Oracle als REST-Service mit ORDS – So geht‘s

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

ORACLE ACE

OPAL Consulting

Modern Application Development - *powered by* Oracle *und* DOAG

Neue Technologie im praktischen Unternehmenseinsatz.

7 Videos. Eine 'live' Diskussionsrunde. Ab Januar 2016. Für Euch kostenlos.

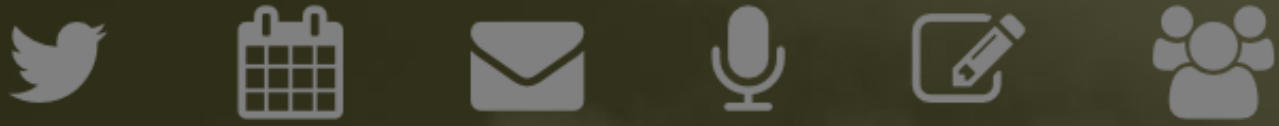
DOAG approved

Start JSON HTML5 REST Document DB Node.js Python NoSQL Q & A

11. Februar: Oracle als REST-Service mit ORDS - So geht's.

Für öffentliche Schnittstellen im Internet sind REST-Services inzwischen der Standard - alle großen Anbieter stellen ihre Dienste als REST-Services bereit. Auch im Unternehmenseinsatz bestechen REST-Services durch ihre einfache Implementierbarkeit, Nutzung und hohe Stabilität. Das Video zeigt, wie einfach sich REST-Services mit Oracle REST Data Services auf der Oracle-Datenbank aufsetzen lassen. Eine vorhanden Tabelle ist mit wenigen Handgriffen "REST-Enabled".

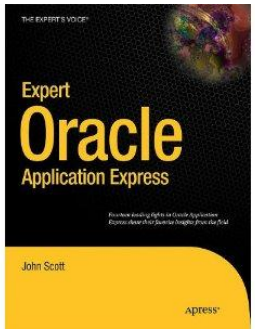
Dietmar Aust - Jetzt ansehen!



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

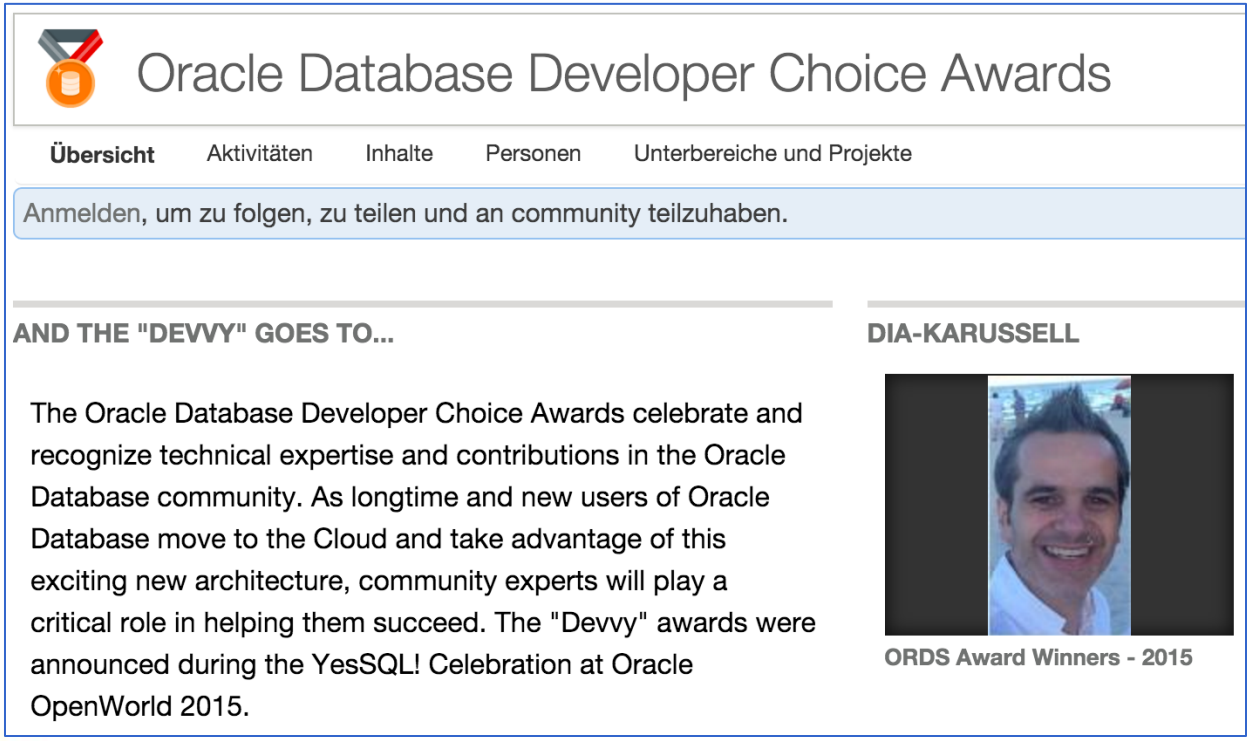
ORACLE ACE



OPAL Consulting

Dietmar Aust

- 2015 Datenbank Entwickler des Jahres in der ORDS Kategorie
  - Schwerpunktmäßig für meine Kenntnisse des ORDS als Technologie für APEX
  - RESTful Services sind noch relativ jung mit ORDS … werden jedoch immer wichtiger und haben großes Potential



Agenda

Agenda

- Abgrenzung (was ist drin / was nicht)
- Was ist REST?
- Was ist ORDS?
- Management der REST – Definitionen mit dem SQL Developer und der API
- Use Cases
  - Navigation / Links / Filter / Sortierungen / Parameter (Input / Output)
- Sicherheit
  - Authentifizierung und Autorisierung
- Weitere Informationsquellen

Abgrenzung / Inhalt des Webinars

Abgrenzung / Inhalt des Webinars

- Zielsetzung des Webinars für die Zuhörer:
  - Überblick gewinnen
  - ORDS Architektur und konkrete Umsetzung verstehen
- Was machen wir NICHT?
  - Clientseitige Programmierung von REST Webservices
  - Auto-REST Aktivierung für Tabellen und Views
- Die Folien werden bereitgestellt über meinen Blog:
  - http://daust.blogspot.de

Was ist REST?

