Oracle® REST Data Services
Quick Start Guide

Release 18.1

E94471-02

May 2019

ORACLE®

Oracle REST Data Services Quick Start Guide, Release 18.1

E94471-02

Primary Authors: Mamata Basapur, Chuck Murray

Contributors: Colm Divilly, Sharon Kennedy, Ganesh Pitchaiah, Kris Rice, Elizabeth Saunders

Contents

Preface

1 Getting Started with Oracle REST Data Services

Index

List of Figures

Preface

Oracle REST Data Services Quick start Guide is designed to let you get started quickly developing RESTful services using Oracle REST Data Services.

Topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions
- Third-Party License Information

Audience

This document is intended for system administrators or application developers who are installing and configuring Oracle REST Data Services. This guide assumes you are familiar with web technologies, especially REST (Representational State Transfer), and have a general understanding of Windows and UNIX platforms.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information and resources relating to Oracle REST Data Services, see the following the Oracle Technology Network (OTN) site:

http://www.oracle.com/technetwork/developer-tools/rest-data-services/

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that is displayed on the screen, or text that you enter.

Third-Party License Information

Oracle REST Data Services contains third-party code. See the *Oracle Database Licensing Information* book for notices Oracle is required to provide.

Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the third-party software, and the terms contained in the following notices do not change those rights.

1

Getting Started with Oracle REST Data Services

This tutorial is designed to let you get started quickly developing RESTful services using Oracle REST Data Services.

Getting Started with RESTful Services

Before you perform the actions in this tutorial, note the following prerequisites and recommendations:

- Ensure that you have installed Oracle REST Data Services and configured it to connect to an Oracle database.
- Ensure that you have installed Oracle SQL Developer 4.1 or later in order to be able to edit RESTful services.
- It is strongly recommended that you install a browser extension that enables you to view JSON in the web browser. Recommended extensions:
    - For Google Chrome: JSON Formatter (`webstore_json_formatter`)
    - For Mozilla Firefox: JSON View (`addons_mozilla_org`)

The examples in this tutorial assume the following:

- Oracle REST Data Services has been installed and configured, and is running in standalone mode on the following server, port, and context path: `localhost:8080/ords/`
- Oracle REST Data Services is configured to connect as its default connection to an Oracle database listening on `localhost:1521`, and the database has a service name of `orcl`.

This tutorial covers the following major topics:

1. REST-Enable a Database Table
2. Creating a RESTful Service through the Connections Navigator
3. Creating a RESTful Service from a SQL Query
4. Protect Resources
5. Register an OAuth Client Application

REST-Enable a Database Table

To enable a table for REST access, follow these steps.

> **✎ Note:**
>
> It is recommended that you follow the steps as closely as possible, including using the specified names for schemas and database objects. After you have successfully completed the tutorial using this approach, feel free to try it again using other values if you wish.

1. Create a user `ordstest` with the following privileges or roles:

```
CREATE USER ordstest IDENTIFIED BY <password>;
GRANT "CONNECT" TO ordstest;
GRANT "RESOURCE" TO ordstest;
GRANT UNLIMITED TABLESPACE TO ordstest
```

2. Connect to the `ordstest` schema. In SQL Developer create a connection to the `ordstest` schema, connect to it, and open a SQL worksheet.

3. Create a database table. For example, enter the following in the SQL Worksheet to create an example table named EMP:

```
CREATE TABLE EMP (
  EMPNO NUMBER(4,0),
  ENAME VARCHAR2(10 BYTE),
  JOB VARCHAR2(9 BYTE),
  MGR NUMBER(4,0),
  HIREDATE DATE,
  SAL NUMBER(7,2),
  COMM NUMBER(7,2),
  DEPTNO NUMBER(2,0),
  CONSTRAINT PK_EMP PRIMARY KEY (EMPNO)
  );
```

4. Insert some sample data into the table. For example:

```
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7369,'SMITH','CLERK',7902,to_date('17-DEC-80','DD-MON-RR'),800,null,20);
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7499,'ALLEN','SALESMAN',7698,to_date('20-FEB-81','DD-MON-RR'),1600,300,30);
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7521,'WARD','SALESMAN',7698,to_date('22-FEB-81','DD-MON-RR'),1250,500,30);
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7566,'JONES','MANAGER',7839,to_date('02-APR-81','DD-MON-RR'),2975,null,20);
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7654,'MARTIN','SALESMAN',7698,to_date('28-SEP-81','DD-MON-RR'),1250,1400,30);
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7698,'BLAKE','MANAGER',7839,to_date('01-MAY-81','DD-MON-RR'),2850,null,30);
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7782,'CLARK','MANAGER',7839,to_date('09-JUN-81','DD-MON-RR'),2450,null,10);
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7788,'SCOTT','ANALYST',7566,to_date('19-APR-87','DD-MON-RR'),3000,null,20);
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7839,'KING','PRESIDENT',null,to_date('17-NOV-81','DD-MON-RR'),5000,null,10);
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7844,'TURNER','SALESMAN',7698,to_date('08-SEP-81','DD-MON-RR'),1500,0,30);
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7876,'ADAMS','CLERK',7788,to_date('23-MAY-87','DD-MON-RR'),1100,null,20);
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7900,'JAMES','CLERK',7698,to_date('03-DEC-81','DD-MON-RR'),950,null,30);
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
```

```
(7902,'FORD','ANALYST',7566,to_date('03-DEC-81','DD-MON-RR'),3000,null,20);
Insert into EMP (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values
(7934,'MILLER','CLERK',7782,to_date('23-JAN-82','DD-MON-RR'),1300,null,10);
commit;
```

5. Enable the schema of the EMP table for REST. In SQL Developer, right-click the `ordstest` connection, and select **REST Services > Enable RESTful Services** to display the following wizard page:

Figure 1-1   Enabling the Schema of the EMP Table for REST
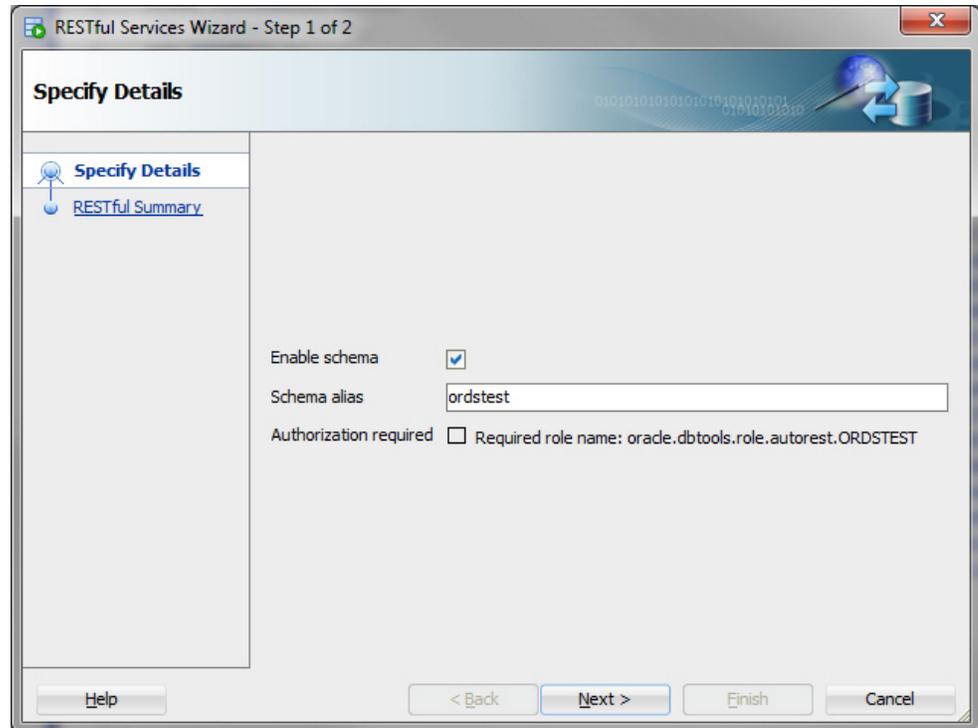
Enable schema: Enable this option.

Schema alias: Accept `ordstest` for the schema alias.

Authorization required: For simplicity, this tutorial does not require authorization, so disable this option.

Click **Next**.

