

# Taco Loco Backend Challenge

**By Ayesha Khader**

## Purpose

A backend service has been created in order to assist truck drivers and to help them manage customer addresses for their taco deliveries. The Delivery Service enables a truck driver to look up the entire deliveries database, add a new customer's name and address, update a delivery and delete a delivery. The database includes the delivery IDs, the customer names and the customer addresses.

## Coding Languages and Techniques

Primarily, the code has been written in Java and has been incorporated with JSON. Moreover, another programming language that has been extensively used is XML. Furthermore, the techniques mainly used in writing the code are: Jersey RESTful API and JAXB. JAXB has been used to convert Java objects to XML and vice versa. Similarly, Jersey uses MOXy to convert Java objects to JSON and vice versa.

## Prerequisites

The following prerequisites are required to run the program:

- [Eclipse IDE for Enterprise Java Developers](#)
- [Advanced REST Client Add-on for Chrome or Firefox](#)

## Functioning of The Taco Loco Delivery Service

In order to run the application, clone the GitHub directory and open the *DeliveryService* project on Eclipse. Next, right click on the project name in the *Project Explorer*

and click on *Run As* and *Run on Server*, as shown in Figure 1 below. Apache Tomcat version 9.0 has been used as the server.

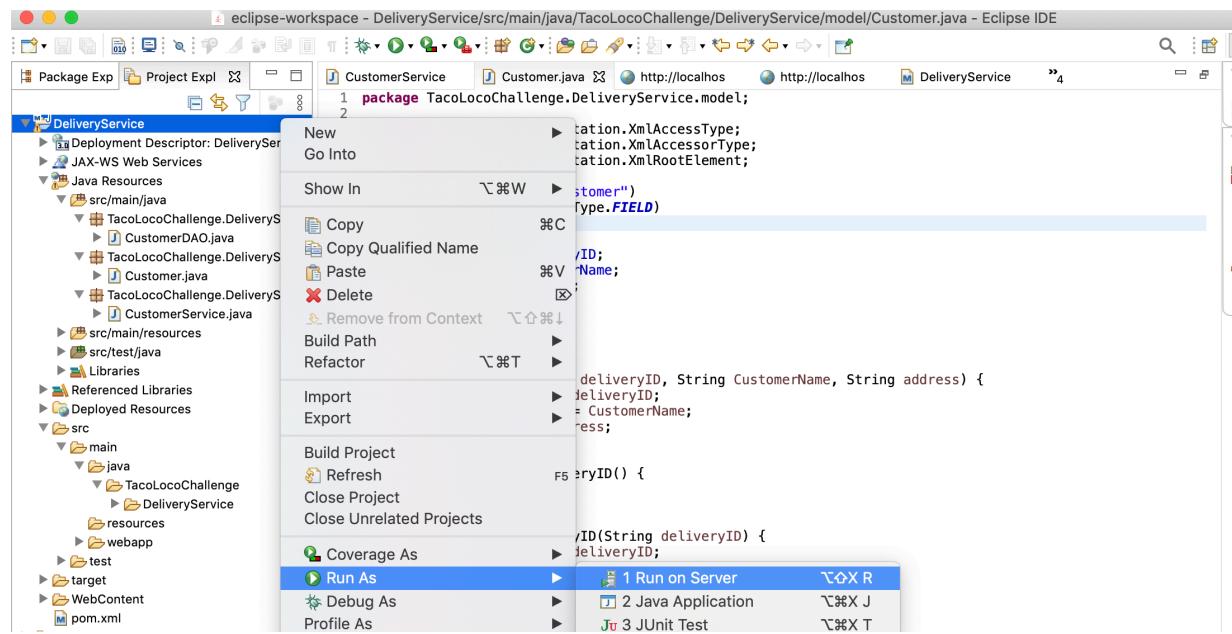


Figure 1: Running the Taco Loco Delivery Service Program

After the server starts running, the user will be navigated to the screen shown in Figure 2, below.