Was ist REST?
Definition

- Ist ein Architekturmodell für Anwendungen, kein Protokoll oder W3C Standard.

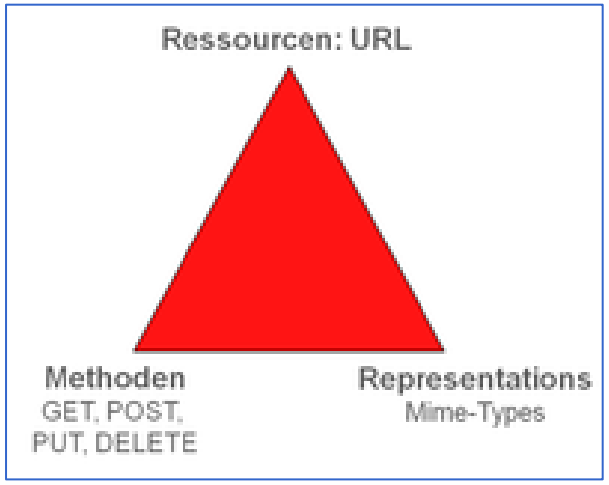
- REST := **Re**presentational **S**tate **T**ransfer aus dem Jahre 2000 von Roy Fielding
  - https://en.wikipedia.org/wiki/Representational_state_transfer

- Eigenschaften:
  - Zustandslos (100% Applikationszustand befindet sich auf dem Client)
  - Das HTTP Protokoll ist die Basis
  - Stark skalierbar
  - REST verwendet die HTTP Methoden (POST, PUT, GET, DELETE, …), um CRUD Operationen (Create / Read / Update / Delete) und andere Aktionen auf den Daten durchzuführen

- Warum?
  - Leichtgewichtige Alternative für RPC (Remote Procedure Calls) und Web Services (SOAP, WSDL, …)
  - Mittlerweile stark verbreitet durch Google, Facebook, Twitter, ... und auch, um Geschäftsprozesse zu automatisieren

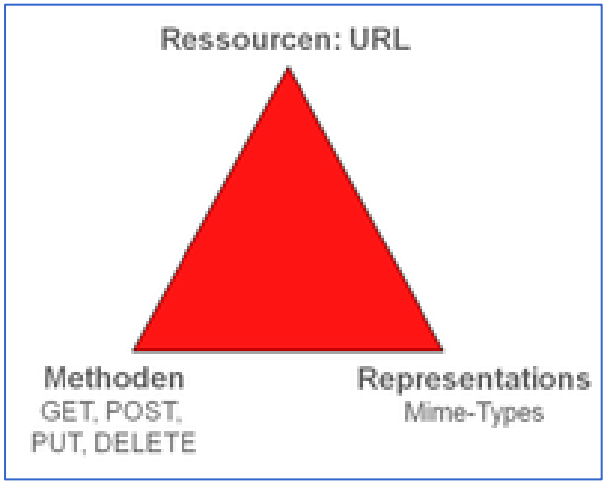
Was ist REST?
Ressourcen

- Ressourcen stellen Dienste bereit und sind eindeutig identifizierbar
  - http://api.example.com/resources/
  - http://api.example.com/resources/item17
  - http://api.example.com/resources/item17/components/
- Multiple URIs für dieselbe Ressource möglich:
  - http://example.org/NewOrleans/traffic/I10
  - http://example.org/traffic/NewOrleans/I10
- Wir modellieren die Ressourcen, keine Aktionen!
  - Verwendung der Hauptwörter im Plural
  - PUT http://example.com/accounts/12345
  - PUT http://example.com/accounts/edit/12345
  - POST http://example.com/accounts/
  - POST http://example.com/accounts/addaccount



Was ist REST?
Methoden

- Methoden dienen dem Abrufen eines konkreten Dienstes
  - Einheitliche Operationen für alle Ressourcen
  - GET, POST, PUT, DELETE, OPTIONS, HEAD
- Viele verschiedene Hauptwörter (unsere Ressourcen) und nur wenige einheitliche Verben, um damit zu arbeiten



