PLSQL logger
  - https://github.com/OraOpenSource/Logger

Further Reading

Further Reading

- Slides to download: http://daust.blogspot.de
- Wikipedia: http://en.wikipedia.org/wiki/Representational_State_Transfer
- REST – API Design
  - http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api
  - https://www.thoughtworks.com/de/insights/blog/rest-api-design-resource-modeling
  - http://blog.octo.com/en/design-a-rest-api/
  - https://restful-api-design.readthedocs.org/en/latest/intro.html
  - http://blog.mwaysolutions.com/2014/06/05/10-best-practices-for-better-restful-api/
- Carsten Czarski Blog about REST: http://json-rest-oracledb.blogspot.de/ (can be translated using google translator)

Weitere Informationsquellen

- RESTful Web Services, by Leonard Richardson and Sam Ruby, available from O'Reilly Media at http://oreilly.com/catalog/9780596529260/
- The source: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm mostly chapters 5 and 6
- A nice 14 minute video introduction: http://www.youtube.com/watch?v=YCcAE2SCQ6k
- HTTP spec: http://tools.ietf.org/html/rfc2616
- URI spec: http://tools.ietf.org/html/rfc3986
- JSON format: http://json.org/

1-Day Developer Workshop ORDS

- In Cologne in June (in German)
- Online as a full day webinar in July (in English)
- Topics
  - Setup / configuration for APEX / mod_plsql and REST
  - Real World project (more complex examples)
  - Different use cases with lots of hands-ons
  - Authentication using WLS, Glassfish and Tomcat
  - OAUTH 2 flow implementation

Contact

Dietmar Aust
Opal-Consulting, Köln

www.opal-consulting.de
daust.blogspot.com
dietmar.aust@opal-consulting.de