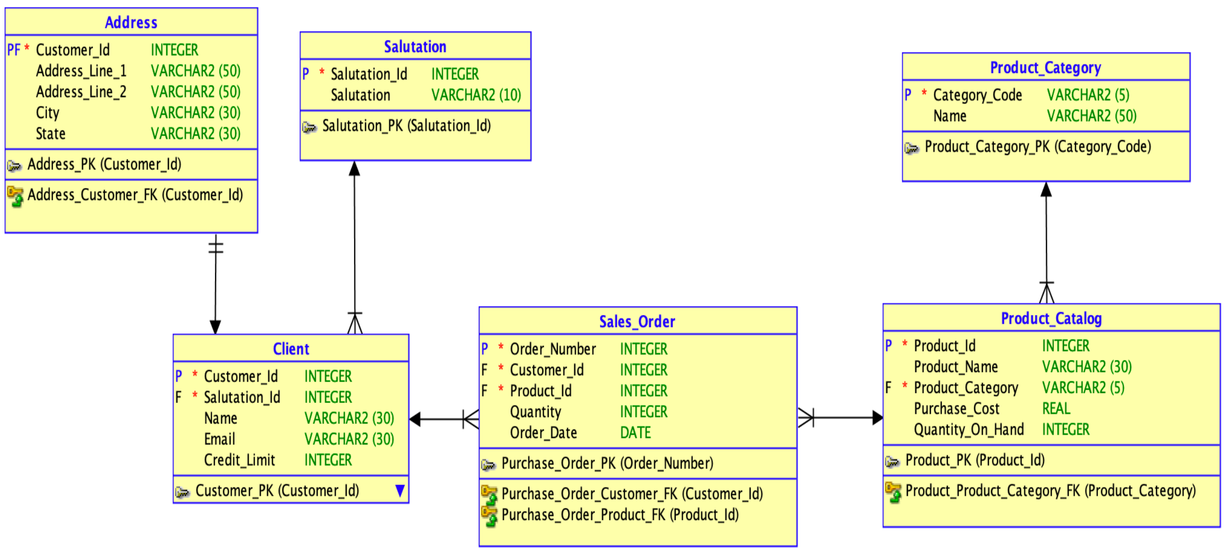
Table of Contents

Description of my database design:

My database is implemented in the Derby Database (Oracle Java DB) and has the following properties:


Address
PF * Customer_Id INTEGER
Address_Line_1 VARCHAR2 (50)
Address_Line_2 VARCHAR2 (50)
City VARCHAR2 (30)
State VARCHAR2 (30)
Address_PK (Customer_Id)
Address_Customer_FK (Customer_Id)
Salutation
P * Salutation_Id INTEGER
Salutation VARCHAR2 (10)
Salutation_PK (Salutation_Id)
Product_Category
P * Category_Code VARCHAR2 (5)
Name VARCHAR2 (50)
Product_Category_PK (Category_Code)
Client
P * Customer_Id INTEGER
F * Salutation_Id INTEGER
Name VARCHAR2 (30)
Email VARCHAR2 (30)
Credit_Limit INTEGER
Customer_PK (Customer_Id)
Sales_Order
P * Order_Number INTEGER
F * Customer_Id INTEGER
F * Product_Id INTEGER
Quantity INTEGER
Order_Date DATE
Purchase_Order_PK (Order_Number)
Purchase_Order_Customer_FK (Customer_Id)
Purchase_Order_Product_FK (Product_Id)
Product_Catalog
P * Product_Id INTEGER
Product_Name VARCHAR2 (30)
F * Product_Category VARCHAR2 (5)
Purchase_Cost REAL
Quantity_On_Hand INTEGER
Product_PK (Product_Id)
Product_Product_Category_FK (Product_Category)

Database Scripts: (For reference)

```
CREATE TABLE address (
   customer_id      INTEGER NOT NULL,
   address_line_1   VARCHAR(50),
   address_line_2   VARCHAR(50),
   city             VARCHAR(30),
   state            VARCHAR(30)
);

ALTER TABLE address ADD CONSTRAINT address_pk PRIMARY KEY ( customer_id );

CREATE TABLE client (
   customer_id     INTEGER NOT NULL,
   salutation_id   INTEGER NOT NULL,
   name            VARCHAR(30),
   email           VARCHAR(30),
   credit_limit    INTEGER
);

ALTER TABLE client ADD CONSTRAINT customer_pk PRIMARY KEY ( customer_id );

CREATE TABLE product_catalog (
   product_id         INTEGER NOT NULL,
   product_name       VARCHAR(30),
   product_category   VARCHAR(5) NOT NULL,
   purchase_cost      REAL,
   quantity_on_hand   INTEGER
);

ALTER TABLE product_catalog ADD CONSTRAINT product_pk PRIMARY KEY ( product_id );
```

```
CREATE TABLE product_category (
   category_code   VARCHAR(5) NOT NULL,
   name            VARCHAR(50)
);

ALTER TABLE product_category ADD CONSTRAINT product_category_pk PRIMARY KEY ( category_code );

CREATE TABLE sales_order (
   order_number   INTEGER NOT NULL,
   customer_id    INTEGER NOT NULL,
   product_id     INTEGER NOT NULL,
   quantity       INTEGER,
   order_date     DATE
);

ALTER TABLE sales_order ADD CONSTRAINT purchase_order_pk PRIMARY KEY ( order_number );

CREATE TABLE salutation (
   salutation_id   INTEGER NOT NULL,
   salutation      VARCHAR(10)
);

ALTER TABLE salutation ADD CONSTRAINT salutation_pk PRIMARY KEY ( salutation_id );

ALTER TABLE address
   ADD CONSTRAINT address_customer_fk FOREIGN KEY ( customer_id )
      REFERENCES client ( customer_id );

ALTER TABLE client
   ADD CONSTRAINT customer_salutation_fk FOREIGN KEY ( salutation_id )
      REFERENCES salutation ( salutation_id );

ALTER TABLE product_catalog
   ADD CONSTRAINT product_product_category_fk FOREIGN KEY ( product_category )
      REFERENCES product_category ( category_code );

ALTER TABLE sales_order
   ADD CONSTRAINT purchase_order_customer_fk FOREIGN KEY ( customer_id )
      REFERENCES client ( customer_id );

ALTER TABLE sales_order
   ADD CONSTRAINT purchase_order_product_fk FOREIGN KEY ( product_id )
      REFERENCES product_catalog ( product_id );
```

Implementation of One-to-One, One-to-Many and Many-to-many relationships:

1. Use at least one One-to-One relationship: The tables 'client' and 'address' are having One-to-One relationship, as one client can have only one official address and one address corresponds to only one client. It was implemented using the foreign key relation: Address_Customer_FK(Customer_ID) in address table.

2. Use at least one One-to-Many relationship:

- The tables 'Client' and 'Salutation' are having one to many relationship, as one client can have one salutation & one salutation can correspond to multiple clients. Eg: A salutation Mr. can be used for client name Reiner & Shubham & Gautham.

- The tables 'Product_Category' and 'Product_Catalog' are having one to many relationship, as one product category can have many product catalog, for eg. A product category called : software can have many product catalog: Facebook, Twitter etc.

3. Use at least one Many-to-Many relationship.

- The tables Client and Product_Catalog are having many to many relationships.

- For implementing many-to-many relationship we need to have a joining table (in this case: sales_order) . For example: The client Shubham can buy 2 products : Facebook and twitter. Another client Gautham can also buy the same products Facebook & Twitter. So the table sales_order will store the details like this:

Order_Number	Customer_Id	Product_Id	Quantity	Order_date
100	10	1000	8	24/04/2020
101	10	1001	9	25/04/2020
102	11	1000	8	26/04/2020
103	11	1001	7	25/04/2020

Justification that my DB is 3NF

- Implementation of First Normal Form: All the tables cells are containing only single values & each record is unique.

- Implementation of Second Normal Form: All the tables are having single column primary key. For example: Client table is having customer_id as primary key.

- Implementation of Third Normal Form: No tables are having transitive functional dependencies. Changing one Non-Key column won't affect any other Non-key column.

  Eg: Changing the non_key column, (Name of Client table) may change Salutation(Mr., Mrs.). So, I divided the table to Client and Salutation and added Salutation_Id to Client table.

Hence, our database is optimized to be in third normal form (3NF).

4. <u>Use at least one column of the following types: INTEGER, VARCHAR, DATE, REAL:</u>

- Integer: Customer_id (Client table), Salutation_id (Salutation table) etc.