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

Was ist REST?
Methoden

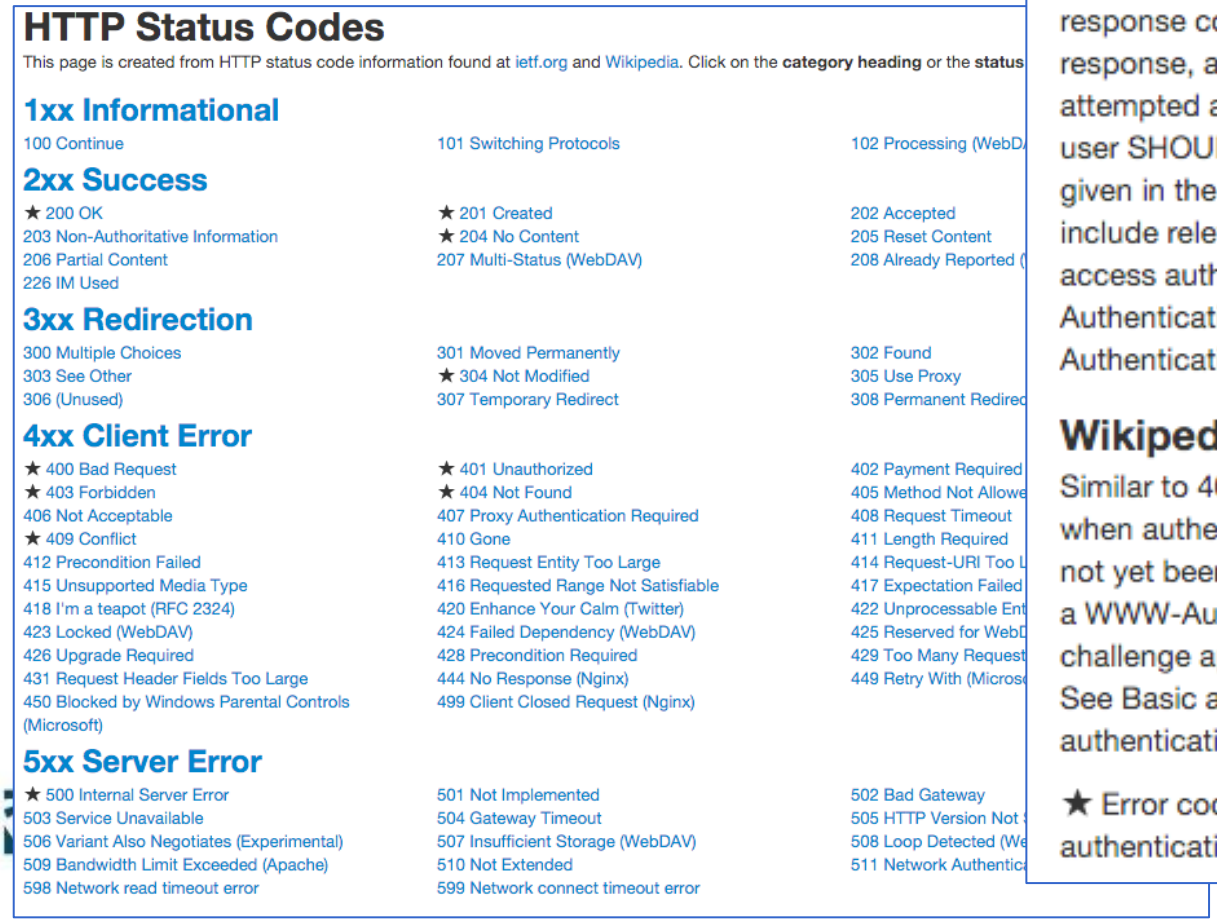
- Beispiel

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.

Was ist REST? Methoden

- Kommunikation der Fehlermeldungen über 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 Authentica
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".

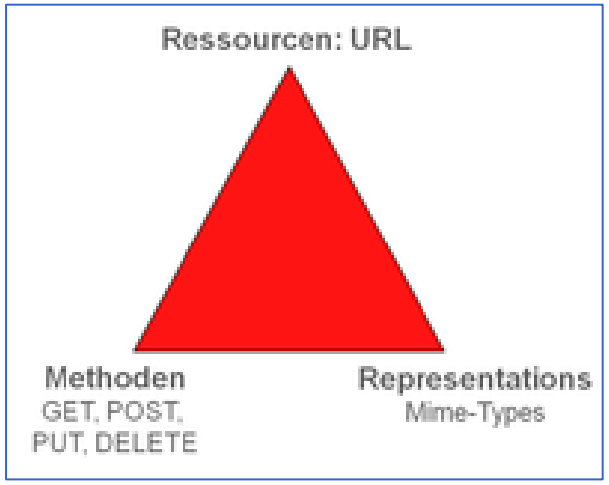
Wikipedia

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.

Was ist REST?
Repräsentationen

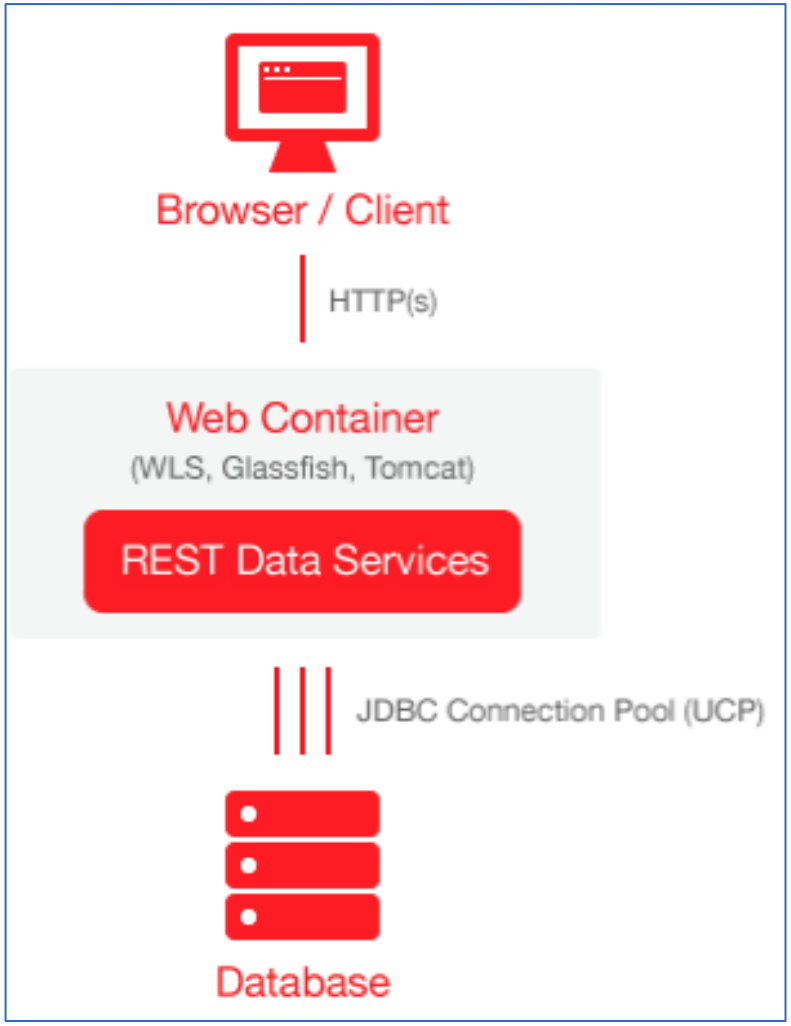
- Repräsentationen bestimmen, wie die Antworten zu interpretieren sind
  - XML Darstellung mit Mime-Type: text/xml
  - JSON Darstellung mit Mime-Type: application/JSON
- Eine Ressource kann mehrere alternative Repräsentationen haben
  - JSON, XML, CSV …
  - Werden “verhandelt”, der Client kommuniziert im Accept-Header eine Liste von Formaten, die er akzeptiert, der Server antwortet mit der besten Antwort und schickt im Content-Type Header mit, was er gesendet hat.



Was ist ORDS?

Was ist ORDS?