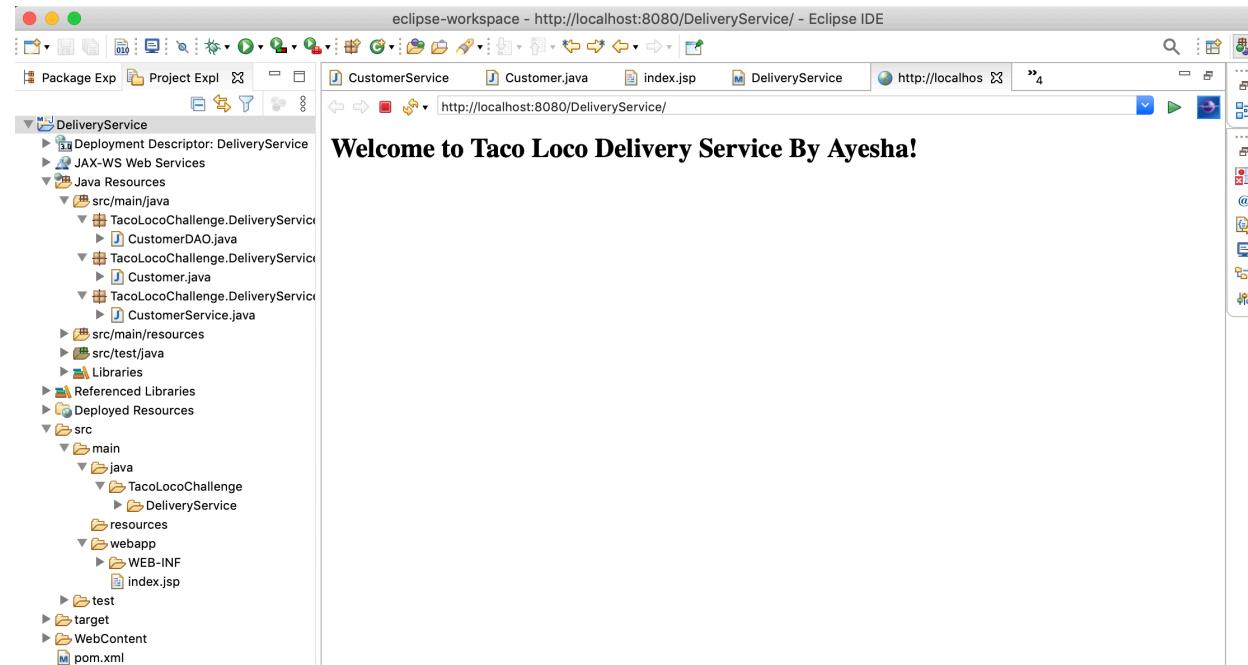


Figure 2: Delivery Service Running on The Server

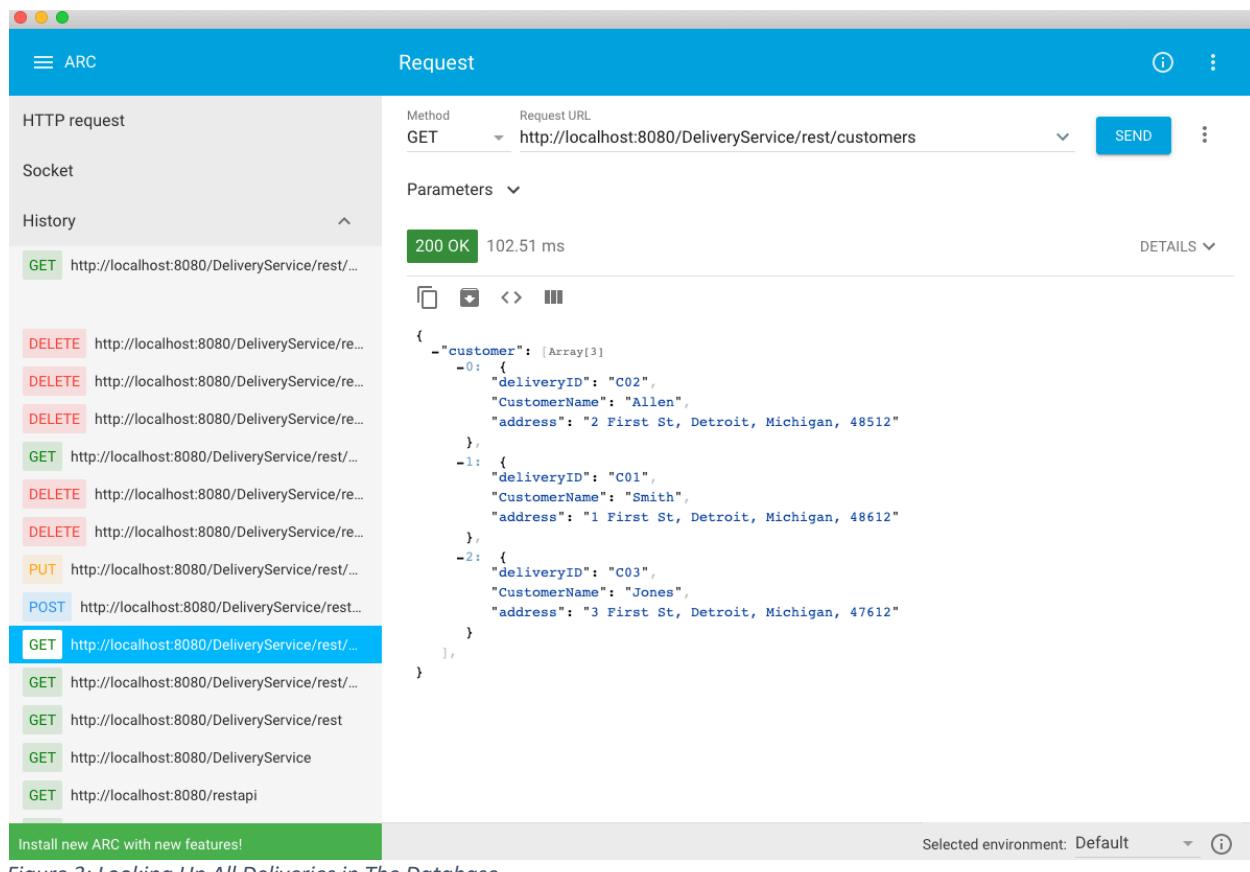
Next, launch the Advanced REST Client Add-on from either Chrome or Firefox.

## Looking Up All Deliveries in The Database

In order to lookup all deliveries in the database, the following path should be used:

<http://localhost:8080/DeliveryService/rest/customers>

Set the Method field to GET and click on the SEND button. This will display the entire database, as shown in Figure 3 below.



The screenshot shows the ARC interface with the following details:

- Request Tab:** Method is set to GET, Request URL is http://localhost:8080/DeliveryService/rest/customers.
- Parameters:** None
- Response:** Status is 200 OK, Time taken is 102.51 ms.
- Response Body (JSON):**

```
{
  "customer": [
    {
      "deliveryID": "C02",
      "CustomerName": "Allen",
      "address": "2 First St, Detroit, Michigan, 48512"
    },
    {
      "deliveryID": "C01",
      "CustomerName": "Smith",
      "address": "1 First St, Detroit, Michigan, 48612"
    },
    {
      "deliveryID": "C03",
      "CustomerName": "Jones",
      "address": "3 First St, Detroit, Michigan, 47612"
    }
  ]
}
```
- History:** Shows a list of previous requests, mostly DELETE operations, and the successful GET operation for the customers.