6. On the RESTful Summary page of the wizard, click **Finish**.

7. Enable the EMP table. In SQL Developer, right-click EMP table in the Connections navigator, and select **REST Services > Enable RESTful Services** to display the following wizard page:

Figure 1-2   REST Enabling the EMP Table
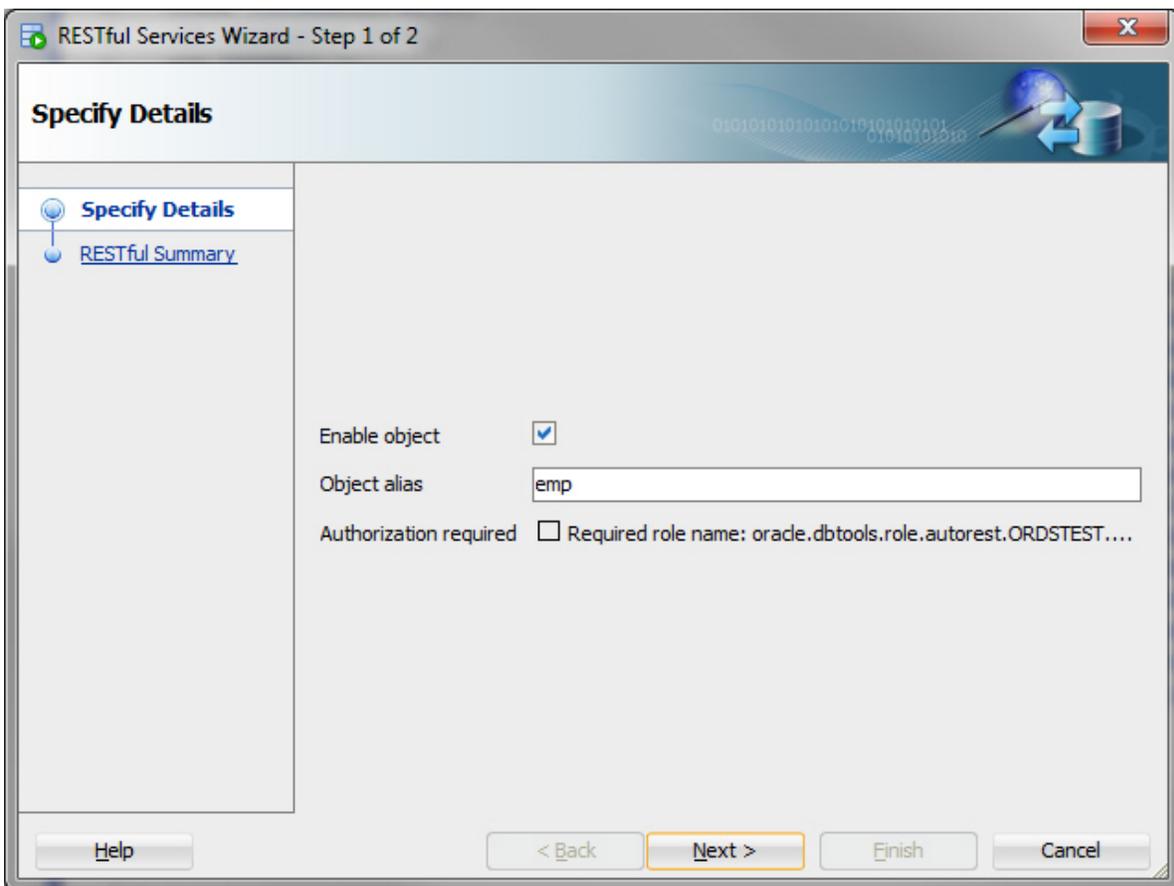
Enable object: Enable this option (that is, enable REST access for the EMP table).

Object alias: Accept `emp` for the object alias.

Authorization required: For simplicity, this tutorial does not require authorisation, so disable this option.

8. On the RESTful Summary page of the wizard, click **Finish**.

   The EMP table is now exposed as a REST HTTP endpoint .

   > **✎ Note:**
   >
   > DELETE, PUT, POST, and metadata-catalog endpoints are also auto-generated.

9. Test the REST endpoint. In a web browser, enter the URL `http://localhost:8080/ords/ordstest/emp/` as shown in the following figure:

   • The ORDSTEST schema has been exposed at the `/ordstest/` path.

   • The EMP table has been exposed at the `/emp/` path.

Figure 1-3   Testing the REST Enabled Table


```
{
    ▼ items: [
        ▼ {
            empno: 7369,
            ename: "SMITH",
            job: "CLERK",
            mgr: 7902,
            hiredate: "1980-12-16T18:30:00Z",
            sal: 800,
            comm: null,
            deptno: 20,
            ▼ links: [
                ▼ {
                    rel: "self",
                    href: http://localhost:8080/ords/ordstest/emp/7369
                }
            ]
        },
        ▼ {
            empno: 7499,
            ename: "ALLEN",
            job: "Director",
            mgr: 7698,
            hiredate: "1981-02-19T18:30:00Z",
            sal: 9999,
            comm: 300,
            deptno: 30,
            ▼ links: [
                ▼ {
                    rel: "self",
                    href: http://localhost:8080/ords/ordstest/emp/7499
                }
            ]
        },
```
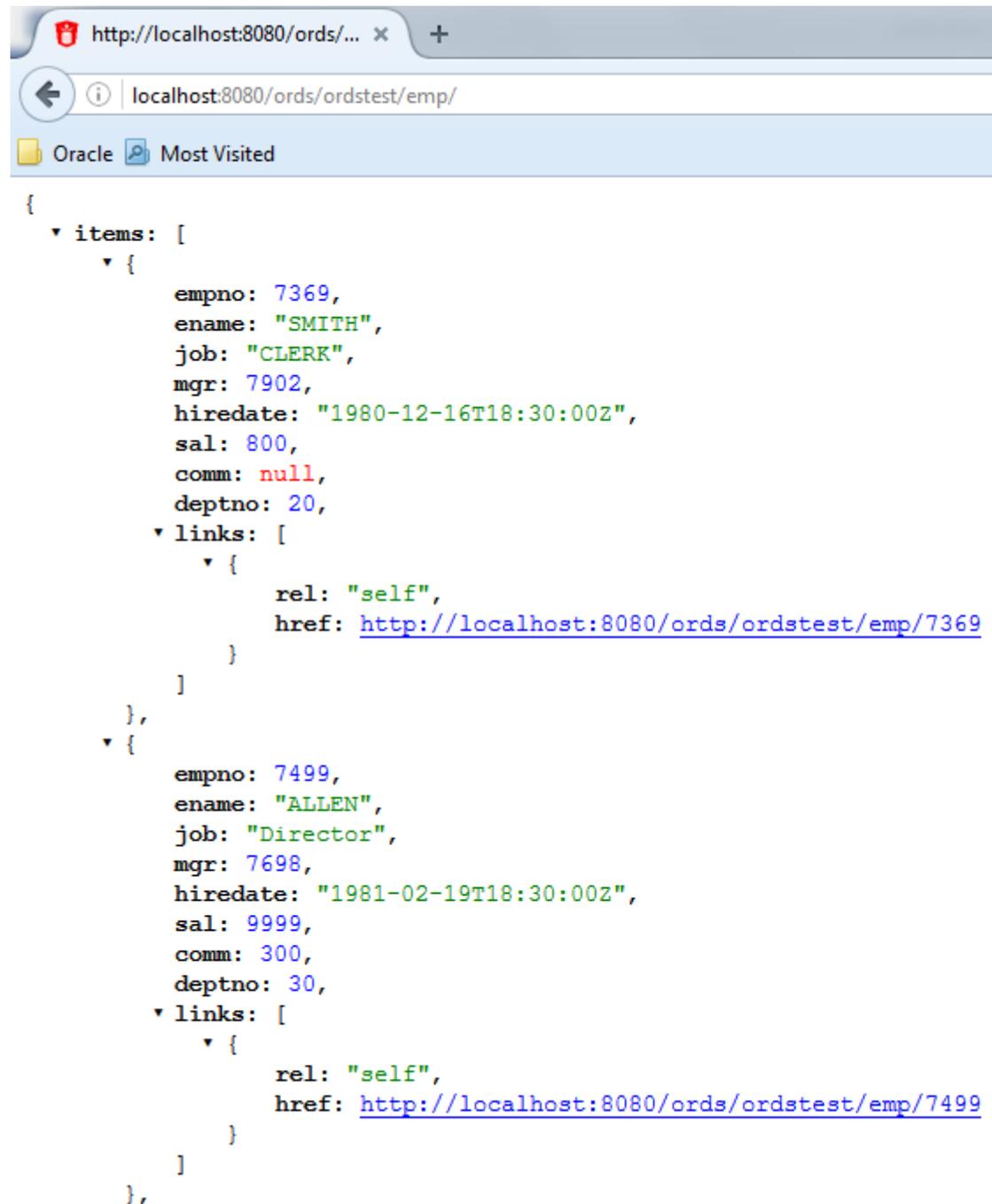
Related Topics

• Automatic Enabling of Schema Objects for REST Access (AutoREST)

Creating a RESTful Service through the Connections Navigator

This section explains how to create a RESTful service by using REST Data Services node in the Connections navigator. Oracle REST Data Services provides an option through the Connections navigator that enables you to create and edit RESTful service definitions.