- Oracle Rest Data Services (ORDS)
  - Middleware Komponente J2EE im Appserver (WLS, Glassfish, Tomcat)
  - Ermöglicht Zugriff auf die Oracle Datenbank
  - Übersetzt URLs in Aufruf in der Datenbank
  - Ersetzt mod_plsql
- Drei wesentliche Einsatzgebiete
  - OWA Toolkit Applikationen
  - Oracle Application Express (APEX)
  - RESTful Webservices



Was ist ORDS?
Die Historie

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,…

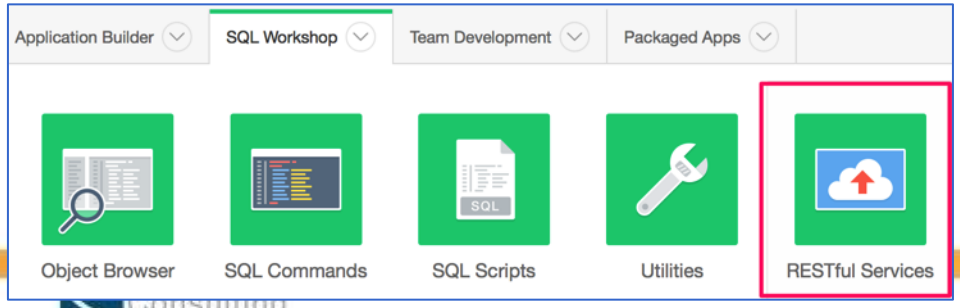
Was ist ORDS?
APEX REST vs. ORDS 3.0 REST Support

ORDS ist aktuell im Übergang zur Unabhängigkeit von APEX

- ORDS benötigt ein REPOSITORY zum Speichern der Webservice-Definitionen

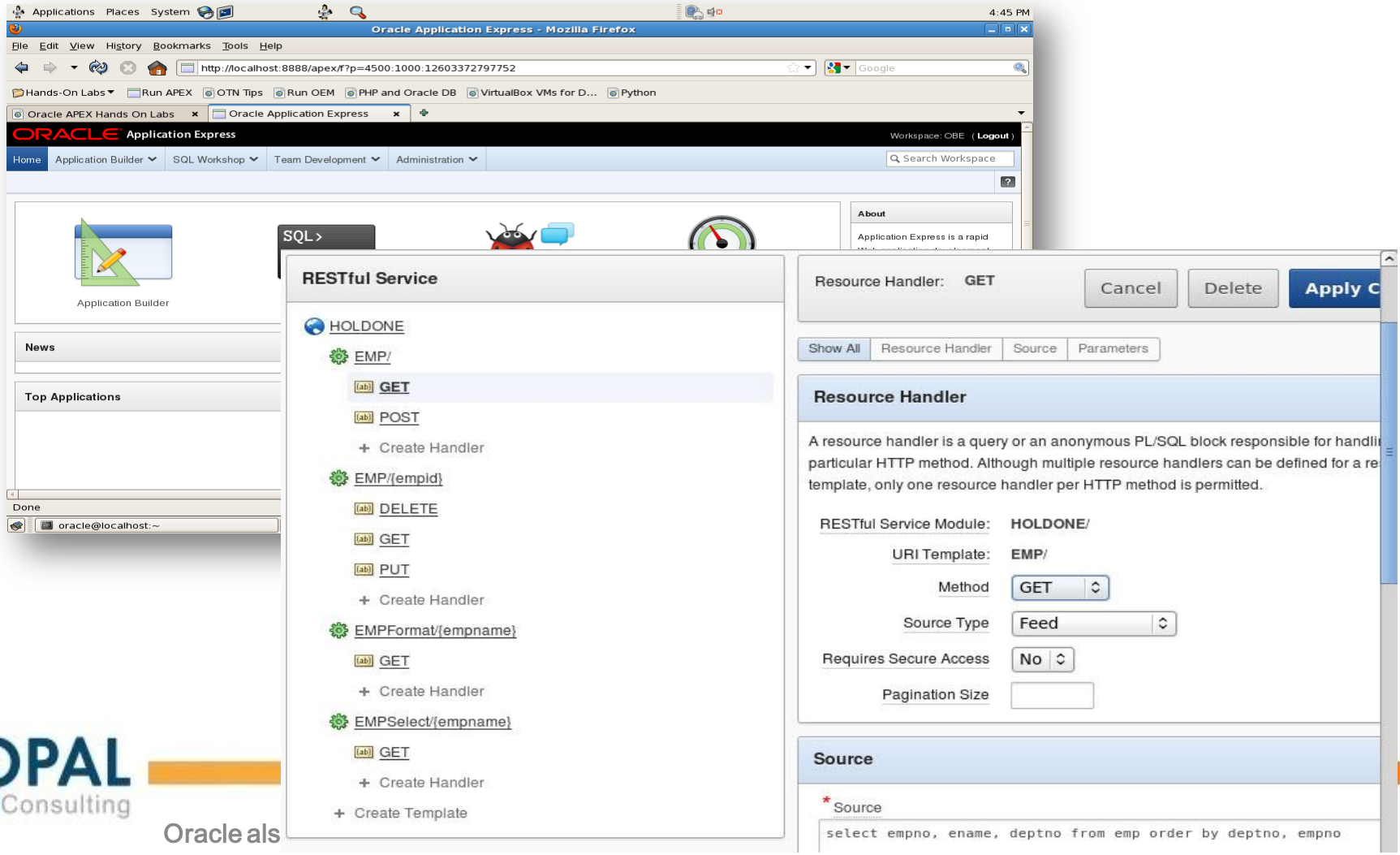
- ORDS 2.0
- Schemas
  - APEX_040200/APEX_050000
  - APEX_LISTENER
  - APEX_REST_PUBLIC_USER
- Konfiguration über
  - APEX SQL Workshop

- ORDS 3.0
- Schemas
  - ORDS_METADATA
  - ORDS_PUBLIC_USER
- Konfiguration über
  - SQL Developer
  - PL/SQL API