- Varchar: Category Code (product_category table) & salutation (salutation table) etc

- Real: Purchase_cost (product_catalog table)

- Date: Order_date (sales_order table)

<u>2.1 Web Service Requirements</u>:

All database interactions are provided as web services. The web services implements all interactions with the database by using JDBC. My web service component has the following properties:

Short description of my web service design (REST)

For our project we have used REST API to provide the web services.

REST (Representational State Transfer) is truly a "web services" API. REST APIs are based on URIs (Uniform Resource Identifier, of which a URL is a specific type) and the HTTP protocol, and use JSON/XML for a data format, which is super browser-compatible.

List of all my web services with a short description

base URL: http://localhost:8080/RESTApp/

Sl. No.	Web Service	REST End Points	Media Type	Description
1	Create tables	/createTables	POST(text/plain	Creates all the tables.
2	Query that involves a table with a One-to-One relationship.	client/{id}, address/{id}	GET(application/xml)	This will get data from Client and Address tables
3	Query that involves a table with a One-to-Many relationship.	/salutation, /client	GET(application/xml)	This will get data from Client and Salutation tables
4	Query that involves a table with a Many-to-Many relationship.	/salesOrder/{id}	GET(application/xml)	This will get data from Sales_Order table
5	Adding entry to tables with One-to-One relationship	/client, /address	POST(application/xml)	This will add data to Address and Client tables.
6	Adding entry to tables with One-to-Many relationship	/salutation, /client	POST(application/xml)	This will add data to Salutation and Client tables.
7	Adding entry to tables with Many-to-Many relationships	/client, /productCatalog, /salesOrder	POST(application/xml)	This will add data to Product_Catalog and Client and Sales_Order tables.

Instructions on how to open the application in NetBeans 8.2

1. Unzip the ZIP file.
2. In NetBeans, Choose File > New Project (Ctrl-Shift-N on Windows/Cmd-Shift-N on OS X).
3. Choose Java Web > Web Application with Existing Sources. Click Next.
4. In the Name and Location page of the wizard, follow these steps:

- In the Location field, enter the folder that contains the web application's source root folders and web page folders (The extracted ZIP file)

5. Click Next to advance to the Server and Settings page of the wizard.
6. Click Next to advance to the Existing Sources and Libraries page of the wizard.
7. Verify all of the fields on the page, such as the values for the Web Pages Folder and Source Package Folders. (In WEB-INF content, browse to /RESTApp/web/WEB-INF)
8. Click Finish.
9. Optional: Click on Resolve the Hamcrest binaries Missing (in RESTApp) problem.

Instruction on how to run the application

Right click on the project, and press run, then a new browser window will be opened which acts as an Application Interface for the web-services.

On the application interface, please click on particular end-points as per the test cases shown below. Please change the ‘method to test’ and ‘Input’ accordingly as mentioned in below table:

Test Case No.	**Web Service**	**End-points**	**Method to test**	**Input**
1	Create tables	createTables	POST(text/plain	

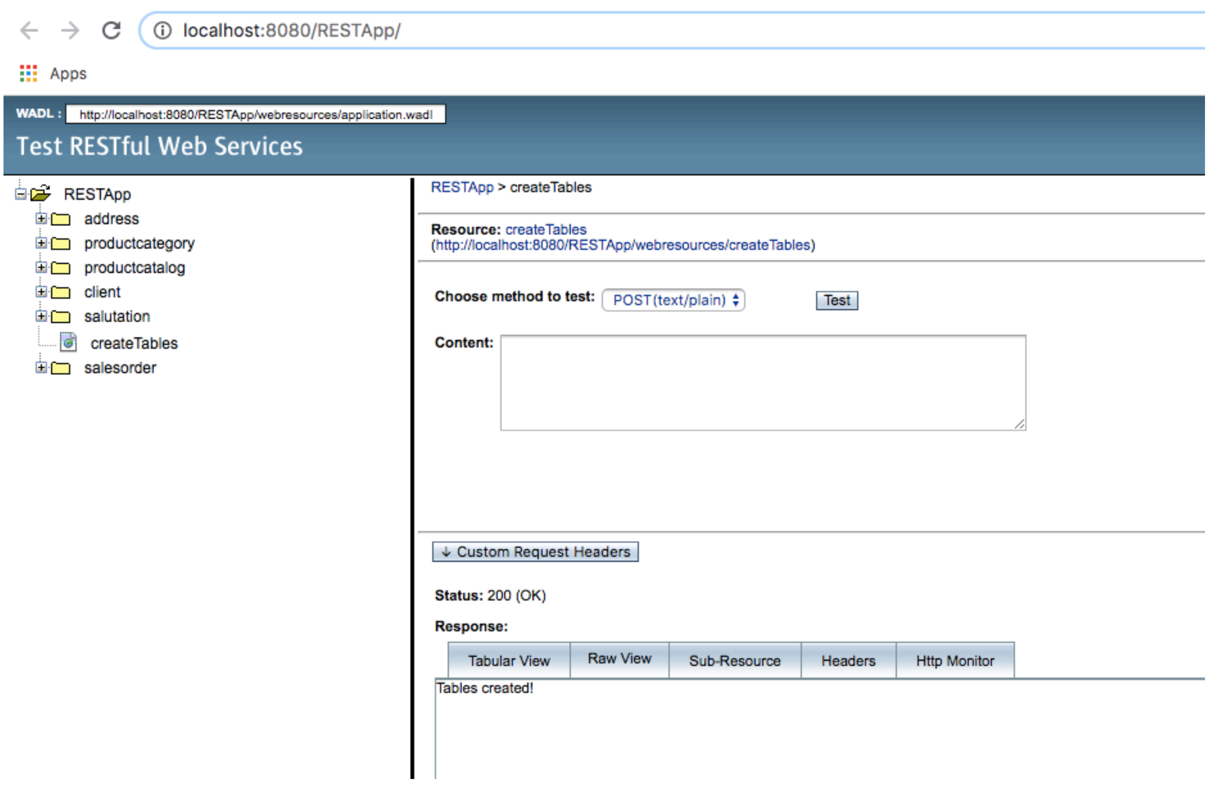
2.1	Adding entry to tables with One-to-Many relationship	salutation	POST(application/xml)	<salutation>
				  <salutation>Mr.</salutation>
				  <salutationId>1</salutationId>
				</salutation>

		client	POST(application/xml)	<client> <creditLimit>500</creditLimit> <customerId>1</customerId> <email>vinod@gmail.com</email> <name>Vinod</name> <salutationId> <salutationId>1</salutationId> </salutationId> </client>
2.2	Adding entry to tables with One-to-Many relationship	productCategory	POST(application/xml)	<productCategory> <categoryCode>SW</categoryCode> <name>software</name> </productCategory>
		productCatalog	POST(application/xml)	<productCatalog> <productCategory> <categoryCode>SW</categoryCode> </productCategory> <productId>1</productId> <productName>Identity Server</productName> <purchaseCost>1000.5</purchaseCost> <quantityOnHand>6</quantityOnHand> </productCatalog>
3	Adding entry to tables with One-to-One relationship	address	POST(application/xml)	<address> <addressLine1>Las Olivas Blvd</addressLine1> <addressLine2>Suite 51</addressLine2> <city>Lauderdale</city> <customerId>1</customerId> <state>KL</state> </address>
4	Adding entry to tables with Many-to-Many relationships	salesOrder	POST(application/xml)	<salesOrder> <customerId> <customerId>1</customerId> <salutationId> <salutationId>1</salutationId> </salutationId> </customerId> <orderDate>2020-04-24T00:00:00+05:30</orderDate> <orderNumber>1</orderNumber> <productId> <productCategory> <categoryCode>SW</categoryCode> </productCategory> <productId>1</productId> </productId> <quantity>5</quantity> </salesOrder>
5	Query that involves a table with a	client/{id}	GET(application/xml)	1
		address/{id}	GET(application/xml)	1

	One-to-One relationship.			
6	Query that involves a table with a One-to-Many relationship.	productcategory/{id}	GET(application/xml)	SW
		productcatalog/{id}	GET(application/xml)	1
7	Query that involves a table with a Many-to-Many relationship.	salesOrder/{id}	GET(application/xml)	1

Results of performing test-cases (respectively in serial order 1-7):

