Consuming APIs: POST, PUT, and DELETE (Java)

In this exercise, you'll work on a command-line application that displays online auction info. A portion of the command-line application is provided. You'll write the remaining functionality.

Your task is to add web API calls using RestTemplate to create new auctions (`POST`), update existing auctions (`PUT`) and delete (`DELETE`) auctions.

Step One: Start the server

Before starting, make sure the web API is up and running. Open the command line and navigate to the `./server/` folder in this exercise.

First, run the command `npm install` to install any dependencies. You won't need to do this on any subsequent run.

To start the server, run the command `npm start`. If there aren't any errors, you'll see the following, which means that you've successfully set up your web API:

```
\{^_^}/ hi!

Loading data-generation.js
Done

Resources
http://localhost:3000/auctions

Home
http://localhost:3000

Type s + enter at any time to create a snapshot of the database
```

You can stop the server, or any other process that you've started from the console, by using the keyboard shortcut `ctrl + c`.

In this exercise, you'll modify data on the server. As you're working, you may come across a situation where you want to reset the data. To do this, first stop the server with `ctrl + c`, then restart it with `npm start`.

Step Two: Explore the API

Before moving on to the next step, explore the web API using Postman. You can access the following endpoints:

- GET: http://localhost:3000/auctions
- GET: http://localhost:3000/auctions/{id} (use a number between 1 and 7 in place of {id})

These are the endpoints you'll work on for this exercise:

- POST: http://localhost:3000/auctions
- PUT: http://localhost:3000/auctions/{id}
- DELETE: http://localhost:3000/auctions/{id}

Step Three: Evaluation criteria and functional requirements

- All unit tests pass `/src/test/java/com/techelevator/services/AuctionServiceTest.java`.
- Code is clean, concise, and readable.

To complete this exercise, you need to complete the `AuctionService` class by implementing the `add()`, `update()`, and `delete()` methods.

Tips and tricks

- There are two helper methods available. One makes an `Auction` object given a CSV string containing either three or four elements. The second creates an `HttpEntity` with a content-type header set to JSON.
- The URL for the API is declared in `App.java` where the `AuctionService` is instantiated. You may need to append a slash depending on the API method you're using.
- The `add()` method takes a string as a parameter that's passed from the console. It's a CSV string for a new auction. Use the helper methods. The `add()` method must return an `Auction` object.
- The `update()` method takes a string as a parameter that's passed from the console. It's a CSV string for an existing auction. Use the helper methods. The `update()` method must return an `Auction` object.
- The `delete()` method takes an integer as a parameter that's passed from the console. It's the `id` of the auction to delete. The `delete()` method doesn't return anything.
- Consider that the server may return an error.

Step 5: Add a new auction

The add method must create a new auction. You can use the helper methods mentioned earlier to make a new auction and http entity. Make sure to handle any exceptions that might be thrown:

```
public Auction add(String auctionString) {
    // place code here
    return null;
}
```

When you've completed the `add()` method, run the unit tests, and verify that the `add()` test passes.

Step 6: Update an existing auction

The update method overrides the existing auction with the updated one. You can use the helper methods mentioned earlier to make a new auction and http entity. Make sure to handle any exceptions that might be thrown:

```
public Auction update(String auctionString) {
    // place code here
```

```
        return null;
    }
```

When you've completed the `update()` method, run the unit tests, and verify that the `update()` test passes.

Step 7: Delete an auction

The delete method removes an auction from the system. Make sure to handle any exceptions that might come up. What happens if you enter an ID for an auction that doesn't exist?

```
public void delete(int id) {
    // place code here
}
```

When you've completed the `delete()` method, run the unit tests, and verify that the `delete()` test passes.

Once all unit tests pass, you've completed this exercise.