Was ist ORDS?
APEX REST vs. ORDS 3.0 REST Support

- APEX REST Unterstützung im SQL Workshop

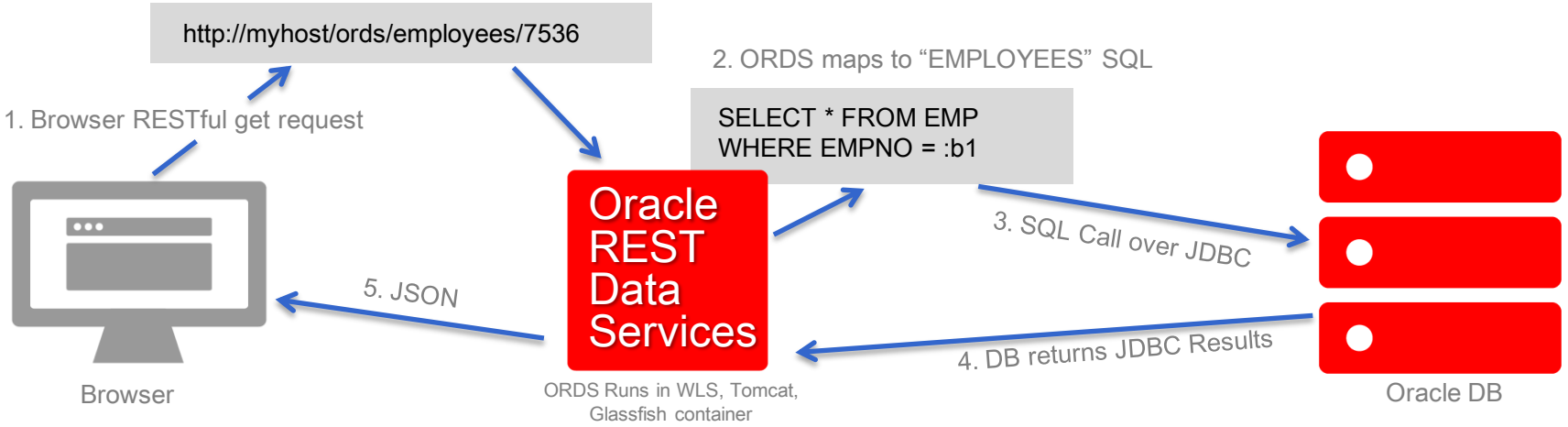


Was ist ORDS?
APEX REST vs. ORDS 3.0 REST Support

- Dokumentation unterscheidet nicht immer genau zwischen REST mit APEX und REST mit ORDS 3.0
- Die Beispiele in der Dokumentation und auch im Internet sind gemischt und zwischen den beiden Implementierungen gibt es funktionale Unterschiede
- Im weiteren Vortrag konzentrieren wir uns **NUR auf REST mit ORDS 3.0**

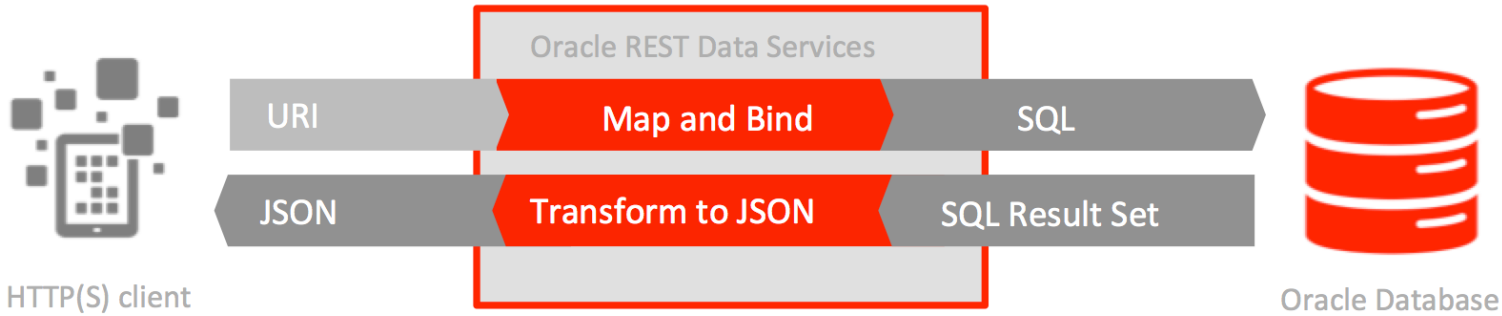
Was ist ORDS?
Architektur

- Wie funktioniert ein Aufruf?



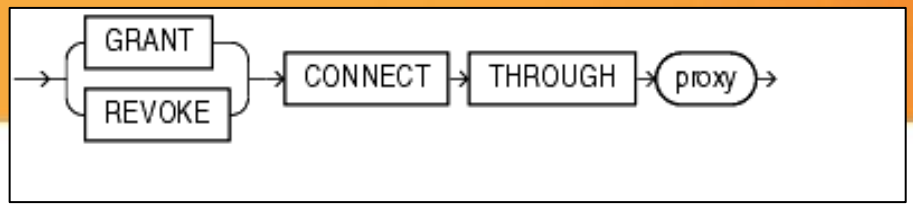
Was ist ORDS?
Architektur

- Binding und Parameter
  - Zugriff auf URI Parameter in URL oder BODY => automatisch, auch JSON
  - Explizite Parameter möglich
  - Zugriff auf Header Variablen