Test-Case 1:-



WADL : http://localhost:8080/RESTApp/webresources/application.wadl

Test RESTful Web Services

RESTApp > createTables

Resource: createTables (http://localhost:8080/RESTApp/webresources/createTables)

Choose method to test: POST(text/plain)

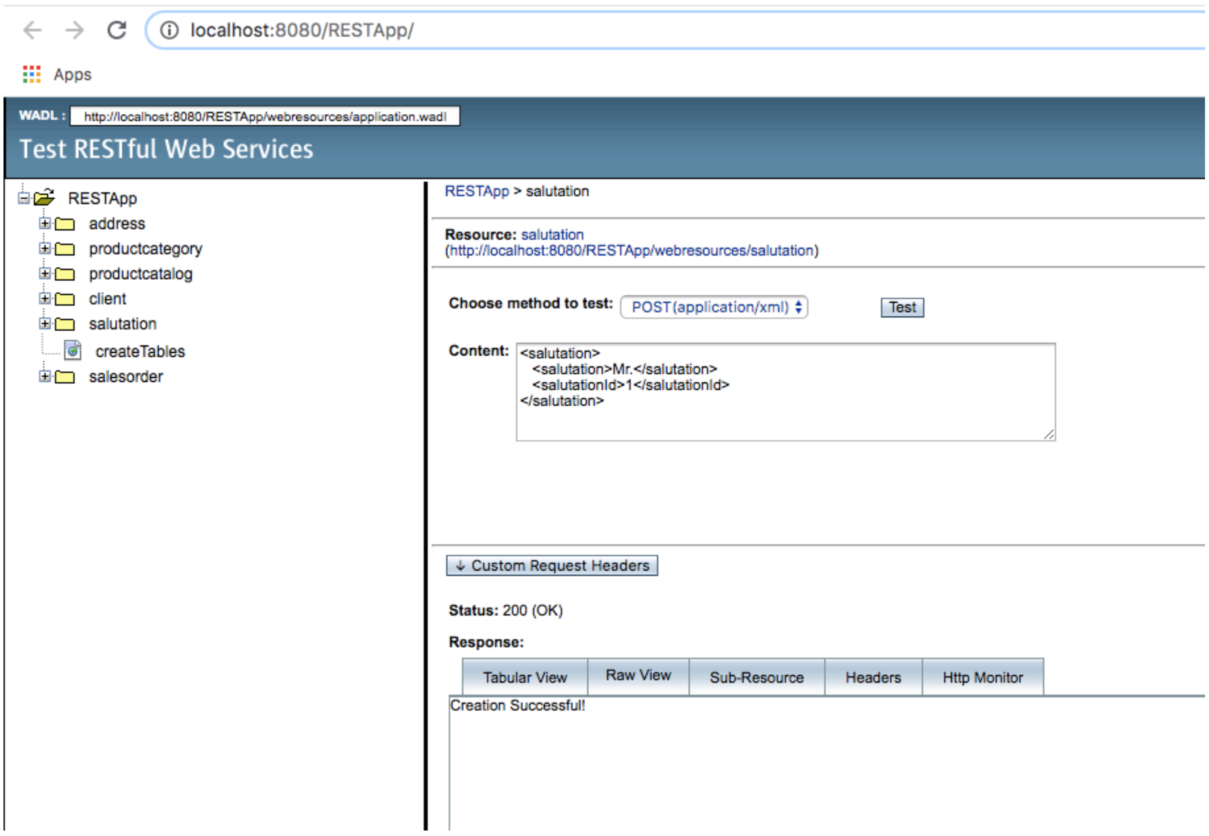
Content:

Status: 200 (OK)

Response:

```
Tables created!
```

Test-Case 2.1:-



localhost:8080/RESTApp/

WADL : http://localhost:8080/RESTApp/webresources/application.wadl

Test RESTful Web Services

- RESTApp
  - address
  - productcategory
  - productcatalog
  - client
  - salutation
  - createTables
  - salesorder

RESTApp > salutation

Resource: salutation (http://localhost:8080/RESTApp/webresources/salutation)

Choose method to test: POST(application/xml)

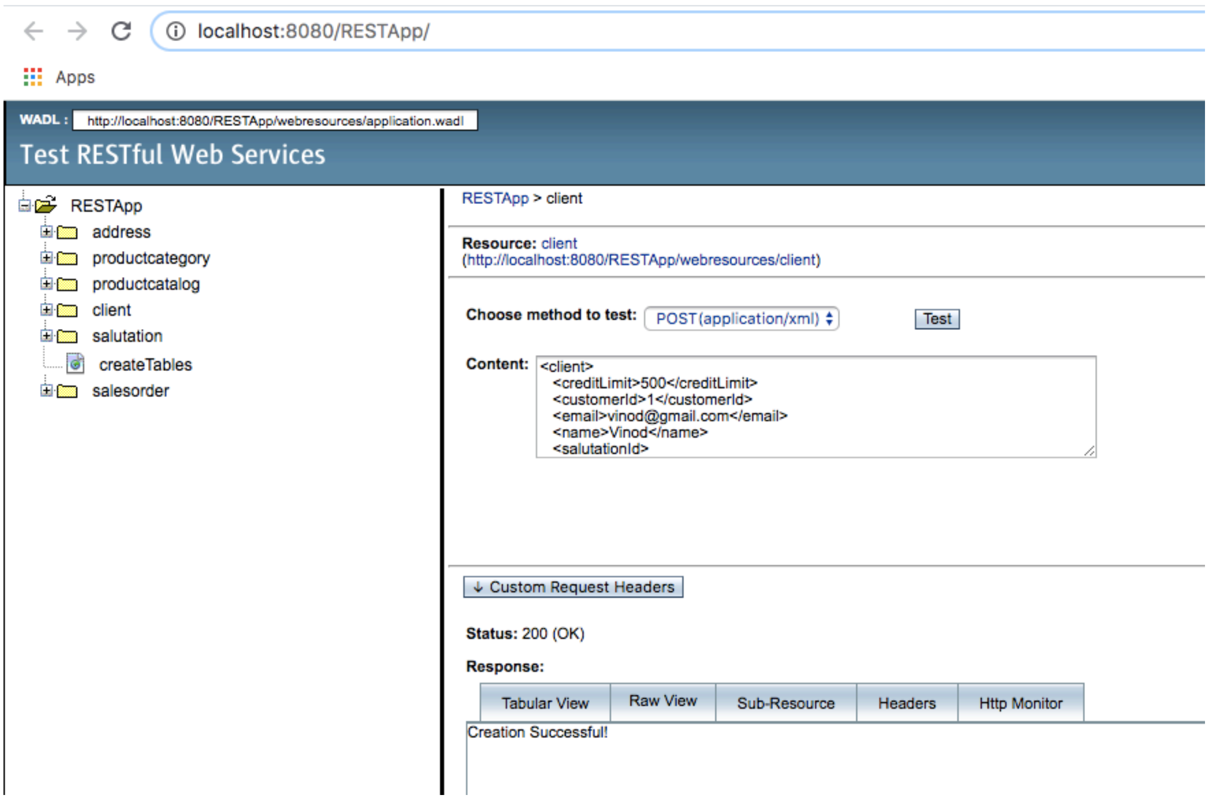
Content:

```
<salutation>
   <salutation>Mr.</salutation>
   <salutationId>1</salutationId>
</salutation>
```

Status: 200 (OK)

Response:

```
Creation Successful!
```



RESTApp > client

Resource: client (http://localhost:8080/RESTApp/webresources/client)

Choose method to test: POST(application/xml)

Content:

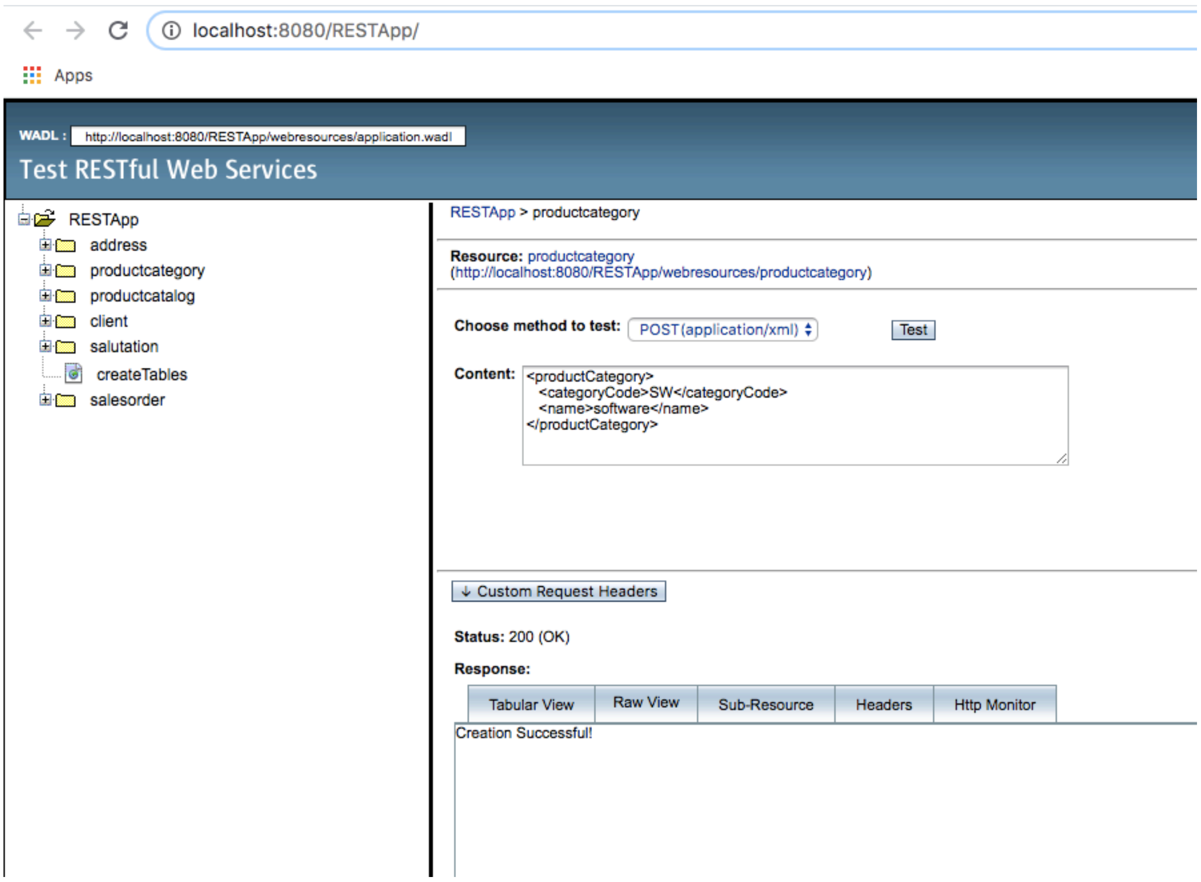
```
<client>
   <creditLimit>500</creditLimit>
   <customerId>1</customerId>
   <email>vinod@gmail.com</email>
   <name>Vinod</name>
   <salutationId>
```

Status: 200 (OK)

Response:

```
Creation Successful!
```

Test-Case 2.2:-



RESTApp > productcategory

Resource: productcategory (http://localhost:8080/RESTApp/webresources/productcategory)

Choose method to test: POST(application/xml)

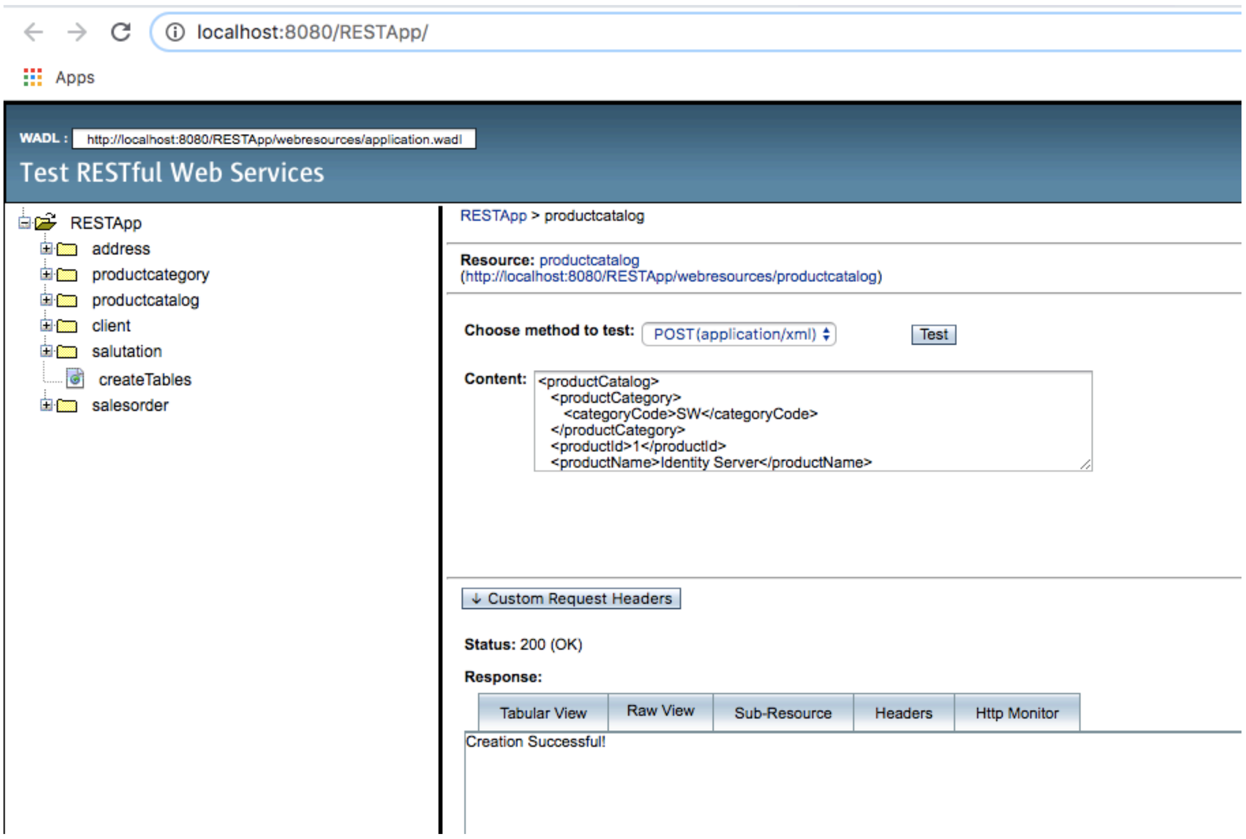
Content:

```
<productCategory>
   <categoryCode>SW</categoryCode>
   <name>software</name>
</productCategory>
```

Status: 200 (OK)

Response:

```
Creation Successful!
```



RESTApp > productcatalog

Resource: productcatalog
(http://localhost:8080/RESTApp/webresources/productcatalog)

Choose method to test: POST(application/xml) Test

Content:

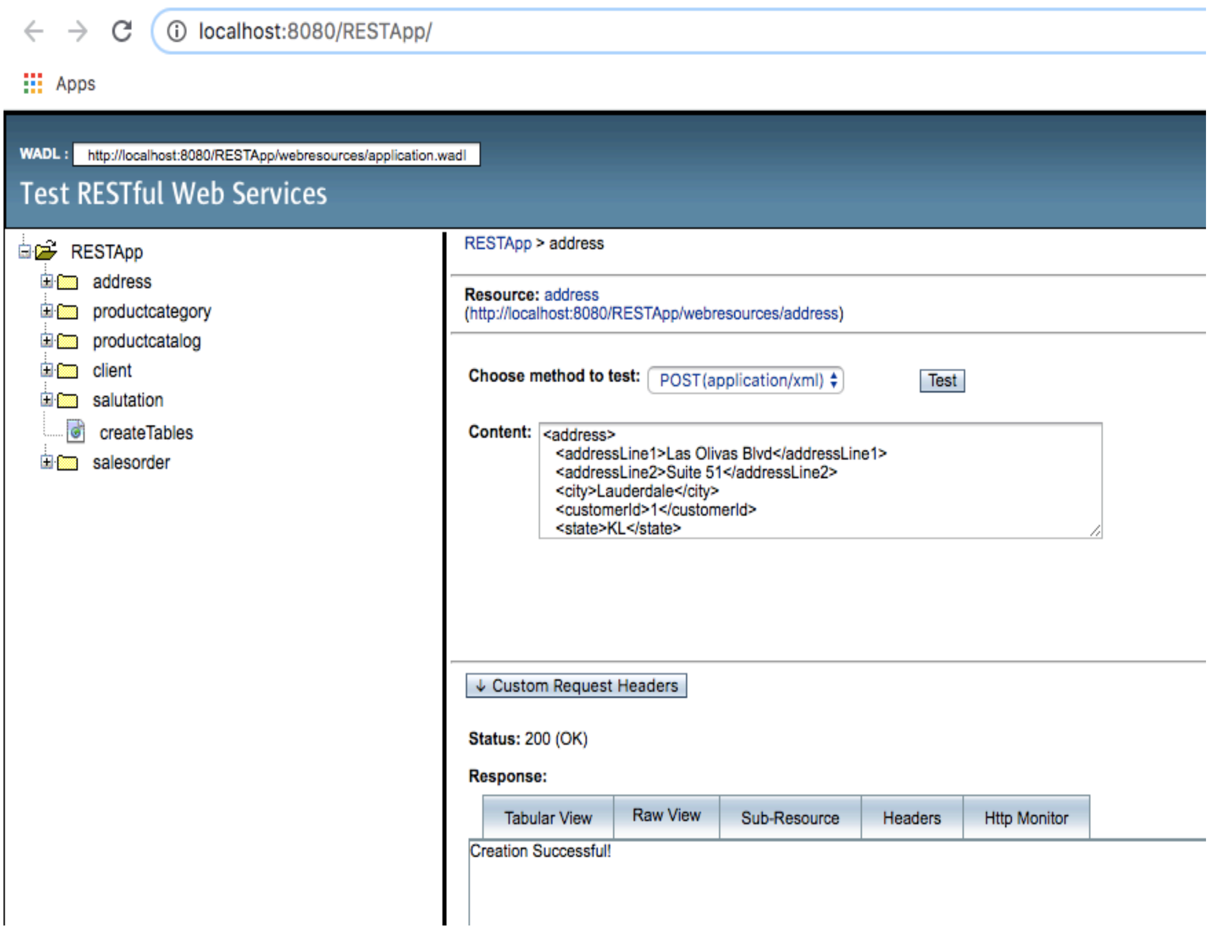
```
<productCatalog>
  <productCategory>
    <categoryCode>SW</categoryCode>
  </productCategory>
  <productId>1</productId>
  <productName>Identity Server</productName>
```

Status: 200 (OK)

Response:

```
Creation Successful!
```

Test-Case 3:-



Resource: address
(http://localhost:8080/RESTApp/webresources/address)

Choose method to test: POST(application/xml) Test

Content:

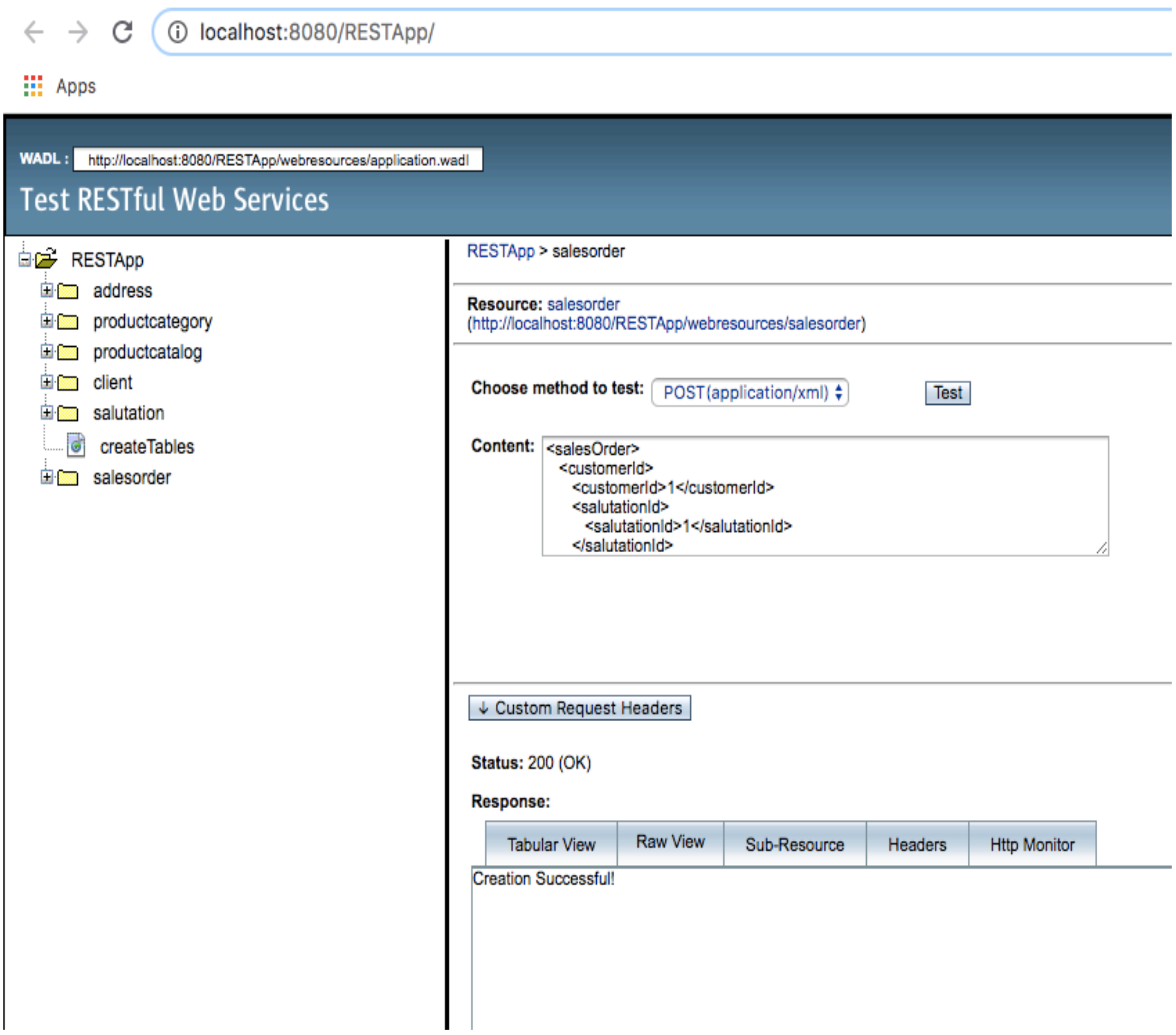
```
<address>
  <addressLine1>Las Olivas Blvd</addressLine1>
  <addressLine2>Suite 51</addressLine2>
  <city>Lauderdale</city>
  <customerId>1</customerId>
  <state>KL</state>
```

Status: 200 (OK)

Response:

```
Creation Successful!
```

Test-Case 4:-



Resource: salesorder (http://localhost:8080/RESTApp/webresources/salesorder)

Choose method to test: POST(application/xml) Test

Content:

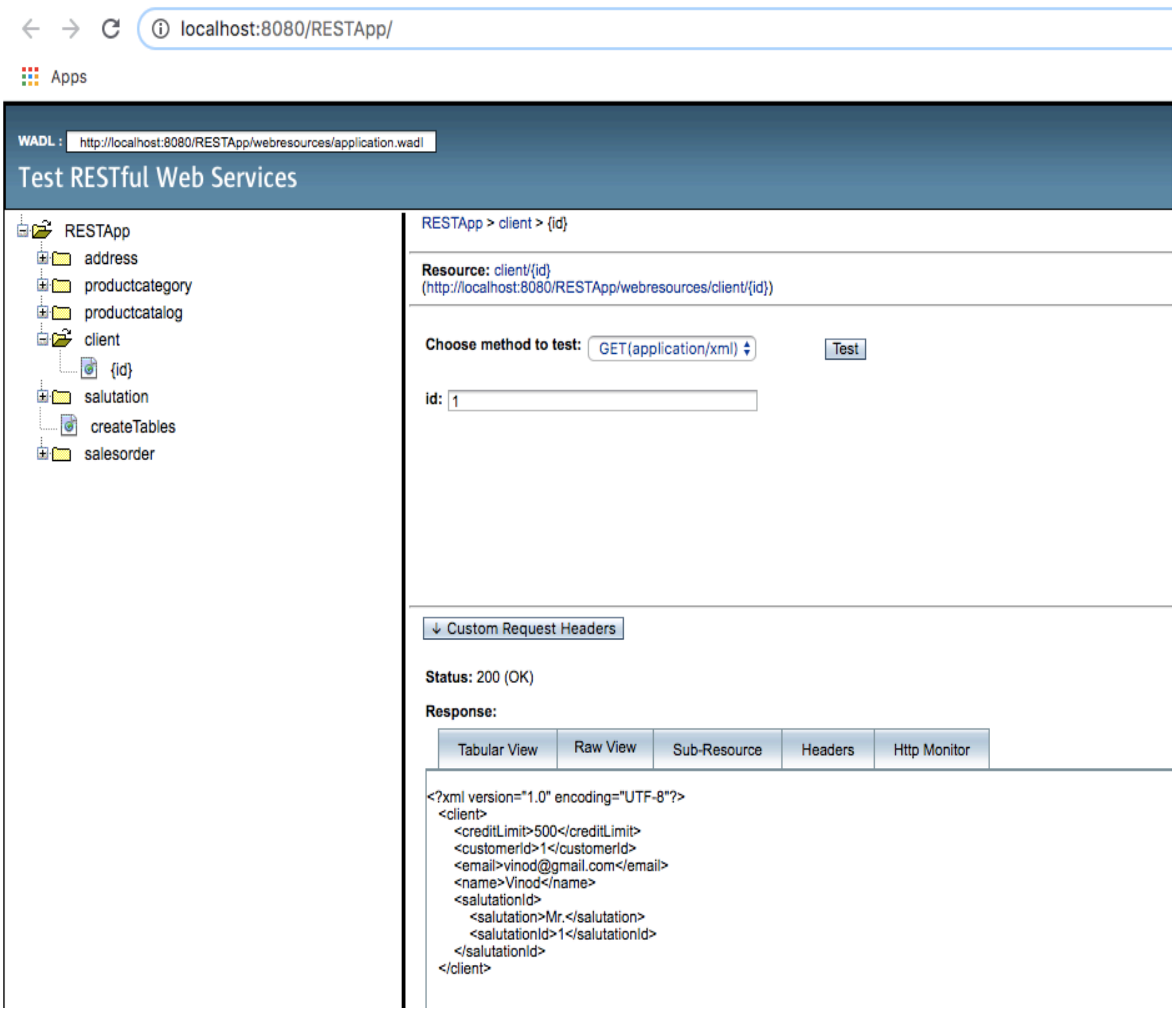
```
<salesOrder>
  <customerId>
    <customerId>1</customerId>
    <salutationId>
      <salutationId>1</salutationId>
    </salutationId>
```

Status: 200 (OK)

Response:

```
Creation Successful!
```

Test-case 5:



Resource: client/{id} (http://localhost:8080/RESTApp/webresources/client/{id})

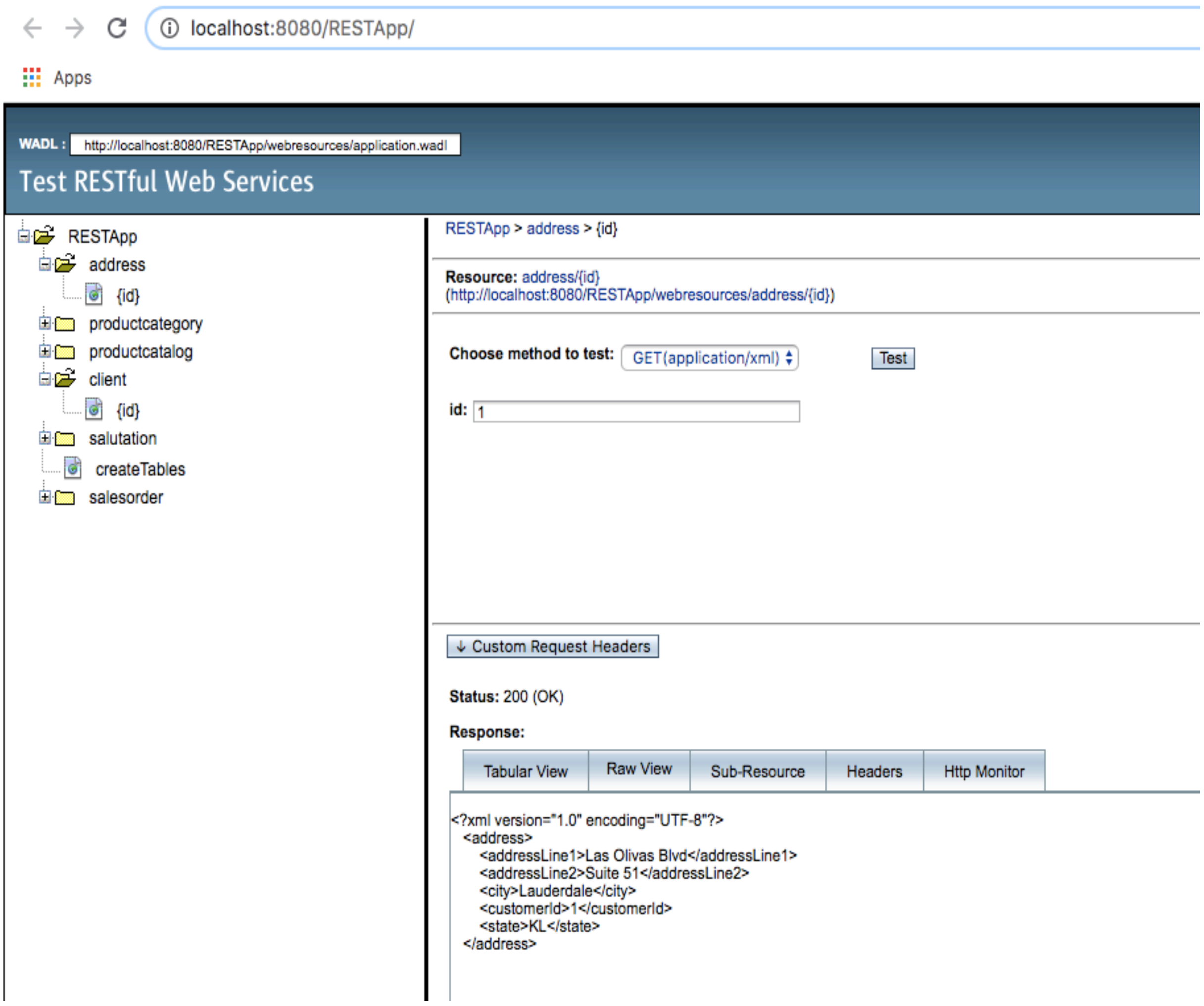
Choose method to test: GET(application/xml)

id: 1

Status: 200 (OK)