To create and test a RESTful service by using REST Data Services node in the Connections navigator, follow these steps:

1. Under **ordstest** schema, select **REST Data Services.**

Figure 1-4   REST Data Services option under Connections Navigator
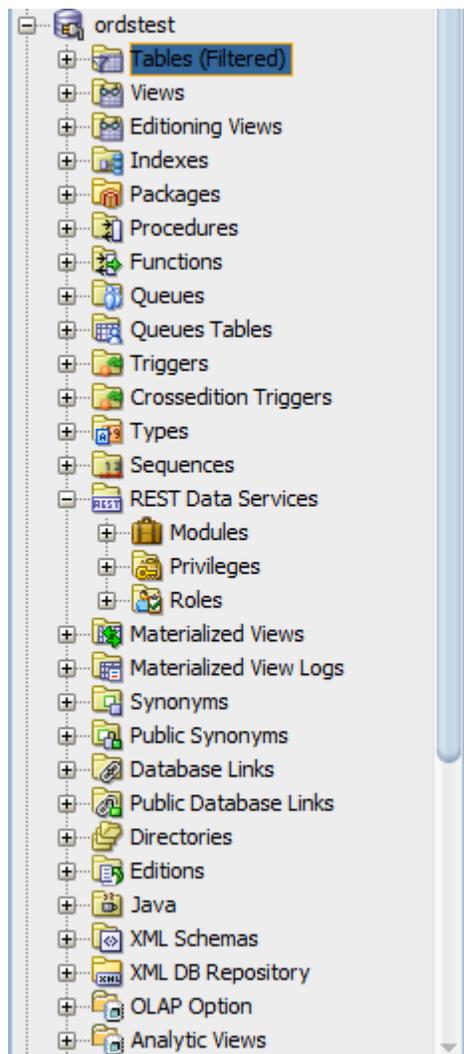
The following steps create and test the RESTful service.

2. Under **REST Data Services** node, right-click the `Modules` node, click **New Module**, and enter information on the Specify Module page:

Figure 1-5   Entering Information on the Specify Module Page

Module Name: Any desired name for the connection. For example, `demo`

URI Prefix: `/demo/`

Publish - Make this RESTful Service available for use: Enable (check).

Pagination Size: `25`

3. Click **Next**, and enter information on the Specify Template page:

Figure 1-6   Entering Information on the Specify Template Page
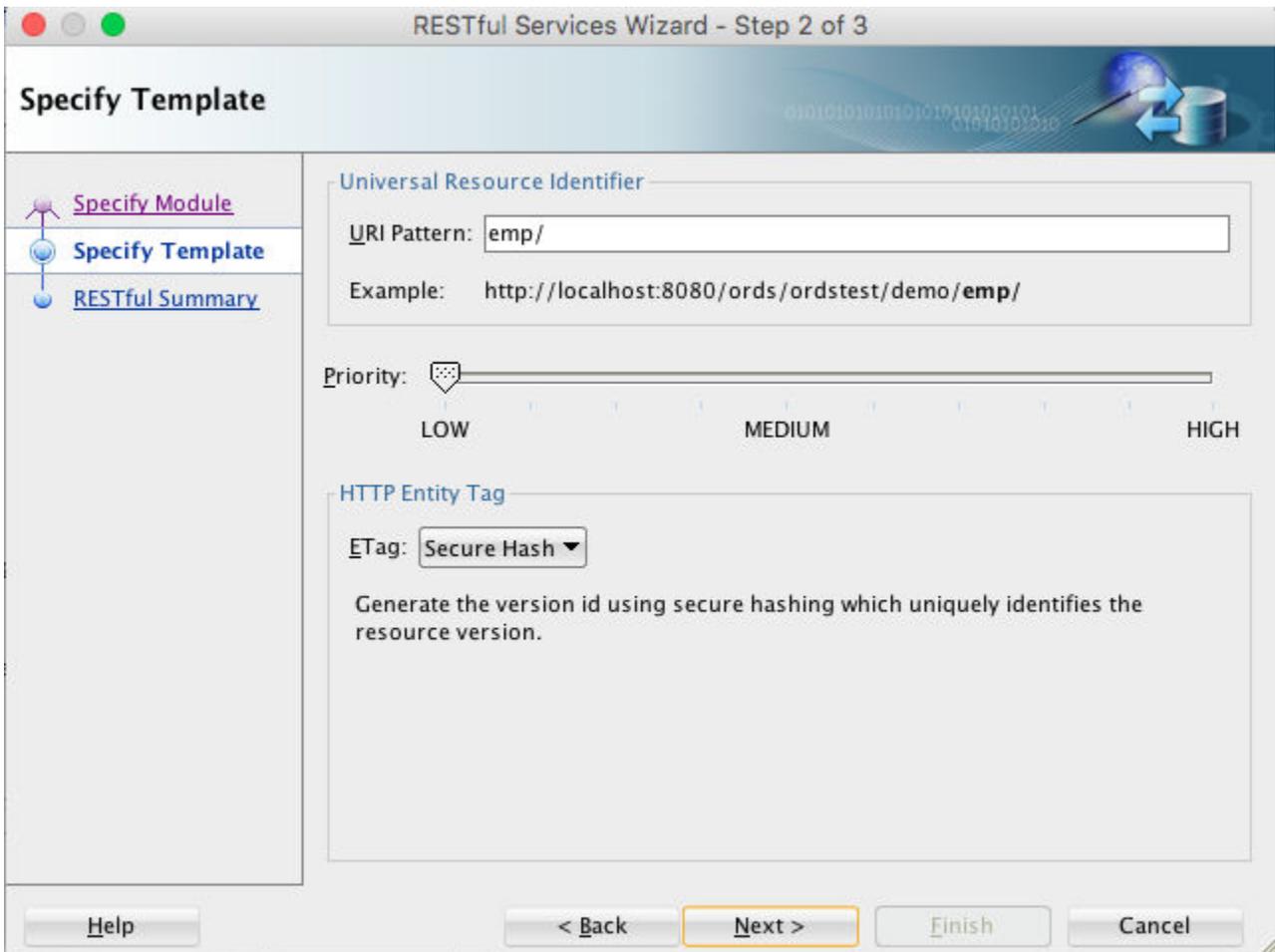
URI Pattern: `emp/`

Accept the defaults for the remaining options.

4. Click Next to go to the RESTful Summary page of the wizard, then click **Finish**. Expand the Modules node to display the resource module that you created.

5. Expand the module, Demo and right click on the `emp/` node, select **Add** handler and then select **GET** method.

6. Enter the information on the Create Resource Handler page.

Figure 1-7   Entering Information on Create Resource Handler Page:
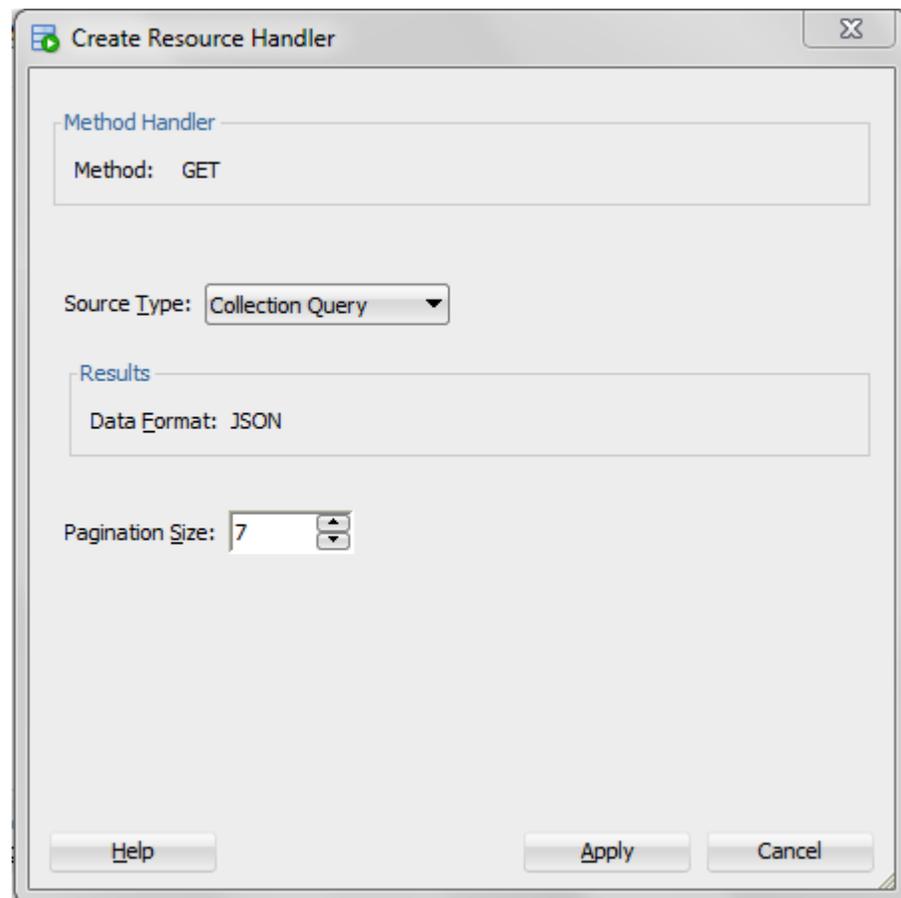
Source Type: `Collection Query`

Pagination Size: `7`

Click **Apply**.

Next step is to define the query for the GET resource handler.

7. In the SQL Worksheet, enter the following query:

```
SELECT
   INITCAP(ENAME) name,
  lower(job) job,
  TO_CHAR(sal,'9G999','NLS_NUMERIC_CHARACTERS=",."') salary,
   hiredate
FROM
emp
```

8. Click **Save REST Handler** icon. A confirmation message appears in the
    Messages - Log pane to confirm that the handler is saved to the database.