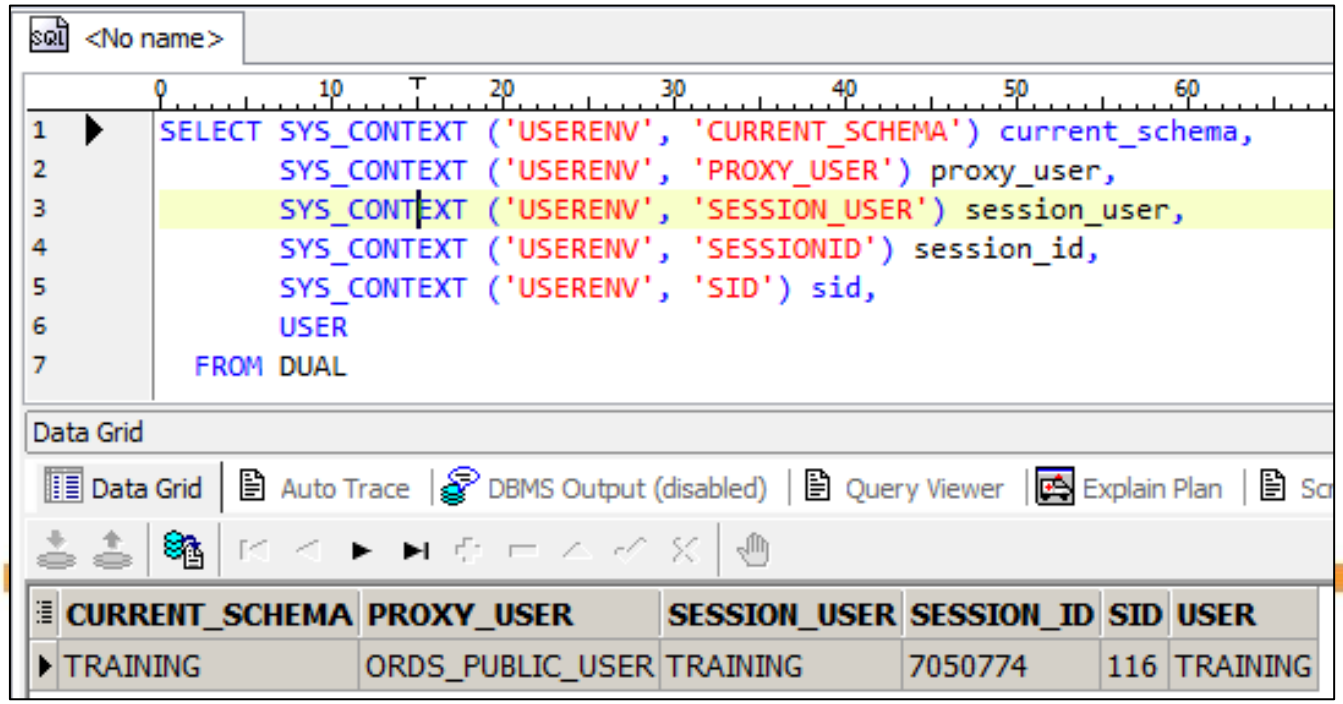


- Transform to JSON
  - Wie ? => Deklarativ (!) oder auch manuell
  - Deklarative Formate: JSON oder CSV, manuell sind alle möglich
  - Status Code kann verändert werden, HTTP Header Variablen gesetzt

Was ist ORDS? Architektur



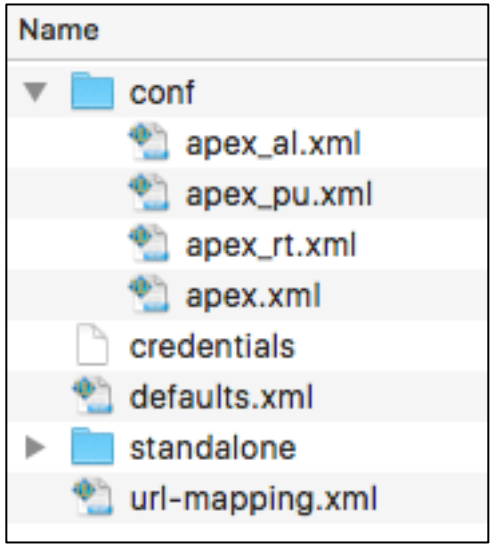
- Connection Pooling
  - Aktivierung von Oracle Usern (Schemata) über einen Proxy Connect
  - Der ORDS User ORDS_PUBLIC_USER kann sich dadurch an der DB anmelden und dann die Identität wechseln auf den Zieluser
  - Dadurch benötigen wir weniger Connection Pools und in der Summe ist er auch viel kleiner
  - Jedes SQL und PL/SQL Statement wird dann mit der originalen User Session durchgeführt (keine aktive Session mit ORDS_PUBLIC_USER)

