Oracle ORDS 101 - Jumpstart your Development

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

Oracle Database Developer Choice Awards

AND THE “DEVVY” GOES TO...

The Oracle Database Developer Choice Awards celebrate and recognize technical expertise and contributions in the Oracle Database community. As longtime and new users of Oracle Database move to the Cloud and take advantage of this exciting new architecture, community experts will play a critical role in helping them succeed. The "Devvy" awards were announced during the YesSQL! Celebration at Oracle OpenWorld 2015.
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 O Gh.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 := Representational State Transfer term coined in 2000 by Roy Fielding

► 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 identifiable
  - 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 resources
  - GET, POST, PUT, DELETE, OPTIONS, HEAD

- We use very few verbs to operate on many different nouns.
What is REST?

Methods

Communication of success and error messages through standard HTTP Response codes 1xx, 2xx, 3xx, 4xx, 5xx


**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 number.

<table>
<thead>
<tr>
<th>Status Code Range</th>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1xx Informational</td>
<td>100 Continue</td>
<td>Continue with processing</td>
</tr>
<tr>
<td></td>
<td>101 Switching Protocols</td>
<td>Switching Protocols</td>
</tr>
<tr>
<td></td>
<td>102 Processing (WebCGI)</td>
<td>Processing (WebCGI)</td>
</tr>
<tr>
<td>2xx Success</td>
<td>200 OK</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>201 Created</td>
<td>Created</td>
</tr>
<tr>
<td></td>
<td>202 Accepted</td>
<td>Accepted</td>
</tr>
<tr>
<td></td>
<td>204 No Content</td>
<td>No Content</td>
</tr>
<tr>
<td></td>
<td>205 Reset Content</td>
<td>Reset Content</td>
</tr>
<tr>
<td></td>
<td>206 Partial Content</td>
<td>Partial Content</td>
</tr>
<tr>
<td></td>
<td>207 Multi-Status (WebDAV)</td>
<td>Multi-Status (WebDAV)</td>
</tr>
<tr>
<td></td>
<td>208 Already Reported</td>
<td>Already Reported</td>
</tr>
<tr>
<td>3xx Redirection</td>
<td>300 Multiple Choices</td>
<td>Multiple Choices</td>
</tr>
<tr>
<td></td>
<td>301 Moved Permanently</td>
<td>Moved Permanently</td>
</tr>
<tr>
<td></td>
<td>302 Found</td>
<td>Found</td>
</tr>
<tr>
<td></td>
<td>303 See Other</td>
<td>See Other</td>
</tr>
<tr>
<td></td>
<td>304 Not Modified</td>
<td>Not Modified</td>
</tr>
<tr>
<td></td>
<td>305 Use Proxy</td>
<td>Use Proxy</td>
</tr>
<tr>
<td></td>
<td>306 Temporary Redirect</td>
<td>Temporary Redirect</td>
</tr>
<tr>
<td>4xx Client Error</td>
<td>400 Bad Request</td>
<td>Bad Request</td>
</tr>
<tr>
<td></td>
<td>401 Unauthorized</td>
<td>Unauthorized</td>
</tr>
<tr>
<td></td>
<td>402 Payment Required</td>
<td>Payment Required</td>
</tr>
<tr>
<td></td>
<td>403 Forbidden</td>
<td>Forbidden</td>
</tr>
<tr>
<td></td>
<td>404 Not Found</td>
<td>Not Found</td>
</tr>
<tr>
<td></td>
<td>405 Method Not Allowed</td>
<td>Method Not Allowed</td>
</tr>
<tr>
<td></td>
<td>406 Not Acceptable</td>
<td>Not Acceptable</td>
</tr>
<tr>
<td></td>
<td>407 Proxy Authentication Required</td>
<td>Proxy Authentication Required</td>
</tr>
<tr>
<td></td>