> **Note:**
>
> If you do not see the Messages - Log pane, go to the **View** menu and then select **Log**.

9. Test the RESTful service. In a web browser enter the URL http://localhost:8080/ords/ordstest/demo/emp/ as shown in the following figure:

   • The ORDSTEST schema has been exposed at the /ordstest/ path.

   • The query has been exposed at the /demo/emp/ path.

Figure 1-8   Testing URL in a Web Browser
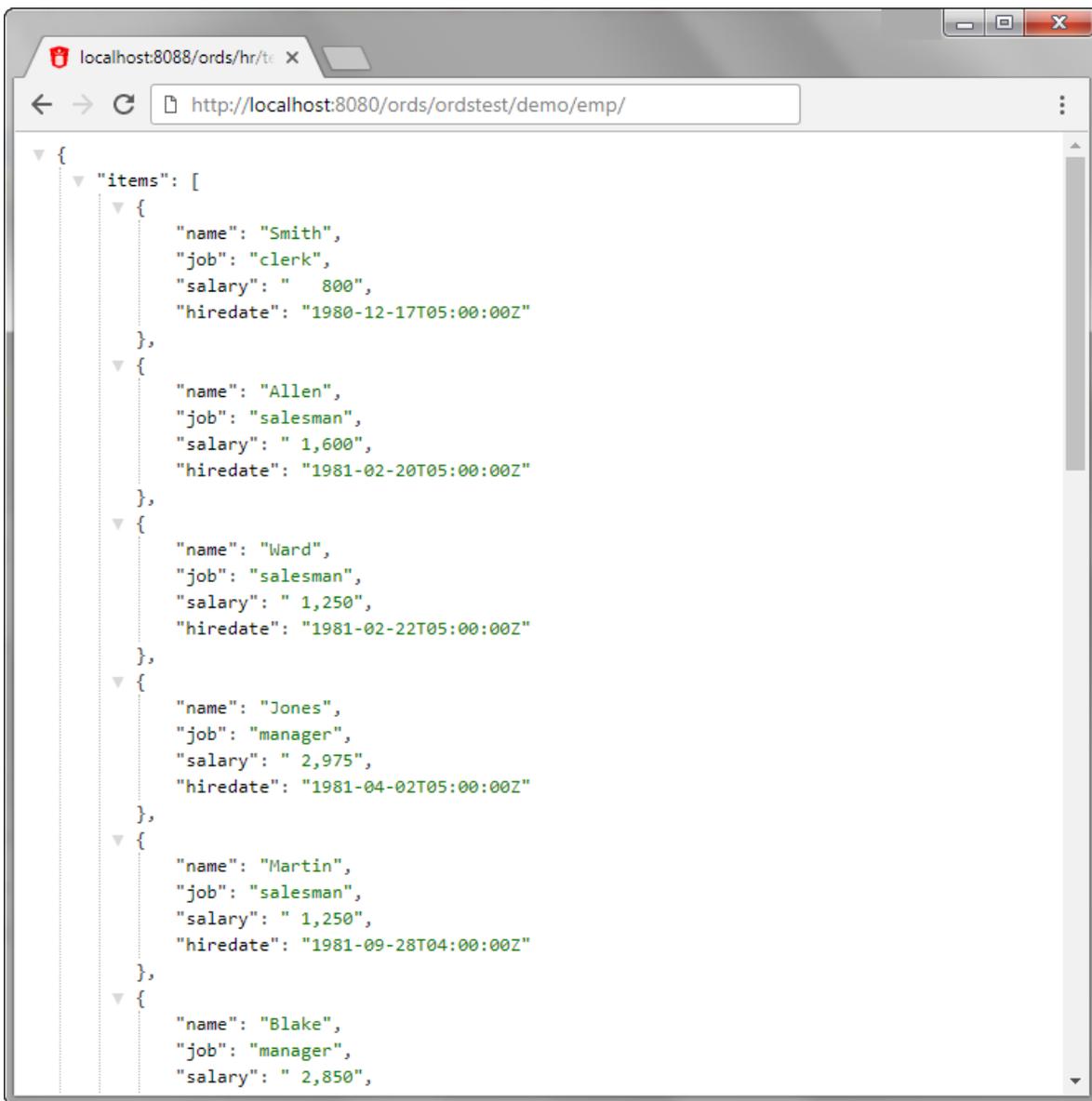
Related Topics

- [Creating a RESTful Service from a SQL Query](#)

Creating a Privilege under REST Data Services

Controlling access to protected resources is done by defining privileges. **Privileges** restrict access to only users having at least one of a set of specified roles. A privilege is then associated with one or more resource modules: before those resource modules can be accessed, the user must be authenticated and then authorized to ensure that the user has one of the required roles.

To protect resources, follow these steps.

1. Create a privilege. In SQL Developer, right-click the `Privileges` node under REST Data Services and select **New Privileges** to display the Create Privilege dialog box:

Figure 1-9   Create Privilege Dialog Box
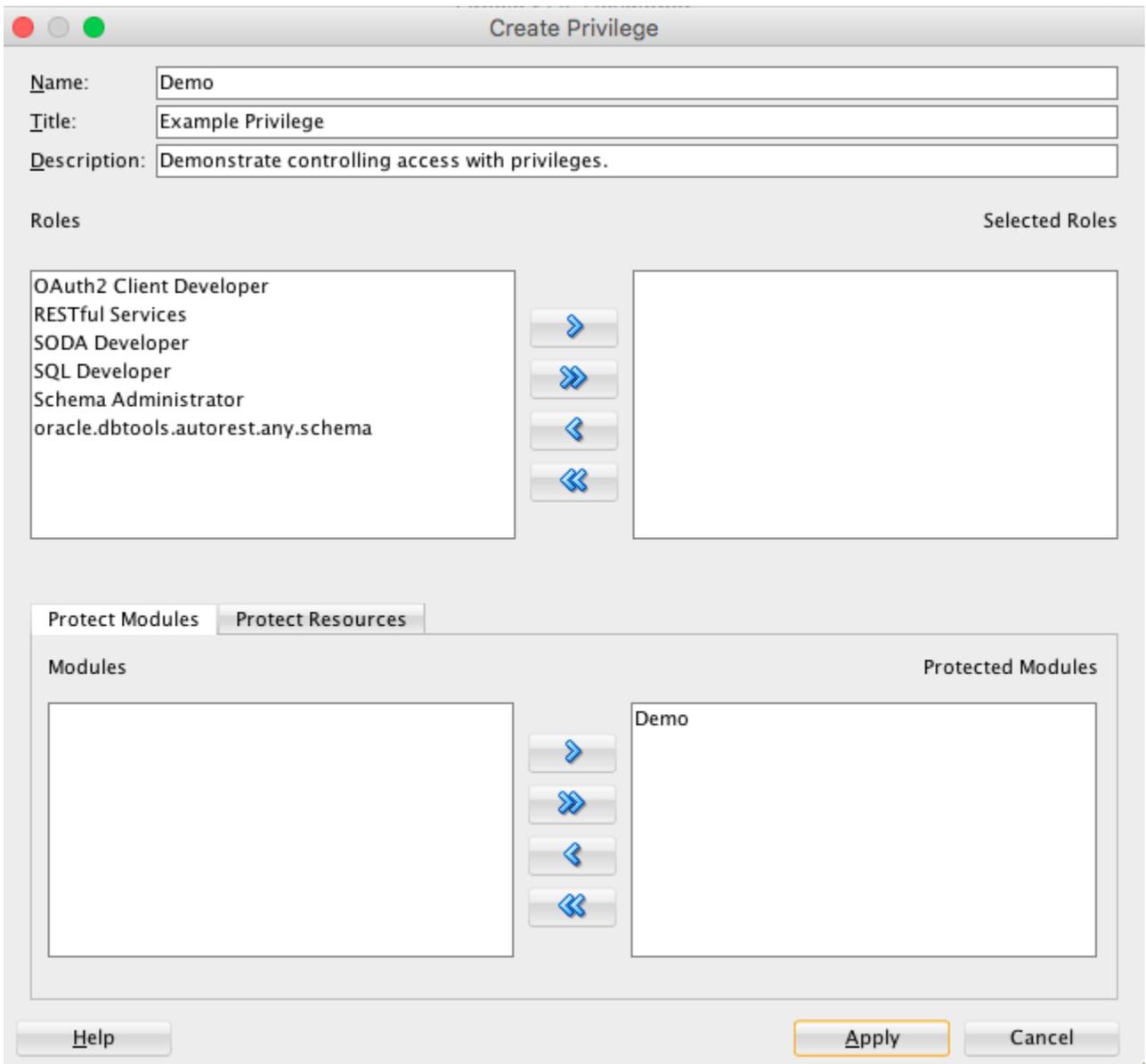
Name: Demo

Title: Example Privilege

Description: Demonstrate controlling access with privileges

Protected Modules: Ensure that the list includes the Demo module. Use the arrow button to move it if necessary.

Click **Apply**.

You have now created a privilege that protects the demo module. However, you have not restricted the privilege to any particular role; this will just require that the user be authenticated before accessing the demo module (the next step).