- Bottom Bar:** Includes a link to "Install new ARC with new features!" and a "Selected environment: Default" dropdown.

Figure 3: Looking Up All Deliveries in The Database

## Looking Up A Delivery in The Database by Delivery ID

In order to lookup a delivery in the database by delivery ID, add the DeliveryID at the end of the URL path.

For example, to look for DeliveryID number “C03”, the following path should be used:

<http://localhost:8080/DeliveryService/rest/customers/C03>

Set the Method field to GET and click on the SEND button. This will display the customer with that particular ID, as shown in Figure 4 below.

The screenshot shows the ARC tool interface. On the left, there's a sidebar with options like 'HTTP request', 'Socket', 'History', and 'Today'. The main area is titled 'Request' and shows a 'Method' dropdown set to 'GET' and a 'Request URL' input field containing 'http://localhost:8080/DeliveryService/rest/customers/C03'. A 'SEND' button is to the right. Below this, a 'Parameters' section shows a '200 OK' status with a duration of '50.67 ms'. The response body is displayed as a JSON object:

```
{  
  "deliveryID": "C03",  
  "CustomerName": "Jones",  
  "address": "3 First St, Detroit, Michigan, 47612"  
}
```

At the bottom, there's a green bar with the text 'Install new ARC with new features!' and a 'Selected environment: Default' dropdown.