<td>408 Request Timeout</td>
<td>Request Timeout</td>
</tr>
<tr>
<td></td>
<td>409 Conflict</td>
<td>Conflict</td>
</tr>
<tr>
<td></td>
<td>410 Gone</td>
<td>Gone</td>
</tr>
<tr>
<td></td>
<td>411 Length Required</td>
<td>Length Required</td>
</tr>
<tr>
<td></td>
<td>412 Precondition Failed</td>
<td>Precondition Failed</td>
</tr>
<tr>
<td></td>
<td>413 Request Entity Too Large</td>
<td>Request Entity Too Large</td>
</tr>
<tr>
<td></td>
<td>414 Request-URI Too Long</td>
<td>Request-URI Too Long</td>
</tr>
<tr>
<td></td>
<td>415 Unsupported Media Type</td>
<td>Unsupported Media Type</td>
</tr>
<tr>
<td></td>
<td>416 Requested Range Not Satisfiable</td>
<td>Requested Range Not Satisfiable</td>
</tr>
<tr>
<td></td>
<td>417 Expectation Failed</td>
<td>Expectation Failed</td>
</tr>
<tr>
<td></td>
<td>418 I’m a teapot (RFC 2324)</td>
<td>I’m a teapot (RFC 2324)</td>
</tr>
<tr>
<td></td>
<td>419 Too Many Requests</td>
<td>Too Many Requests</td>
</tr>
<tr>
<td></td>
<td>420 Enhance Your Calm (Twitter)</td>
<td>Enhance Your Calm (Twitter)</td>
</tr>
<tr>
<td></td>
<td>422 Unprocessable Entity</td>
<td>Unprocessable Entity</td>
</tr>
<tr>
<td></td>
<td>423 Locked (WebDAV)</td>
<td>Locked (WebDAV)</td>
</tr>
<tr>
<td></td>
<td>424 Failed Dependency (WebDAV)</td>
<td>Failed Dependency (WebDAV)</td>
</tr>
<tr>
<td></td>
<td>425 Reserved for WebDAV</td>
<td>Reserved for WebDAV</td>
</tr>
<tr>
<td></td>
<td>426 Upgrade Required</td>
<td>Upgrade Required</td>
</tr>
<tr>
<td></td>
<td>428 Precondition Required</td>
<td>Precondition Required</td>
</tr>
<tr>
<td></td>
<td>429 Too Many Requests</td>
<td>Too Many Requests</td>
</tr>
<tr>
<td></td>
<td>431 Request Header Fields Too Large</td>
<td>Request Header Fields Too Large</td>
</tr>
<tr>
<td></td>
<td>444 No Response (Nginx)</td>
<td>No Response (Nginx)</td>
</tr>
<tr>
<td></td>
<td>449 Client Closed Request (Nginx)</td>
<td>Client Closed Request (Nginx)</td>
</tr>
<tr>
<td>5xx Server Error</td>
<td>500 Internal Server Error</td>
<td>Internal Server Error</td>
</tr>
<tr>
<td></td>
<td>501 Not Implemented</td>
<td>Not Implemented</td>
</tr>
<tr>
<td></td>
<td>502 Bad Gateway</td>
<td>Bad Gateway</td>
</tr>
<tr>
<td></td>
<td>503 Service Unavailable</td>
<td>Service Unavailable</td>
</tr>
<tr>
<td></td>
<td>504 Gateway Timeout</td>
<td>Gateway Timeout</td>
</tr>
<tr>
<td></td>
<td>505 HTTP Version Not Supported</td>
<td>HTTP Version Not Supported</td>
</tr>
<tr>
<td></td>
<td>506 Variant Also Negotiates (Experimental)</td>
<td>Variant Also Negotiates (Experimental)</td>
</tr>
<tr>
<td></td>
<td>507 Insufficient Storage (WebDAV)</td>
<td>Insufficient Storage (WebDAV)</td>
</tr>
<tr>
<td></td>
<td>508 Loop Detected (WebDAV)</td>
<td>Loop Detected (WebDAV)</td>
</tr>
<tr>
<td></td>
<td>509 Bandwidth Limit Exceeded (Apache)</td>
<td>Bandwidth Limit Exceeded (Apache)</td>
</tr>
<tr>
<td></td>
<td>510 Not Extended</td>
<td>Not Extended</td>
</tr>
<tr>
<td></td>
<td>511 Network Authentication Required</td>
<td>Network Authentication Required</td>
</tr>
<tr>
<td></td>
<td>598 Bandwidth Limit Exceeded (Apache)</td>
<td>Bandwidth Limit Exceeded (Apache)</td>
</tr>
<tr>
<td></td>
<td>599 Network connect timeout error</td>
<td>Network connect timeout error</td>
</tr>
</tbody>
</table>