2. Test the RESTful service. In a web browser enter the following URL:

http://localhost:port/ords/ordstest/demo/emp/

3. Click the link to sign in, and enter the `test_developer` credentials.

> **Note:**
>
> To create a test_developer user refer to **Create a RESTful Service from a SQL Query** section.

A JSON document similar to the following is displayed:

Figure 1-10   JSON Document After Signing in

Creating a Role

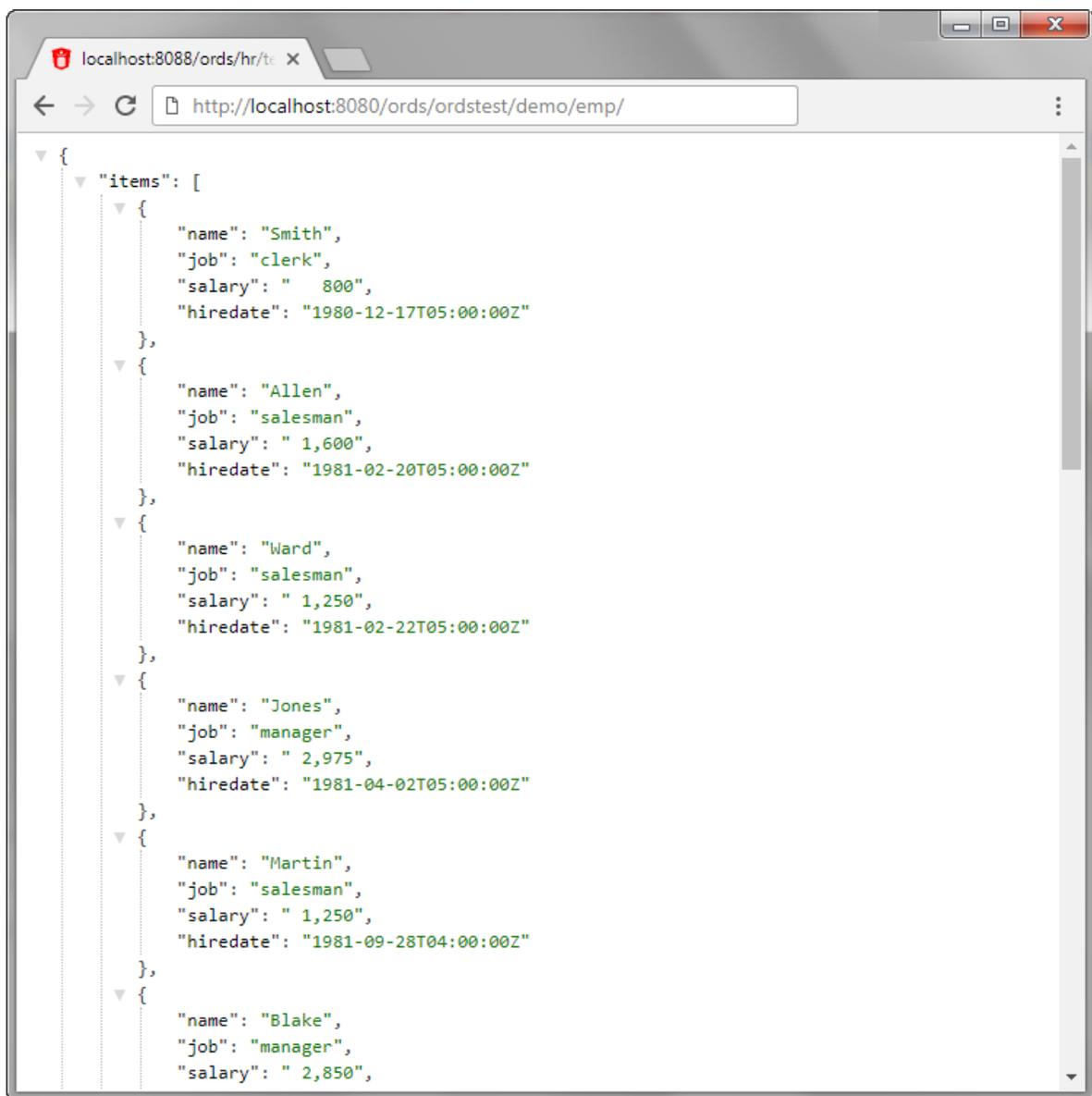This section explains how to create and delete a role.

To create a role, follow these steps:

1. Under **REST Data Services**, right click **Roles** and then click **New Role**.

2. In the **Create Role** dialog box, enter the name of the role you want to create.

Figure 1-11   Entering the New Role Name
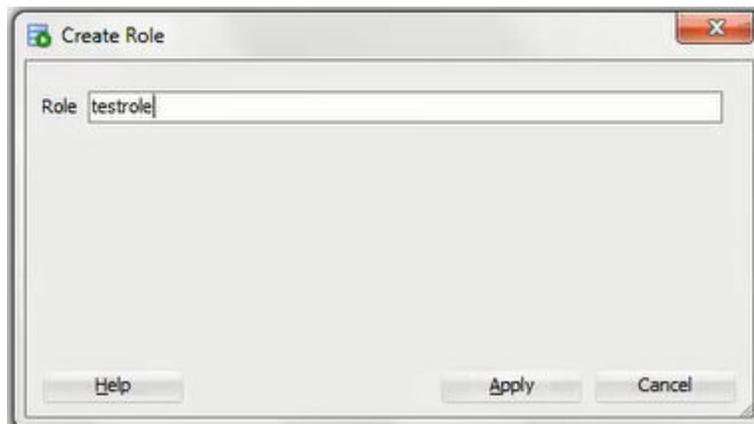
3. Click **Apply**, the new role is now created.

   To rename or delete a role, right click on the role name and choose one of the following options:

   • **Rename**: to change the role name.

   • **Delete**: to remove the role.

Creating a RESTful Service from a SQL Query

Oracle REST Data Services provides a REST API (called the Resource Modules API) that enables Oracle SQL Developer to create and edit RESTful service definitions. This option is available when you do not have direct access to the database. Access to the Resource Modules API is protected, a user with the correct role must be provisioned, and the created user's credentials must be used when accessing the API from SQL Developer.

To create a RESTful service from a SQL query, follow these steps.

1. In the folder where Oracle REST Data Services was installed, enter the following command at a command prompt:

   ```
   java -jar ords.war user test_developer "SQL Developer"
   ```

   • You will be prompted to enter a password.

   • This command creates a user named `test_developer` and grants the user the role named `SQL Developer`. Only users with the `SQL Developer` role are permitted to access the resource module's API.

- The user details are stored in a file named `credentials` in the ORDS configuration folder. However, it is not recommended to store user credentials in the credentials file in production deployments; instead, users should be provisioned in the host application server.

The remaining steps create and test the RESTful service.

2. Create RESTful connection. In SQL Developer, select **View > REST Data Services > Development**.

3. In the REST Development pane, right-click **REST Data Services > Connect**.

4. In the RESTful Services Connection dialog box, click the **+** (plus sign) icon to add a connection to the list available for selection.

5. In the New RESTful Services Connection dialog box, enter the necessary information:

Figure 1-12   Entering Information for New RESTful Services Connection
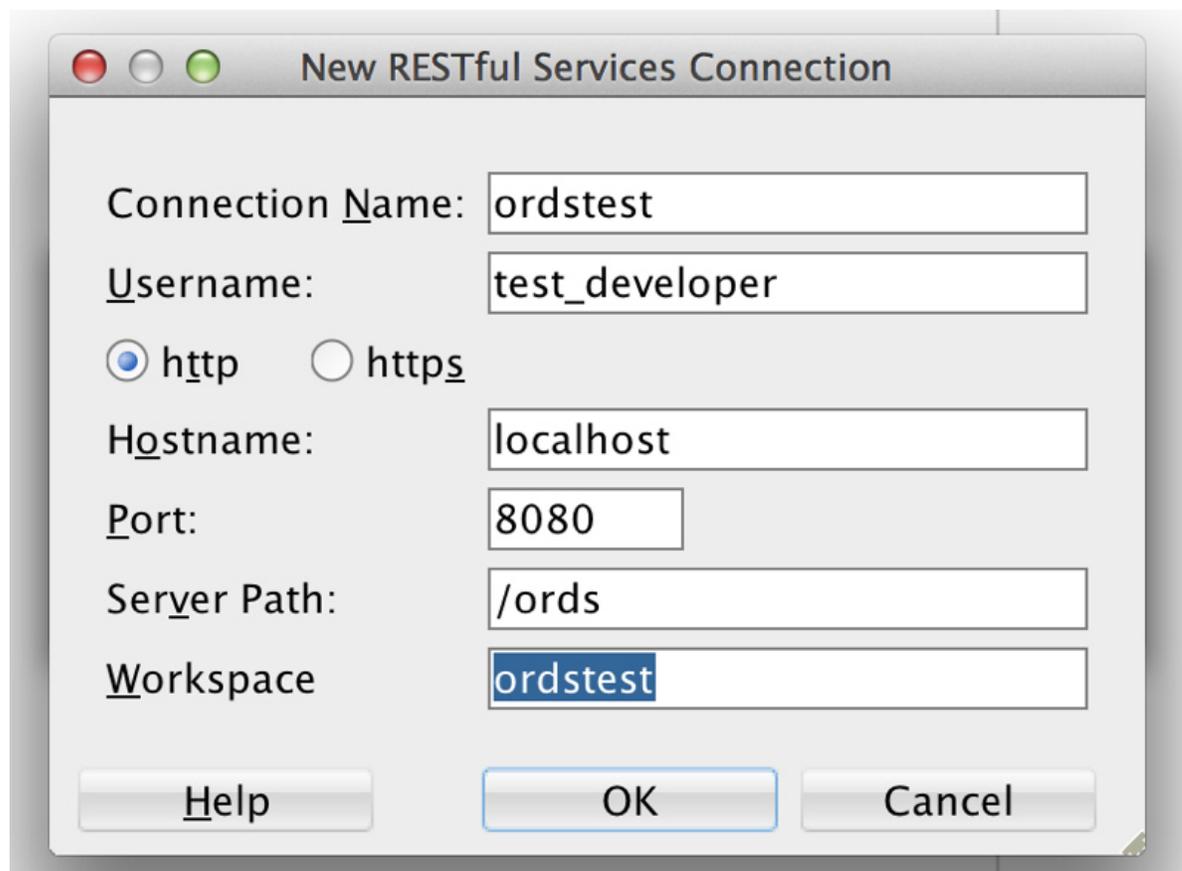
Connection Name: Any desired name for the connection. Example: `ordstest`