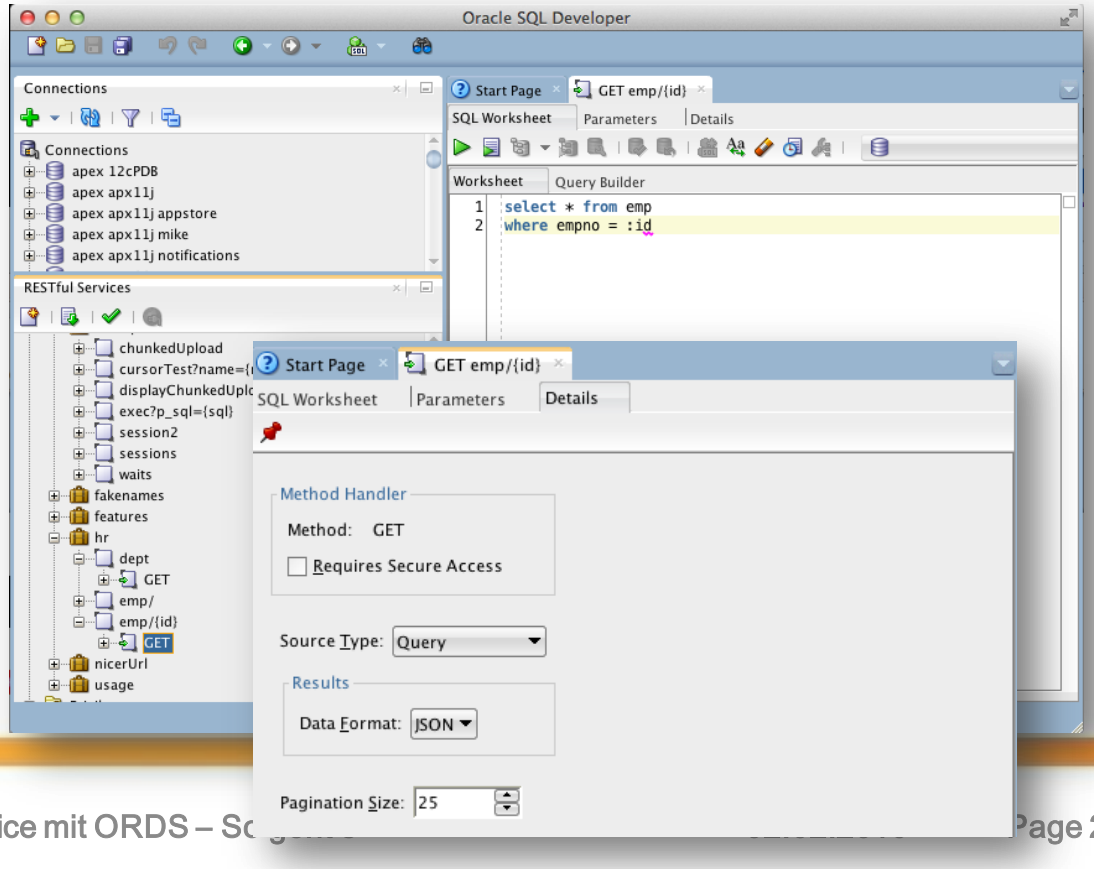


Management der REST – Definitionen mit SQL Developer und API

REST Definitionen
Management mit dem SQL Developer

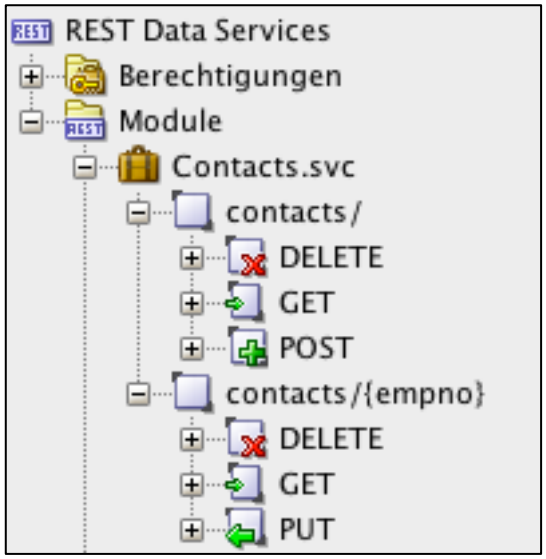
- Management der REST Definitionen mit dem SQL Developer
- Zuerst muss ein passender User im Passwort Store des ORDS mit der Rolle “SQL Developer” eingerichtet werden.





REST Definitionen
Module, Ressourcen Templates und Handler

- Aufbau der Komponenten im ORDS
  - Module
    - Resource Templates
      - Methoden / Handler (GET, PUT, POST, DELETE)



REST Definitionen
Management über die API

- Management über die PL/SQL API
- Komplette Datei - hat alle Resource Templates und Methoden zu einem Modul
- Zuerst existierende Definition löschen und dann neu installieren.
- Besonders gut geeignet für das Deployment, da alles skriptbasiert ist
- API Referenz: http://docs.oracle.com/cd/E56351_01/doc.30/e56293/ords_ref.htm#AELIG90180

url-sample.sql

```
declare
  l_moduleid   number;
  l_templateid number;
  l_handlerid  number;
begin
  -- delete module if exists
  ords_services.delete_module('URL-Sample');

  -- REST Modul anlegen
  l_moduleid := ords_services.create_module(
    p_name       => 'URL-Sample'
   ,p_uri_prefix => 'URL-Sample/'
   ,p_status     => 'PUBLISHED'
  );

  -- 1. URL-Template
  l_templateid := ords_services.add_template(
    p_module_id    => l_moduleid
   ,p_uri_template => 'employees/'
  );

  ---------------------------------------
  -- POST-Handler for the url template:
  -- POST URL-Sample/employees/
  ---------------------------------------
  l_handlerid := ords_services.add_handler(
    p_template_id  => l_templateid
   ,p_source_type  => ords.source_type_plsql
   ,p_source       => q'[

declare
  l_empno emp.empno%type;

  l_err_code number;
  l_err_msg varchar2(4000);
  l_err_stack varchar2(4000);
begin
  begin
      insert into emp(empno, ename, job, deptno)
        values (:empno, :ename, :job, :deptno)
        returning empno into l_empno;

      if l_empno is null then
```

REST Definitionen
Handler - Typen

- Handler – Typen
  - *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)
    - Alles manuell komplett selbst generieren
  - Media (source_type_media)
    - Binäre Repräsentationen

REST Definitionen
Handler - Typen

- Handler – Typ: SQL Query (legacy) (source_type_query)
  - Link auf sich selbst

```
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 Definitionen
Handler - Typen

- Handler – Typ SQL: SQL Query (eine Zeile) (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 Definitionen
Handler - Typen

- Handler – Typ SQL: Collection (source_type_collection_feed)

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

- Komplett inkl. Navigations-Links:
  - Self
  - Describedby
  - First
  - Next (nur bei pagination oder limit)
  - Previous (nur bei 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 Definitionen
Handler - Typen

- Handler – Typ SQL: Collection Item (source_type_collection_item)
  - Links auf die Collection mit integriert.

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

REST Definitionen
Handler - Typen

- 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/test"
    }
}
```

Use Cases

Use Cases
Navigation und Links

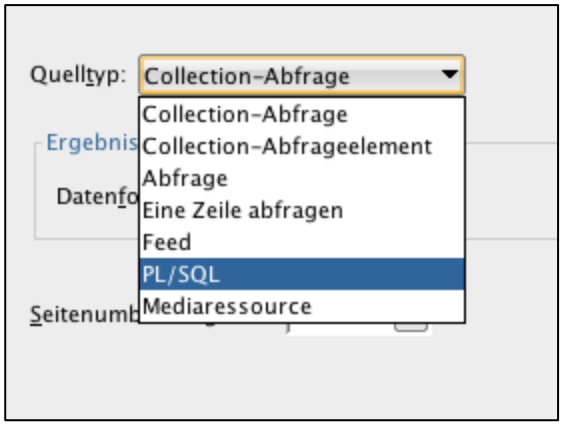
- Zuerst die allgemeine Navigation in den Daten abbilden
- Links verwenden
  - Link auf den aktuellen Datensatz
  - Link auf ein Image oder eine untergeordnete Liste (von order items zu orders z.B.)
  - Link auf den Vater
  - Link auf andere “Geschwister-Entitäten” ../.. ...

Use Cases
Daten Bearbeiten

- Datensatz erstellen (POST)
- Datensatz aktualisieren (PUT)
- Datensatz löschen (DELETE)

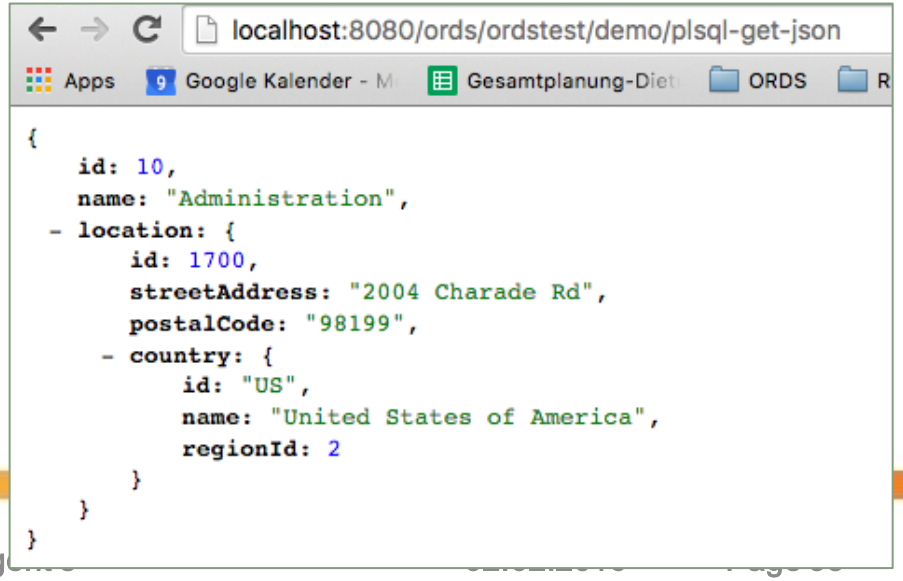
Use Cases
PL/SQL Handler – alles selbst machen