---

**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 resource 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?
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

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

1. Browser RESTful get request

2. ORDS maps to “EMPLOYEES” SQL

3. SQL Call over JDBC

4. DB returns JDBC Results

5. JSON
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
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?

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
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
```

<table>
<thead>
<tr>
<th>SESSION_USER</th>
<th>PARSING_SCHEMA</th>
<th>APPLICATION_USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>APEX_PUBLIC_USER</td>
<td>TRAINING</td>
<td>DIETMAR.AUST</td>
</tr>
</tbody>
</table>
Management of the REST definitions with SQL Developer and the API
Management of the REST definitions 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

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

- **Oracle REST Data Services**
  - http/https
  - jdbc

  - **Local filestore, Admin user needs role “SQL Developer”**
  - Credentials: `conf/credentials`

- **APEX_050000**
- **ORDS_METADATA**
REST Definitions
Modules, Resource 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!!!
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
**Handler – Typ: SQL Query (legacy) (source_type_query)**
- Contains a link to itself

```sql
select emp.*
from emp
```
REST Definitions
Handler - Typen

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

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

```json
{
  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)

```sql
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)
Handler – Typ SQL: Collection Item (source_type_collection_item)

- Contains a link to the collection itself

```sql
select emp.*
from emp
where empno=:empno
```
Handler – Typ SQL: Feed (source_type_feed)

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

```
{
  - items: [ 
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    + {...},
    - uri: {
        $ref: "http://localhost:8080/ords/ordstest/handler-test/7934"
    },
    empno: 7934,
    ename: "MILLER",
    job: "CLERK",
    mgr: 7782,
    hiredate: "1982-01-22T23:00:00Z",
    sal: 1000,
    comm: null,
    deptno: 10
  },
  - first: {
    $ref: "http://localhost:8080/ords/ordstest/handler-test/1"
  }
}
```
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. ../..
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
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
Display an image

- GET handler (type Mediaressource)

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

- Allows to paginate through the result set
- Only applicable for handler type collection (source_type_collection_feed)
- [Link](http://docs.oracle.com/cd/E56351_01/doc.30/e56293/develop.htm#BABIHBDD)
- Also creates the links “NEXT”, “PREVIOUS” and “FIRST” links

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)
  ▪ [link](http://docs.oracle.com/cd/E56351_01/doc.30/e56293/develop.htm#AELIG90104)

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: GET</td>
<td><a href="http://localhost:8080/ords/ordstest/emp/?q=%7B%22deptno%22:%7B%22$lte%22:20%7D%7D">http://localhost:8080/ords/ordstest/emp/?q={&quot;deptno&quot;:{&quot;$lte&quot;:20}}</a></td>
</tr>
</tbody>
</table>

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

| localhost:8080/ords/ordstest/demo/employees/?{"$orderby":{"SALARY":"ASC","ENAME":"DESC"}} |
### 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 declarative parameters)**
  - All regular http header variables
  - All variables that are passed in the URL or in the content body

<table>
<thead>
<tr>
<th>SQL-Arbeitsblatt</th>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Bind-Parameter</td>
<td>Zugriffsmethode</td>
</tr>
<tr>
<td>empno</td>
<td>empno</td>
<td>IN</td>
</tr>
<tr>
<td>User-Agent</td>
<td>user_agent</td>
<td>IN</td>
</tr>
<tr>
<td>msg</td>
<td>msg</td>
<td>IN</td>
</tr>
<tr>
<td>X-APEX-STATUS-CODE</td>
<td>status</td>
<td>OUT</td>
</tr>
</tbody>
</table>
Output parameters

- Explicit (using declarative 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

<table>
<thead>
<tr>
<th>SQL-Arbeitsblatt</th>
<th>Parameter</th>
<th>Details</th>
<th>Bind-Parameter</th>
<th>Zugriffsmethode</th>
<th>Quelltyp</th>
<th>Datentyp</th>
</tr>
</thead>
<tbody>
<tr>
<td>empno</td>
<td>empno</td>
<td>IN</td>
<td>URI</td>
<td>IN</td>
<td>HTTP HEADER</td>
<td>STRING</td>
</tr>
<tr>
<td>User-Agent</td>
<td>user_agent</td>
<td>IN</td>
<td>HTTP HEADER</td>
<td>STRING</td>
<td>HTTP HEADER</td>
<td>STRING</td>
</tr>
<tr>
<td>msg</td>
<td>msg</td>
<td>IN</td>
<td>URI</td>
<td>IN</td>
<td>HTTP HEADER</td>
<td>STRING</td>
</tr>
<tr>
<td>X-APEX-STATUS-CODE</td>
<td>status</td>
<td>OUT</td>
<td>HTTP HEADER</td>
<td>STRING</td>
<td>HTTP HEADER</td>
<td>STRING</td>
</tr>
</tbody>
</table>
Demo
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-oracleblog.blogspot.de/2015/12/vorher-anmelden-bitte-authentifizierung.html](http://json-rest-oracleblog.blogspot.de/2015/12/vorher-anmelden-bitte-authentifizierung.html)
- [http://json-rest-oracleblog.blogspot.de/2016/01/ords-und-3-legged-oauth-so-gehts.html](http://json-rest-oracleblog.blogspot.de/2016/01/ords-und-3-legged-oauth-so-gehts.html)
Authorization := Protect access to resources 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)
Demo
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
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
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/](https://cdivilly.wordpress.com/2013/03/08/configuring-logging-in-oracle-application-express-listener-2-0-1/)
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

 Advanced REST client
angeboten von restforchrome.blogspot.com

► PLSQL logger
  - https://github.com/OraOpenSource/Logger
Further Reading
Further Reading

► Slides to download: http://daust.blogspot.de


► REST – API Design
  - https://www.thoughtworks.com/de/insights/blog/rest-api-design-resource-modeling
  - 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


- A nice 14 minute video introduction: [http://www.youtube.com/watch?v=YCcAE2SCQ6k](http://www.youtube.com/watch?v=YCcAE2SCQ6k)


- JSON format: [http://json.org/](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