Username: `test_developer`

http or **https**: Select `http` for simplicity in this tutorial.

Hostname: `localhost`

Port: `8080`

Server Path: `/ords`

Workspace: `ordstest`

Click **OK**, then enter the password for the `test_developer` user at the prompt.

6. Create the module. Right-click the `Modules` node in the REST Development view, click **New Module**, and enter information on the Specify Module page:

Figure 1-13   Entering Information on the Specify Module Page
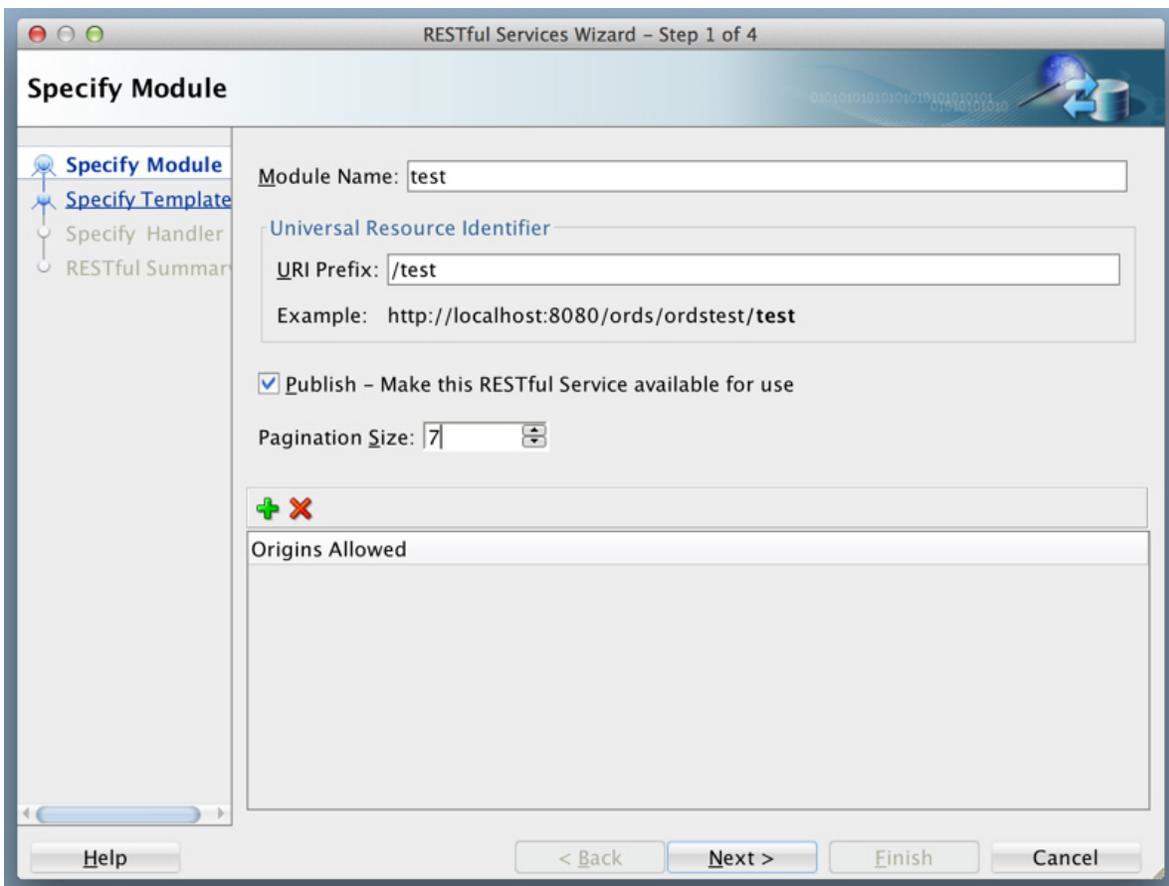
Module Name: Any desired name for the connection. Example: `test`

URI Prefix: `/test`

Publish - Make this RESTful Service available for use: Enable (check).

Pagination Size: `7`

7. Click **Next**, and enter information on the Specify Template page:

Figure 1-14   Entering Information on the Specify Template Page
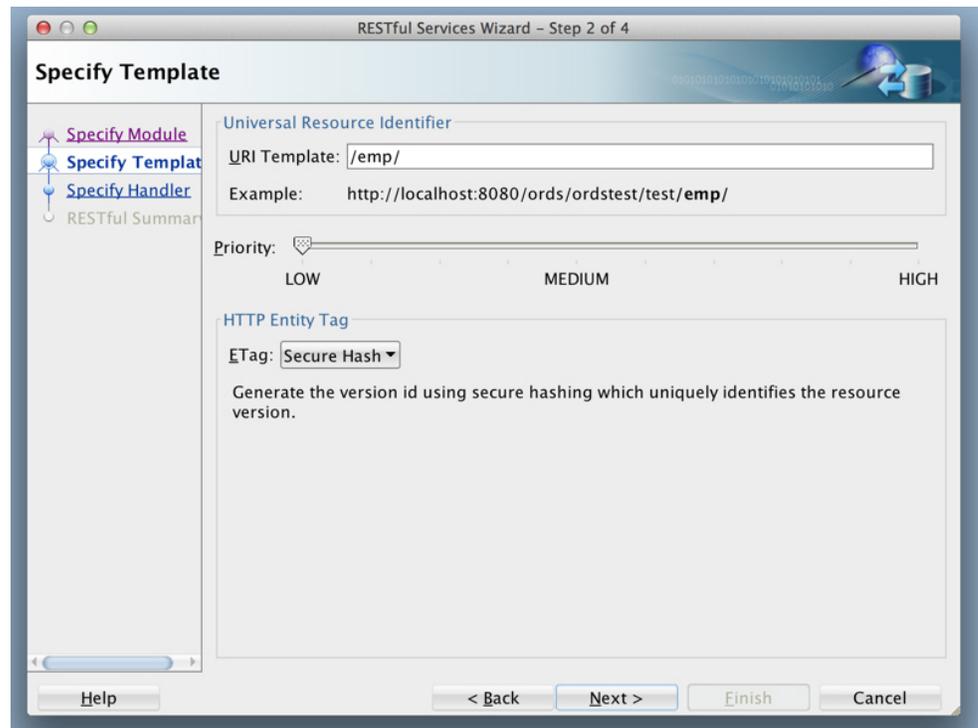
URI Template: /emp/

Accept the defaults for the remaining options.

8. Click **Next**, and enter information on the Specify Handler page:

Figure 1-15   Entering Information on the Specify Handler Page:
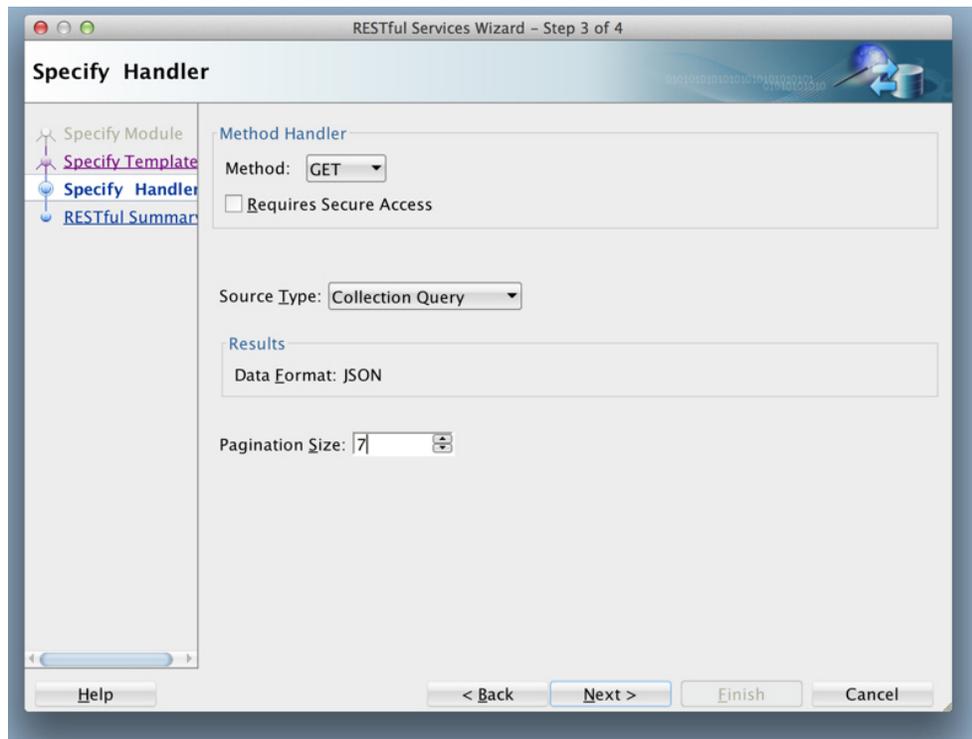
Method: GET

Requires Secure Access: Disable (uncheck) for this tutorial.

