Connecting your Car Dealership to a Database

Workbook 9's Workshop

Project Description

In this workshop you will add onto the car dealership that you have been building in previous workshops. You will convert your application into a RESTful API

Using the SpringBoot Initializer you will create a new maven project with the MySQL Driver dependency. Don't forget to add any other dependencies required for your project to your `pom.xml` file. (hint: BasicDataSource)

NOTE: Once you have created your new project you may want to copy your Jdbc DAOs from Workshop8 to use as starter files.

Project Details

The purpose of this project is to convert your CLI application into a web based API.

Programming Process

You should spend the first few minutes setting up your project and porting your existing Workshop5 project. There are many ways to go about this, but if you are unsure, we suggest:

1. Create a new GitHub repository for this workshop named CarDealershipAPI and clone it to your `C:\pluralsight\workshops` folder
2. Go to `https://start.spring.io` and create a new Java Maven project (dealership-api)
3. Add the Spring Web dependency
4. Create a package for the source code (com.pluralsight.dealership)
5. Generate and unzip your starter project to your repo
6. Use your Workshop8 project to copy your car dealership models and database DAO classes to your new project
7. Build and test your program to make sure it you are able to connect to your API
8. Commit and push your changes.

Now you are ready to begin!

If you need any additional models to complete this workshop, add them to the project.

Phase 1: Add Dependency Injection

Add application.properties for your database connection string

Create a `DatabaseConfiguration` class to create a BasicDataSource bean.

Add the `@Component` annotation to your JDBC data access objects.

Phase 2: Adding Controllers

Your Data Access Objects should have all the capabilities to interact with your database. The DAOs should have the ability to perform all CRUD operations.

VehiclesController

Create a `GET` method: that accepts query string parameters for:

- `minPrice`
- `maxPrice`
- `make`
- `model`
- `minYear`
- `maxYear`
- `color`
- `minMiles`
- `maxMiles`
- `type`

Create a `POST` method: to allow a user to add a new vehicle

Create a `PUT` method: to allow the user to update vehicle information

Create a `DELETE` method: to allow the user to delete a vehicle

SalesContractsController

Create a `GET` method: to get a sales contract by Id

Create a `POST` method: to add a new Sales contract to the database

LeaseContractsController

Create a `GET` method: to get a lease contract by Id

Create a `POST` method: to add a new Lease contract to the database

What Makes a Good Workshop Project?

- **You should:**
  - Create controllers and API endpoints for all CRUD operations
  - Implement the ability for a user to search/add/remove vehicles from the database

- **You should adhere to best practices such as:**
  - Create a Java Project that follows the Maven folder structure
  - Create appropriate Java packages and classes
  - Class names should be meaningful and follow proper naming conventions (PascalCase)
  - Use good variable naming conventions (camelCasing, meaningful variable names)
  - Your code should be properly formatted easy to understand
  - use Java comments effectively
  - Include a script for the Car Dealership database

- **Make sure that:**
  - Your code is free of errors and that it compiles and runs
  - Test your API endpoints using Postman to ensure that all methods work as expected

- **Build a PUBLIC GitHub Repo for your code**
  - Include a README.md file that describes your project and includes screen shots of
    - your home screen
    - your products display screen
    - one calculator page that shows erroneous inputs and an error message.
  - ALSO make sure to include one interesting piece of code and a description of WHY it is interesting.