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <client>
     <creditLimit>500</creditLimit>
     <customerId>1</customerId>
     <email>vinod@gmail.com</email>
     <name>Vinod</name>
     <salutationId>
        <salutation>Mr.</salutation>
        <salutationId>1</salutationId>
     </salutationId>
  </client>
```



Resource: address/{id} (http://localhost:8080/RESTApp/webresources/address/{id})

Choose method to test: GET(application/xml) Test

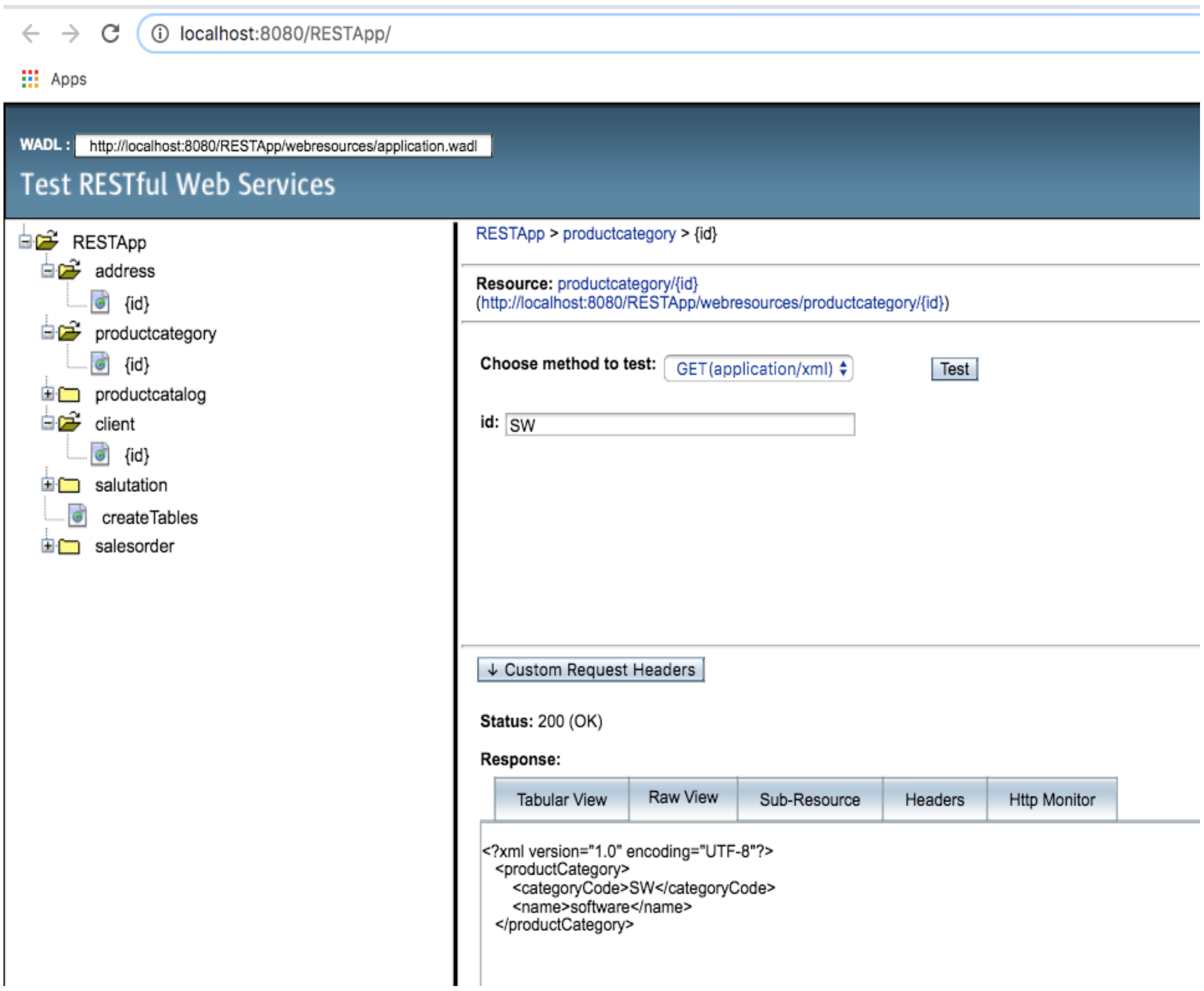
id: 1

Status: 200 (OK)

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
 <address>
   <addressLine1>Las Olivas Blvd</addressLine1>
   <addressLine2>Suite 51</addressLine2>
   <city>Lauderdale</city>
   <customerId>1</customerId>
   <state>KL</state>
 </address>
```

Test-case 6:



Resource: productcategory/{id} (http://localhost:8080/RESTApp/webresources/productcategory/{id})

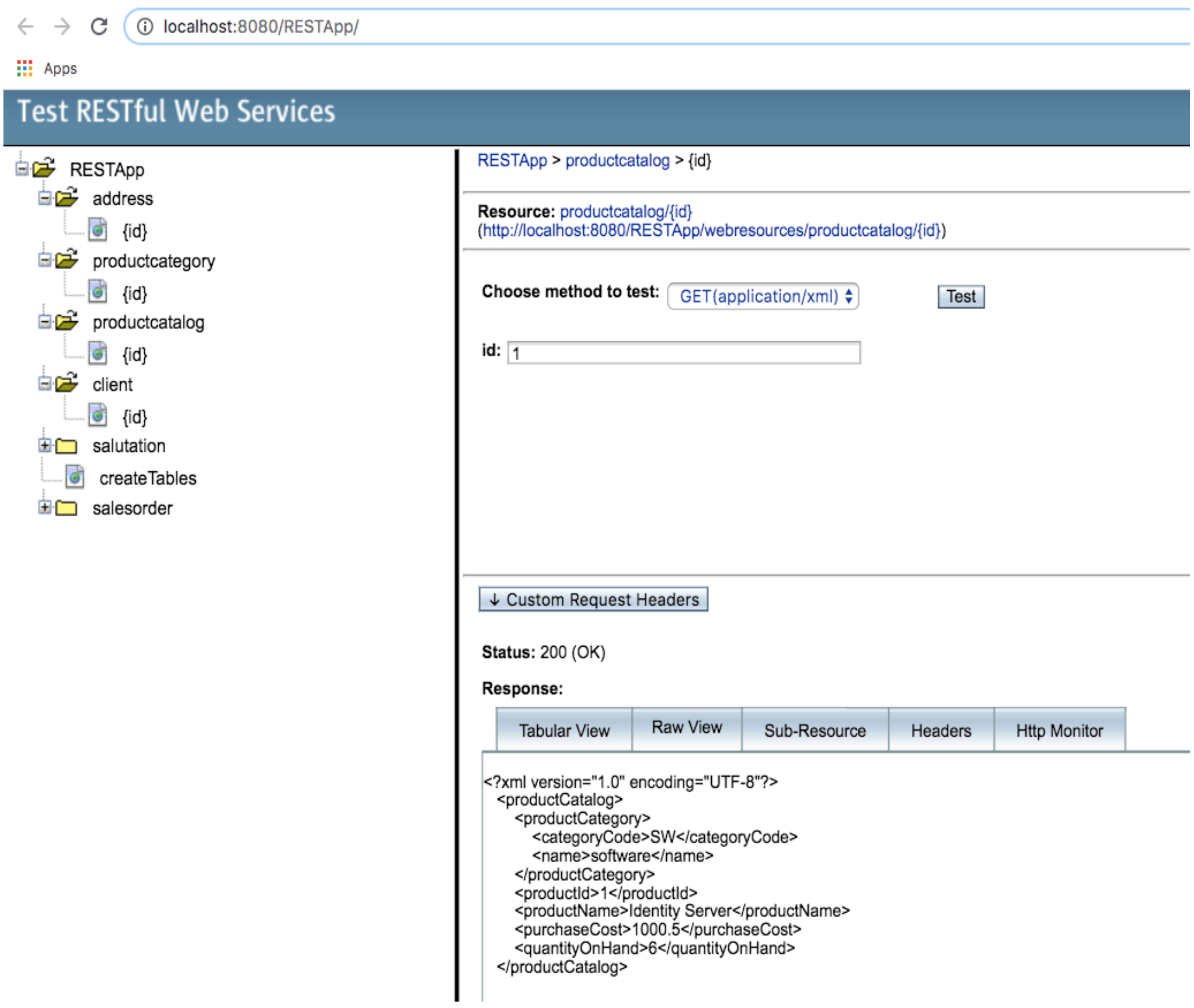
Choose method to test: GET(application/xml)

id: SW

Status: 200 (OK)