Source Type: Collection Query

Pagination Size: 7

9. Click Next to go to the RESTful Summary page of the wizard, then click **Finish**.

   The resource module is now created, the next step is to define the query for the GET resource handler.

10. Define the query for the GET resource handler.

   a. Expand the test node under the Modules node in the REST Development view.

   b. Expand the /emp/ node, right-click the GET node, and select **Open**.

   c. In the SQL Worksheet that opens for GET /emp/, enter the following SQL query:

   ```
   select * from emp
   ```

   d. Right-click on the test node under the 'Modules' node in the 'REST Development' view

   e. Click 'Upload...'. A confirmation dialog will appear confirming the module has been uploaded.

11. Test the RESTful service. In a web browser enter the URL http://localhost:8080/ords/ordstest/test/emp/ as shown in the following figure:

- The ORDSTEST schema has been exposed at the `/ordstest/` path.

- The query has been exposed at the `/test/emp/` path.

Figure 1-16   Testing the RESTful Service Created from a SQL Query
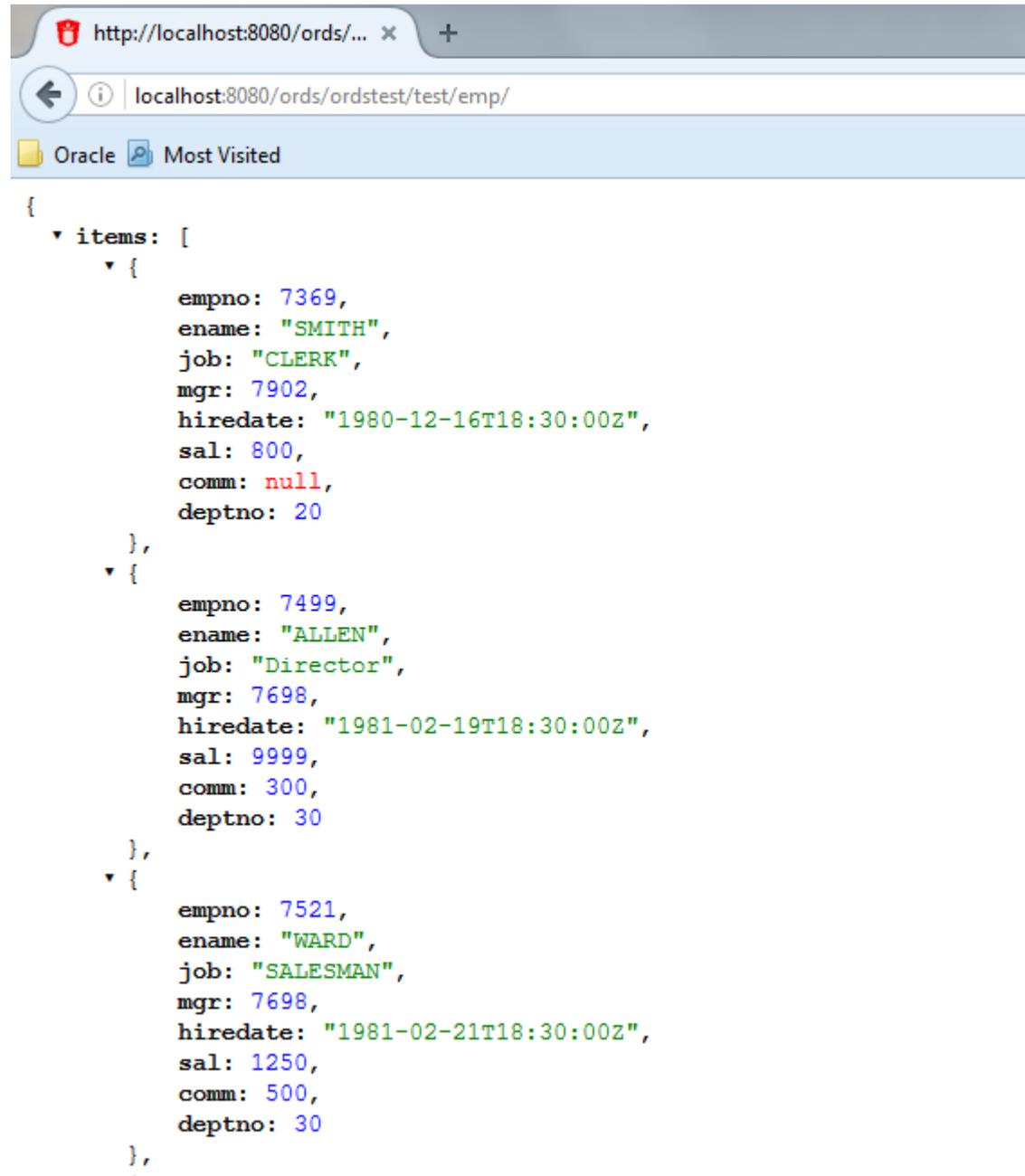
Related Topics

- Creating a RESTful Service through the Connections Navigator

Protect Resources

Up to this point the tutorial has deliberately disabled security on the RESTful endpoints you created, because it is easier to test them without security. In this topic you protect the `/test/emp/` service, requiring users to authenticate before accessing the service.

Controlling access to protected resources is done by defining privileges. **Privileges** restrict access to only users having at least one of a set of specified roles. A privilege is then associated with one or more resource modules: before those resource modules can be accessed, the user must be authenticated and then authorized to ensure that the user has one of the required roles.

To protect resources, follow these steps.

1. Create a privilege. In SQL Developer, right-click the `Privileges` node in the REST Development view and select **New Privileges** to display the Edit Privilege dialog box:

Figure 1-17   Edit Privilege Dialog Box
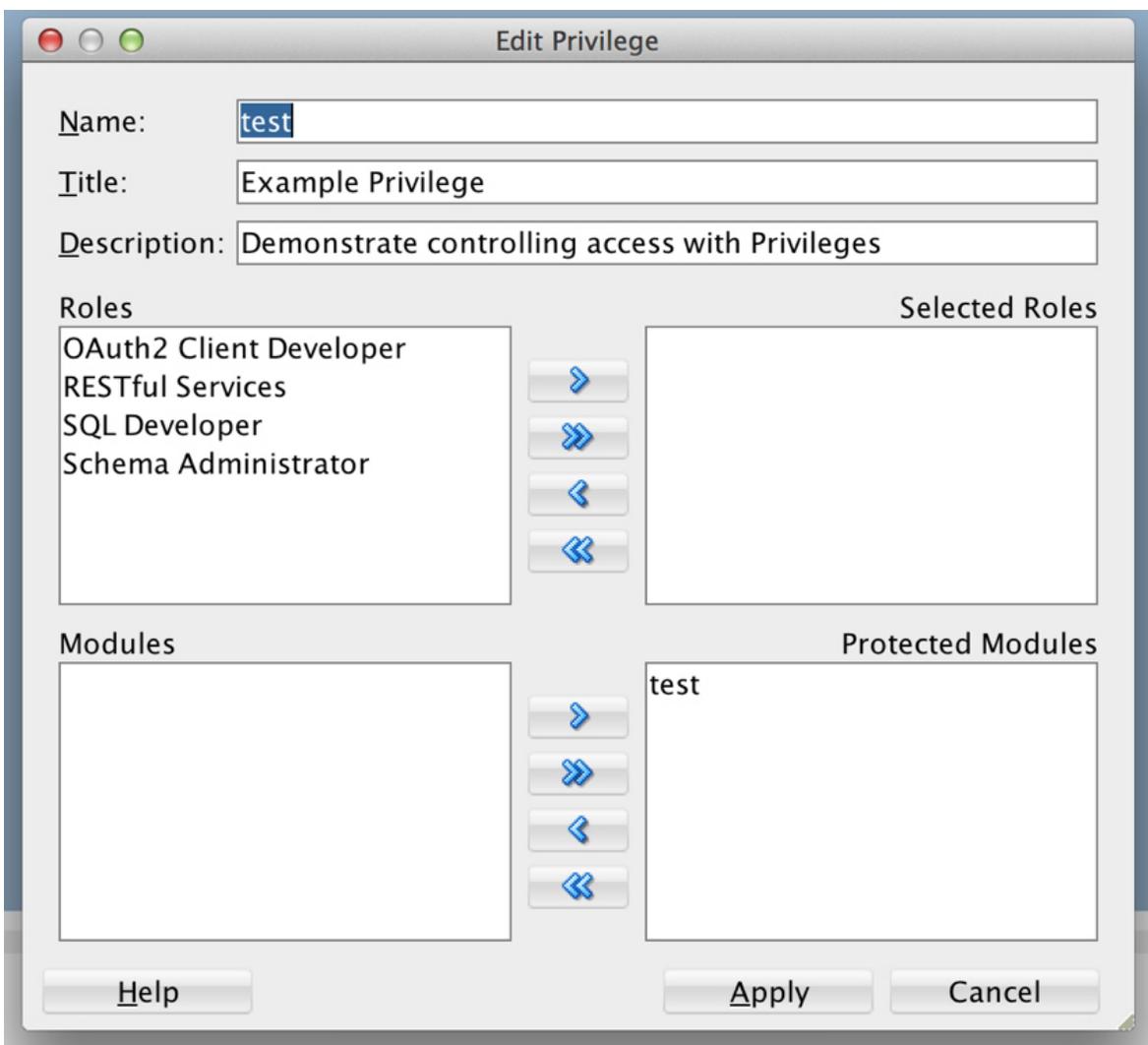
> **Name**: `test`
>
> **Title**: `Example Privilege`
>
> **Description**: `Demonstrate controlling access with privileges`
>
> **Protected Modules**: Ensure that the list includes the `test` module. Use the arrow button to move it if necessary.
>
> Click **Apply**.