- Mit PL/SQL alles rendern
  - GET Handler vom Typ PL/SQL
  - Mit OWA Toolkit Methoden alles selbst ausgeben
    - APEX_JSON, PL/JSON
    - 12c JSON Funktionen



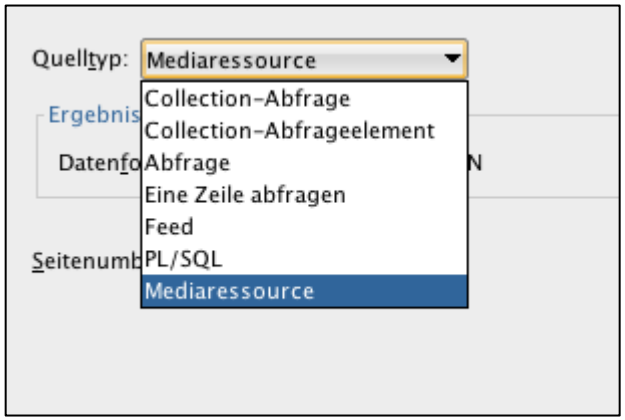
Arbeitsblatt | Query Builder

```
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;
```

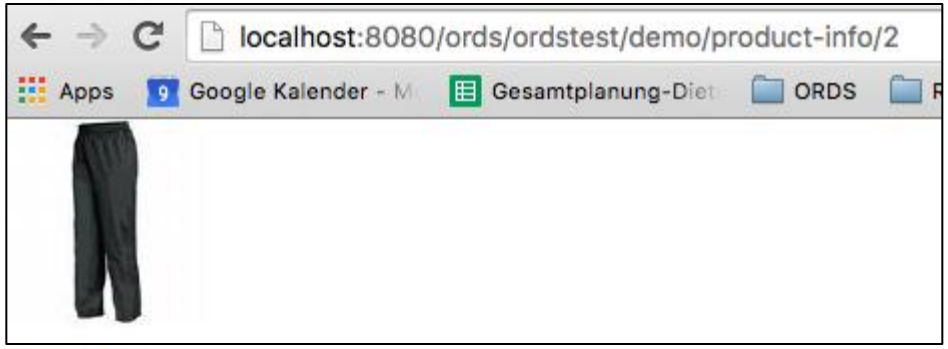


Use Cases
Media-Ressourcen

- Ein Image anzeigen
  - GET Handler vom Typ Mediaressource




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



Use Cases
Pagination

- Pagination
  - Erlaubt “Paging” durch die gesamte Ergebnismenge
  - Gilt nur für Handler-Typ 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>
  - Liefert auch gleich die Links “NEXT”, “PREVIOUS” und “FIRST” mit

- Beispiel:

```
- {
      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
Filtern und Sortieren

- Result Set Filtering
  - Query Syntax, um Collection zu filtern
  - Gilt nur für Handler-Typ 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, um Collection zu sortieren
  - Gilt nur für Handler-Typ Collection (source_type_collection_feed)
  - http://docs.oracle.com/cd/E56351_01/doc.30/e56293/develop.htm#AELIG90104

Use Cases
Parameter

- Parameterübergabe IN
  - Implizit
    - Alle Variablen, die in der URL oder im Body übergeben werden
    - :content-type (varchar2, z.B. application/json)
    - :body (als BLOB)
  - Explizit (über deklarierte Parameter)
    - Alle normalen HTTP Header Variablen
    - Alle Variablen, die in der URL oder im Body übergeben werden (hier expliziten Typ setzen)

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

- Parameterübergabe OUT
  - Explizit (über deklarierte Parameter)
    - Setzen der Response über (einfache) Variablen, die automatisch in JSON konvertiert werden
    - Einen http Response Code setzen (Pseudo-Header: X-APEX-STATUS-CODE)
    - Einen Rewrite URL auslösen (Pseudo-Header: X-APEX-FORWARD)
    - Eine http Header Variable setzen

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

Sicherheit

Sicherheit

- Authentifizierung mit dem integrierten Passwort-Store (nur zum Test / Development)
  - http://json-rest-oracledb.blogspot.de/2015/12/vorher-anmelden-bitte-authentifizierung.html
- Authentifizierung über den Applikationsserver (Authentication wird delegiert, z.B. an den Glassfish)
- Authentifizierung mit OAUTH2
  - Etablierter Standard
- Weitere Details zur OAUTH2 Integration auf dem BLOG von Carsten
  - 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

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

Debugging / Troubleshooting

Debugging / Troubleshooting

- Fehlermeldungen direkt im Browser ausgeben (nur auf dem Entwicklungssystem, nicht Produktion!)
  - Anpassung der defaults.xml Datei

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

- Volles Logging mit allen Details mittels java.util.logging
  - https://cdivilly.wordpress.com/2013/03/08/configuring-logging-in-oracle-application-express-listener-2-0-1/

Weitere Informationsquellen

Weitere Informationsquellen

- Modern Application Development Webinar – Reihe:
  https://apex.oracle.com/pls/apex/f?p=75474
- Folien zum Download: http://daust.blogspot.de
- Carsten Czarski Blog über REST: http://json-rest-oracledb.blogspot.de/
- 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/

Weitere Informationsquellen

- RESTful Web Services, by Leonard Richardson and Sam Ruby, available from O'Reilly Media at http://oreilly.com/catalog/9780596529260/
- Wikipedia: http://en.wikipedia.org/wiki/Representational_State_Transfer
- 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/

Kontakt

Dietmar Aust
Opal-Consulting, Köln

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