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <productCategory>
     <categoryCode>SW</categoryCode>
     <name>software</name>
  </productCategory>
```



RESTApp > productcatalog > {id}

Resource: productcatalog/{id}
(http://localhost:8080/RESTApp/webresources/productcatalog/{id})

Choose method to test: GET(application/xml)

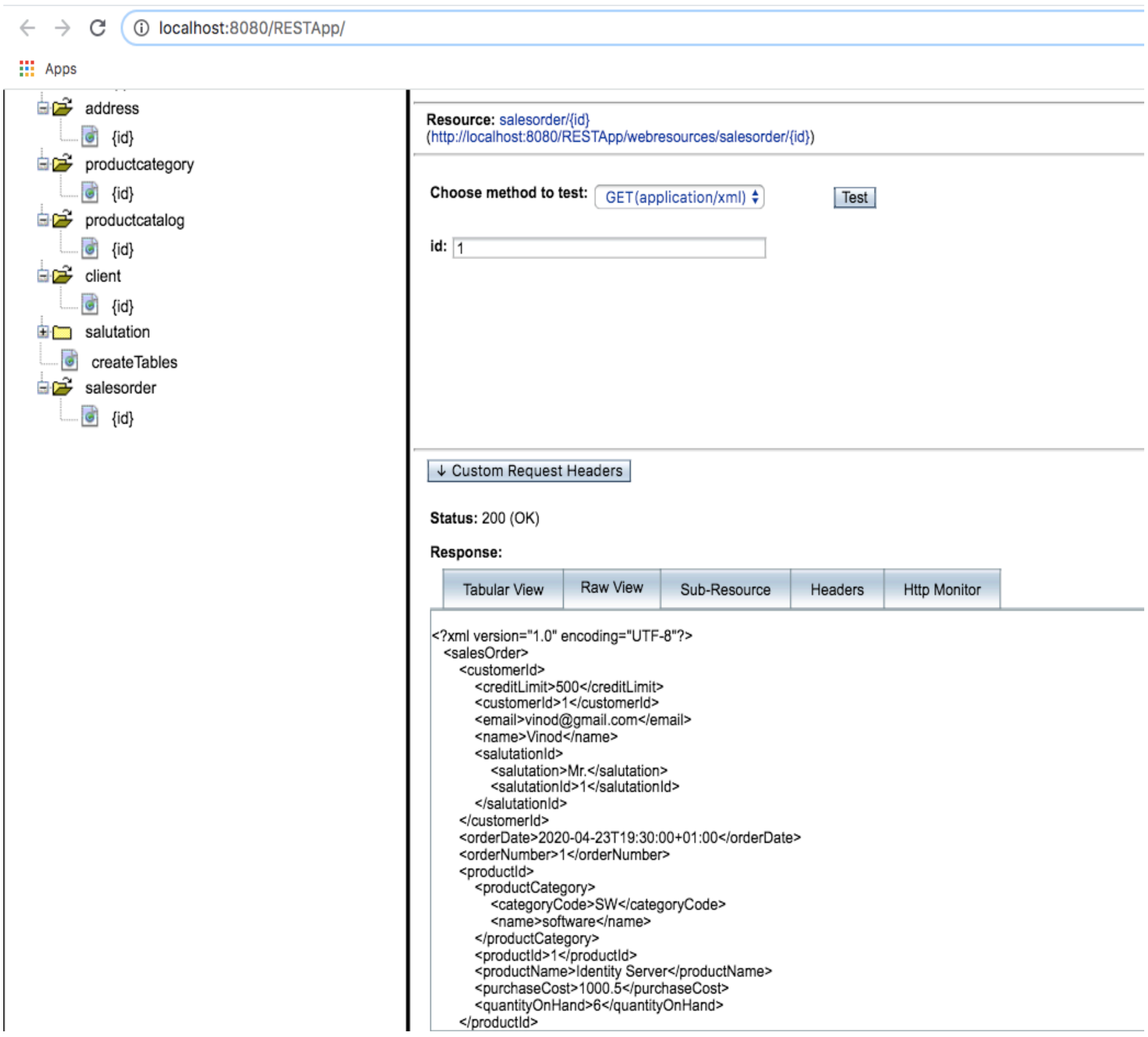
id: 1

Status: 200 (OK)

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <productCatalog>
     <productCategory>
        <categoryCode>SW</categoryCode>
        <name>software</name>
     </productCategory>
     <productId>1</productId>
     <productName>Identity Server</productName>
     <purchaseCost>1000.5</purchaseCost>
     <quantityOnHand>6</quantityOnHand>
  </productCatalog>
```

Test-Case 7:



Resource: salesorder/{id}
(http://localhost:8080/RESTApp/webresources/salesorder/{id})

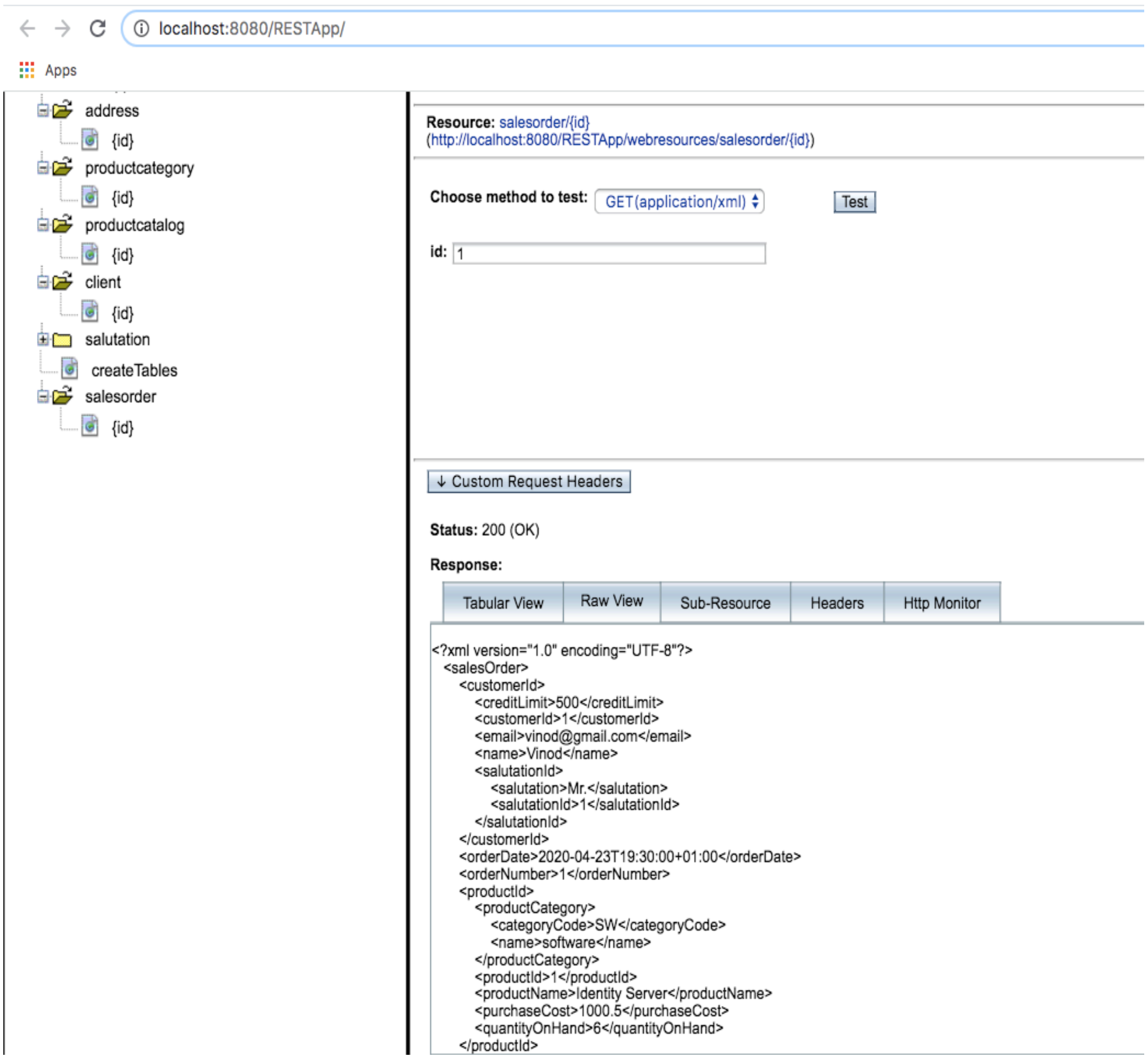
Choose method to test: GET(application/xml) Test

id: 1

Status: 200 (OK)

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <salesOrder>
    <customerId>
      <creditLimit>500</creditLimit>
      <customerId>1</customerId>
      <email>vinod@gmail.com</email>
      <name>Vinod</name>
      <salutationId>
        <salutation>Mr.</salutation>
        <salutationId>1</salutationId>
      </salutationId>
    </customerId>
    <orderDate>2020-04-23T19:30:00+01:00</orderDate>
    <orderNumber>1</orderNumber>
    <productId>
      <productCategory>
        <categoryCode>SW</categoryCode>
        <name>software</name>
      </productCategory>
      <productId>1</productId>
      <productName>Identity Server</productName>
      <purchaseCost>1000.5</purchaseCost>
      <quantityOnHand>6</quantityOnHand>
    </productId>
```



Resource: salesorder/{id}
(http://localhost:8080/RESTApp/webresources/salesorder/{id})

Choose method to test: GET(application/xml)

id: 1

Status: 200 (OK)

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
 <salesOrder>
    <customerId>
       <creditLimit>500</creditLimit>
       <customerId>1</customerId>
       <email>vinod@gmail.com</email>
       <name>Vinod</name>
       <salutationId>
          <salutation>Mr.</salutation>
          <salutationId>1</salutationId>
       </salutationId>
    </customerId>
    <orderDate>2020-04-23T19:30:00+01:00</orderDate>
    <orderNumber>1</orderNumber>
    <productId>
       <productCategory>
          <categoryCode>SW</categoryCode>
          <name>software</name>
       </productCategory>
       <productId>1</productId>
       <productName>Identity Server</productName>
       <purchaseCost>1000.5</purchaseCost>
       <quantityOnHand>6</quantityOnHand>
    </productId>
```