2. Right click the `test` privilege and click **Upload**.

   A dialog box confirms that the privilege has been uploaded.

   You have now created a privilege that protects the test module. However, you have not restricted the privilege to any particular role; this will just require that the user be authenticated before accessing the test module (the next step).

3. Test the RESTful service. In a web browser enter the following URL:

   `http://localhost:8080/ords/ordstest/test/emp/`

4. Click the link to sign in, and enter the `test_developer` credentials.

   The contents of the JSON document are displayed.

Register an OAuth Client Application

This topic explains how to register your applications (called "third-party" applications here) to access a REST API.

OAuth 2.0 is a standard Internet protocol that provides a means for HTTP servers providing REST APIs to give limited access to third party applications *on behalf of* an end user.

- The author of the third-party application must register the application to gain client credentials.

- Using the client credentials the third party application starts a web flow that prompts the end-user to approve access.

So, before a third party application can access a REST API, it must be registered and the user must approve access. And before the application can be registered, it must be assigned a user identity that enables the user to register applications. Users possessing the `SQL Developer` role (such as the `test_developer` user created in Creating a RESTful Service from a SQL Query) are permitted to register OAuth clients.

> 💡 **Tip:**
>
> In a real application, you may want to provision specific users that can register OAuth clients; these users should be granted the `OAuth Client Developer` role.

This topic outlines how to complete these actions. It is not a full-featured demonstration of how to create and integrate a third party application; it just outlines the concepts involved.

1. Register the client application.

a. In a web browser enter the following URL:

```
http://localhost:8080/ords/ordstest/oauth/clients/
```

b. At the prompt, click the link to sign in and enter the credentials for the `test_developer` user.

c. Click **New Client** and enter the following information:

Name: `Test Client`

Description: `An example OAuth Client`

Redirect URI: `http://example.org/redirect`

Support e-mail: `info@example.org`

Support URI: `http://example.org/support`

Required Privileges: `Example Privilege`

d. Click **Create**.

The client registration is created, and the Authorization URI for the client is displayed. You have created a client that will use the Implicit Grant authorization flow (explained at https://tools.ietf.org/html/rfc6749#section-4.2).

Note the Client Identifier assigned to the client and the Authorization URI value. These values are used to start the authorization flow (next major step).

2. Approve the client application.

In a real third-party client application, the client will initiate the approval flow by directing a web browser to the Authorization URI. The end user will be prompted to sign in and approve access to the client application. The browser will be redirected back to the client's registered Redirect URI with a URI fragment containing the `access_token` for the approval. To simulate this process:

a. In a web browser, enter the Authorization URI that you noted in the previous step. The URL should look like the following (though you should not copy and paste in this example value):

```
http://localhost:8080/ords/ordstest/oauth/auth?
response_type=token&client_id=5B77A34A266EFB0056BE3497ED7099.&state=d5b7944-
d27d-8e2c-4d5c-fb80e1114490&_auth_=force
```

The `client_id` value must be the value of the client identifier assigned to the application. Be sure you are using the correct `client_id` value. Do not use the value in the preceding example; replace it with the client identifier assigned to your application.

The `state` value should be a unique, unguessable value that the client remembers, and can use later to confirm that the redirect received from Oracle REST Data Services is in response to this authorisation request. This value is used to prevent Cross Site Request Forgery attacks; it is very important, cannot be omitted, and must not be guessable or discoverable by an attacker.

b. At the prompt, click the link to sign in and enter the credentials for the `test_developer` user.

c. Review the access being requested, and click **Approve**.

The browser is redirected to a URL similar to the following:

```
http://example.org/redirect#token_type=bearer&access_token=-
i_Ows8j7JYu0p07jOFMEA..&expires_in=3600
```

When registering the OAuth client, you specified `http://example.org/`redirect as the Redirect URI. On completion of the approval request, the browser is redirected to this registered redirect URI. Appended to the URI is the information about the access token that was generated for the approval.

In a real application, the third party application would respond to the redirect to the redirect URI by caching the access token, redirecting to another page to show the user that they are now authorized to access the REST API, and including the access token in every subsequent request to the REST API. However, in this tutorial you just make note of the access token value and manually create a HTTP request with the access token included, as explained in the next major step.

The value of the access token (which in the preceding example is `-i_Ows8j7JYu0p07jOFMEA..`) will change on every approval.

Note that the access token expires. In the preceding example it expires after 3600 seconds (`&expires_in=3600`), that is, one hour.

3. Issue an authorized request.

After an access token has been acquired, the client application must remember the access token and include it with every request to the protected resource. The access token must be included in the HTTP Authorization request header (explained at `http://tools.ietf.org/html/rfc2616#section-14.8`) as in the following example:

```
Host: localhost:8080
GET /ords/ordstest/test/emp/
Authorization: Bearer -i_Ows8j7JYu0p07jOFMEA..
```

To emulate creating a valid HTTP request, use the cURL command line tool (if necessary, install cURL). In a real application this request would be performed by the client making an HTTP request, such as an `XMLHttpRequest`. For example:

```
curl -i -H'Authorization: Bearer -i_Ows8j7JYu0p07jOFMEA..' http://localhost:8080/
ords/ordstest/test/emp/
```

However, in this example replace `-i_Ows8j7JYu0p07jOFMEA..` with the access token value that you previously noted.

Output similar to the following JSON document should be displayed:

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked

{
 "items":[
  {"empno":7369,"ename":"SMITH","job":"CLERK","mgr":
7902,"hiredate":"1980-12-17T00:00:00Z","sal":800,"comm":null,"deptno":20},
  {"empno":7499,"ename":"ALLEN","job":"SALESMAN","mgr":
7698,"hiredate":"1981-02-20T00:00:00Z","sal":1600,"comm":300,"deptno":30},
  {"empno":7521,"ename":"WARD","job":"SALESMAN","mgr":
7698,"hiredate":"1981-02-22T00:00:00Z","sal":1250,"comm":500,"deptno":30},
  {"empno":7566,"ename":"JONES","job":"MANAGER","mgr":
7839,"hiredate":"1981-04-01T23:00:00Z","sal":2975,"comm":null,"deptno":20},
  {"empno":7654,"ename":"MARTIN","job":"SALESMAN","mgr":
7698,"hiredate":"1981-09-27T23:00:00Z","sal":1250,"comm":1400,"deptno":30},
```

```
  {"empno":7698,"ename":"BLAKE","job":"MANAGER","mgr":
7839,"hiredate":"1981-04-30T23:00:00Z","sal":2850,"comm":null,"deptno":30},
  {"empno":7782,"ename":"CLARK","job":"MANAGER","mgr":
7839,"hiredate":"1981-06-08T23:00:00Z","sal":2450,"comm":null,"deptno":10}
 ],
 "hasMore":true,
 "limit":7,
 "offset":0,
 "count":7,
 "links":[
  {"rel":"self","href":"http://localhost:8080/ords/ordstest/test/emp/"},
  {"rel":"describedby","href":"http://localhost:8080/metadata-catalog/test/
emp/"},
  {"rel":"first","href":"http://localhost:8080/ords/ordstest/test/emp/"},
  {"rel":"next","href":"http://localhost:8080/ords/ordstest/test/emp/?offset=7"}
 ]
}
```

However, if the `Authorization` header is omitted, then the status `401`
`Unauthorized` is returned instead.

> **See Also:**
>
> curl_haxx