Figure 4: Looking Up A Delivery by the DeliveryID

## Adding A New Delivery in The Database

In order to add a new delivery, the following path should be used:

<http://localhost:8080/DeliveryService/rest/customers>

Set the Method field to POST and expand the parameters section. The Headers section should be updated similar to Figure 5 shown below.

The screenshot shows the 'Headers' section of the ARC tool. It has a title 'Headers' and a toolbar with icons for 'Toggle source mode' and 'Insert headers set'. Below this, there are two rows for setting headers:

Header name	Header value
content-type	application/json

Figure 5: Headers Section

Update the body to add the new delivery's information and click on the SEND button. An example is shown in Figure 6 below.

The screenshot shows the ARC (Advanced REST Client) application window. The title bar says "Request". The left sidebar has a "HTTP request" section with "Socket" and "History" tabs. The history tab lists various requests, including several POST and GET requests to "http://localhost:8080/DeliveryService/rest/...". The main panel shows a "Request" configuration screen. The "Method" is set to "POST" and the "Request URL" is "http://localhost:8080/DeliveryService/rest/customers". The "Headers" tab is visible, and the "Body" tab is selected. Under "Body content type", "application/json" is chosen. The "Raw input" field contains the following JSON:

```
{
  "deliveryID": "C04",
  "CustomerName": "Ayesha",
  "address": "007 Franklin rd, Southfield, Michigan, 48512"
}
```

Below the body, the response status is "200 OK" and the time is "130.25 ms". There are "DETAILS" and "Selected environment: Default" buttons at the bottom right. A green banner at the bottom left says "Install new ARC with new features!"

Figure 6: Adding A New Delivery to The Database

Now, if a GET request is sent again to display all the deliveries in the database, the new delivery and the customer's information should be added, as shown in Figure 7 below.

The screenshot shows the ARC tool interface. In the top navigation bar, there are three colored dots (red, yellow, green), the text "ARC", and a "Request" tab. On the right side of the header are icons for help, settings, and more. The main area is titled "HTTP request". Under "Method", "GET" is selected, and the "Request URL" is "http://localhost:8080/DeliveryService/rest/customers". A "SEND" button and a more options menu are to the right. Below this, a "Parameters" dropdown is shown with "200 OK" and "24.04 ms". A "DETAILS" dropdown is also present. The main content area displays a list of recent requests and the detailed response for the current request. The response body is a JSON object:

```
{
  "customer": [Array[4]
    -0: {
      "deliveryID": "C02",
      "CustomerName": "Allen",
      "address": "2 First St, Detroit, Michigan, 48512"
    },
    -1: {
      "deliveryID": "C01",
      "CustomerName": "Smith",
      "address": "1 First St, Detroit, Michigan, 48612"
    },
    -2: {
      "deliveryID": "C04",
      "CustomerName": "Ayesha",
      "address": "007 Franklin rd, Southfield, Michigan, 48512"
    },
    -3: {
      "deliveryID": "C03",
      "CustomerName": "Jones",
      "address": "3 First St, Detroit, Michigan, 47612"
    }
  ]
}
```

At the bottom left, there's a green button for "Install new ARC with new features!". At the bottom right, it says "Selected environment: Default" with a dropdown arrow and a help icon.

Figure 7: Database Updated with The New Customer's Delivery Information

## Updating A Delivery in The Database

In order to update a delivery, the following path should be used:

<http://localhost:8080/DeliveryService/rest/customers>

Set the Method field to PUT and expand the parameters section. The Headers section should be updated similar to Figure 8 shown below.

Header name	Header value
content-type	application/json

Below the table are two buttons: "Toggle source mode" and "Insert headers set".

Figure 8: Headers Section

Update the body to update the delivery's information and click on the SEND button. An example is shown in Figure 9 below. Here, the customer's building number has been updated from 007 to 168 in the address field.

