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

In this exercise, you'll work on a command-line application that displays online auction info. Most of the command-line application is provided. You'll add more code that calls the API.

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 in `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 is a helper method available that creates an `HttpEntity` with a content-type header set to JSON.
- The `add()` method takes an `Auction` object as a parameter that's passed from the console. The `add()` method must return the `Auction` object that comes back from the API.
- The `update()` method takes an `Auction` object as a parameter that's passed from the console. The `update()` method must return `true` if no errors occur when calling the API, and `false` otherwise.
- 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 must return `true` if no errors occur when calling the API, and `false` otherwise.
- When writing all three methods, consider that an error may occur when calling the API.

Step Four: Add a new auction

The `add()` method sends the `newAuction` parameter to the API. You can use the helper method mentioned earlier to make a new `HttpEntity`. Make sure to handle any exceptions that might be thrown:

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

When you've completed the `add()` method, run the unit tests, and verify that the three tests that begin with `add_` all pass.

Step Five: Update an existing auction

The `update()` method sends the `updatedAuction` parameter to the API to replace the existing one. You can use the helper method mentioned earlier to make a new `HttpEntity`. Make sure to handle any exceptions that might be thrown:

```
public boolean update(Auction updatedAuction) {
    // place code here
    return false;
}
```

When you've completed the `update()` method, run the unit tests, and verify that the three tests that begin with `update_` all pass.

Step Six: Delete an auction

The `delete()` method removes the auction from the system with the ID that matches the `auctionId` parameter. 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 boolean delete(int auctionId) {
    // place code here
    return false;
}
```

When you've completed the `delete()` method, run the unit tests, and verify that the three tests that begin with `delete_` all pass.

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