The screenshot shows the ARC (Apache REST Client) interface. On the left, there's a sidebar with 'HTTP request' selected, showing a history of requests. The main area is titled 'Request' with 'Method' set to 'PUT' and 'Request URL' as 'http://localhost:8080/DeliveryService/rest/customers'. The 'Body' tab is active, showing JSON code that updates a customer's address from '007' to '168'. Below the body, the response is shown as '200 OK' with a duration of '16.84 ms'. At the bottom, there's a note to 'Install new ARC with new features!' and a 'Selected environment: Default' dropdown.

```
{
  "deliveryID": "C04",
  "CustomerName": "Ayesha",
  "address": "168 Franklin rd, Southfield, Michigan, 48512"
}
```

Figure 9: Updating a Delivery

Now, if a GET request is sent again, to display all the deliveries in the database, the new delivery and the customer's information should be updated, as shown in Figure 10 below.

The screenshot shows the ARC tool interface. In the top navigation bar, there's a 'Request' tab and a 'SEND' button. The main area is titled 'HTTP request' and shows a 'Method' dropdown set to 'GET' and a 'Request URL' input field containing 'http://localhost:8080/DeliveryService/rest/customers'. Below this, there's a 'Parameters' dropdown and a status indicator '200 OK 37.72 ms'. The response body is displayed as a JSON object:

```
{
  "customer": [Array[4]
    -0: {
      "deliveryID": "C02",
      "CustomerName": "Allen",
      "address": "2 First St, Detroit, Michigan, 48512"
    },
    -1: {
      "deliveryID": "C01",
      "CustomerName": "Smith",
      "address": "1 First St, Detroit, Michigan, 48612"
    },
    -2: {
      "deliveryID": "C04",
      "CustomerName": "Ayesha",
      "address": "168 Franklin rd, Southfield, Michigan, 48512"
    },
    -3: {
      "deliveryID": "C03",
      "CustomerName": "Jones",
      "address": "3 First St, Detroit, Michigan, 47612"
    }
  ]
}
```

At the bottom left, there's a green button 'Install new ARC with new features!' and at the bottom right, a 'Selected environment: Default' dropdown.

Figure 10: Updated Customer's Address in the Delivery Database

## Deleting A Delivery from The Database

In order to delete a delivery in the database, add the DeliveryID at the end of the URL path.

For example, to delete the DeliveryID number “C04”, the following path should be used:

<http://localhost:8080/DeliveryService/rest/customers/C04>

Set the Method field to DELETE and click on the SEND button, as shown in Figure 11 below.

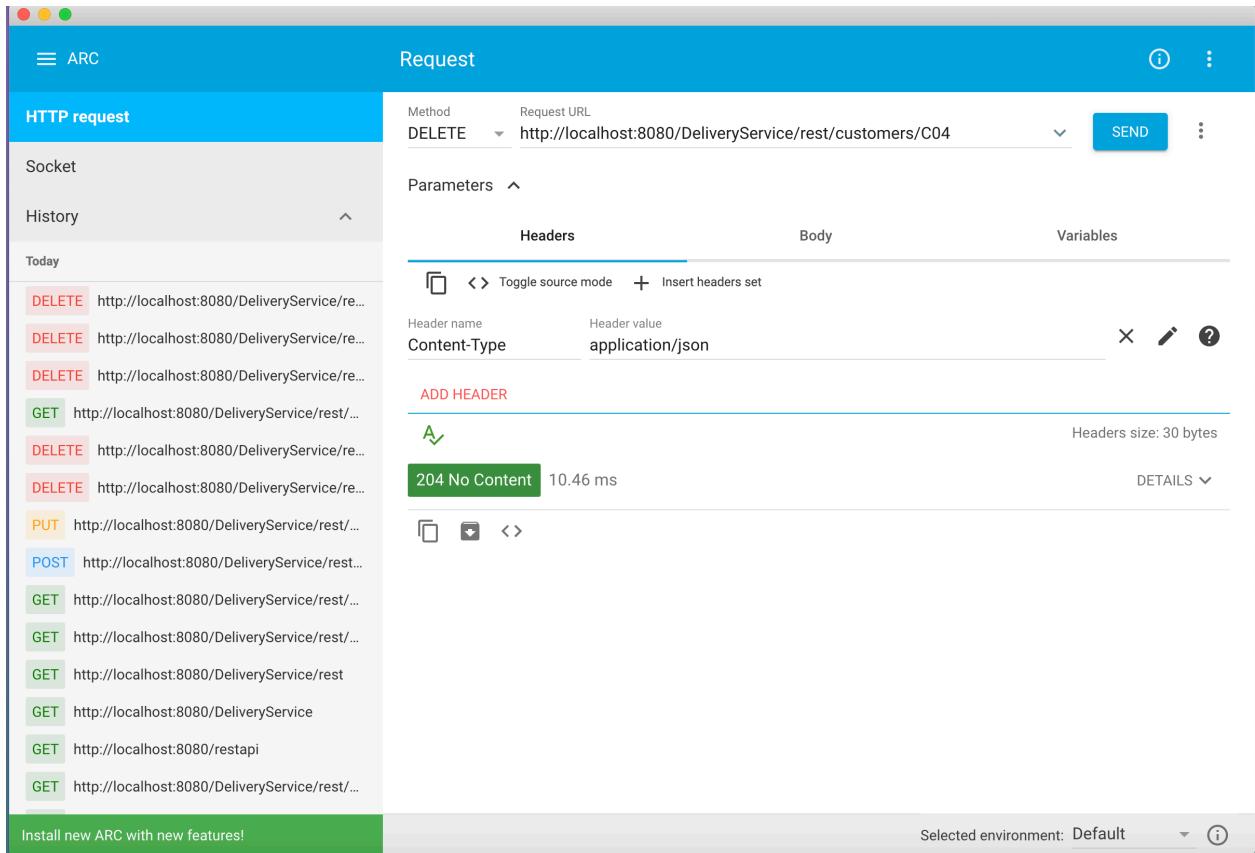


Figure 11: Deleting A Delivery from the Database

Now, if a GET request is sent again, to display all the deliveries in the database, the customer with that particular ID will be deleted, as shown in Figure 12.

The screenshot shows the ARC application window. In the top navigation bar, there are three dots (red, yellow, green), the title 'ARC', and a 'Request' button. To the right of the title are icons for help and more options.

The main area is titled 'HTTP request'. On the left, there's a sidebar with sections for 'Socket', 'History' (with a dropdown arrow), and 'Today'. Under 'History', there is a list of recent requests:

- DELETE** http://localhost:8080/DeliveryService/re...
- DELETE** http://localhost:8080/DeliveryService/re...
- DELETE** http://localhost:8080/DeliveryService/re...
- GET** http://localhost:8080/DeliveryService/rest/...
- DELETE** http://localhost:8080/DeliveryService/re...
- DELETE** http://localhost:8080/DeliveryService/re...
- PUT** http://localhost:8080/DeliveryService/rest/...
- POST** http://localhost:8080/DeliveryService/rest/...
- GET** http://localhost:8080/DeliveryService/rest/...
- GET** http://localhost:8080/DeliveryService/rest/...
- GET** http://localhost:8080/DeliveryService/rest/...
- GET** http://localhost:8080/DeliveryService
- GET** http://localhost:8080/restapi
- GET** http://localhost:8080/DeliveryService/rest/...

The main content area shows a successful request to 'http://localhost:8080/DeliveryService/rest/customers'. The status is '200 OK' and the time is '16.85 ms'. There are buttons for 'DETAILS' and a dropdown menu. Below the status, there are icons for copy, open in new tab, and refresh. The response body is displayed as JSON:

```
{
  "customer": [Array[3]]
  -0: {
    "deliveryID": "C02",
    "CustomerName": "Allen",
    "address": "2 First St, Detroit, Michigan, 48512"
  },
  -1: {
    "deliveryID": "C01",
    "CustomerName": "Smith",
    "address": "1 First St, Detroit, Michigan, 48612"
  },
  -2: {
    "deliveryID": "C03",
    "CustomerName": "Jones",
    "address": "3 First St, Detroit, Michigan, 47612"
  }
}
```

At the bottom left, there is a green button that says 'Install new ARC with new features!'. At the bottom right, it says 'Selected environment: Default' with a dropdown icon and a help icon.

Figure 12: Deleted DeliveryID "C04" from the Delivery Database