Connecting your Car Dealership to a Database

Workbook 8's Workshop

Project Description

Use the car-dealership project that you created in Workshop 5 as a starting point for this workshop. You will combine that application with the database script that you created in Workshop 7.

Use IntelliJ to create a new maven project. Modify your `pom.xml` file to add a MySQL Driver dependency. Don't forget to add any other dependencies required for your project. (hint: BasicDataSource) Use Workshop 5 as a guide for the application flow and the classes that you will need. Keep in mind that because you are working with a database, the models and application flow will not be identical.

NOTE: You can also choose to copy the files from your Workshop 5 project into your new repository, and then manually add the database dependencies to your `pom.xml` file.

Project Details

The purpose of this project is to move your application away from using `.csv` files, and instead connect to the CarDealership database that you created last week to store and manage all your application data.

Programming Process

You should spend time setting up your project and porting your existing Workshop 5 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 and clone it to your `C:/pluralsight/workshops` folder
2. Create a new Java Maven project (jdbc-dealership-project)
3. Create a package for the source code (com.pluralsight.dealership)
4. Copy your existing Workshop 5 files to your new project
5. Build and test your program to make sure it works
6. Commit and push your changes.

Now you are ready to begin!

Your application should already have the models that you will need for this project. But you may need to modify them slightly so that they will work with your database (i.e. some of the models may need to have an id field added to them)

Models should include:
`Dealership, Vehicle, SalesContract, LeaseContract`

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

Changes to your Persistence Logic

Currently all data is stored in `.csv` files and you use a `FileManager` class to read from and write to the `.csv` files.

You need to create new `DataManager` classes which connect to your database and perform all CRUD operations. You should work through all of the options on your dealership home screen, and ensure that the data on each page is coming from (or going to) the database instead of the .csv files.

NOTE: You can choose to create a single DataManager class, or you can create a different DataManager (or DAO) for each table in the database. (Using individual DAO's will help you separate the logic into smaller components - this will be useful next week)

Sale vs Lease

When a user selects the SALE option the contract data should be stored in the `sales_contracts` table.

When a user selects the LEASE option the contract data should be stored in the `lease_contracts` table.

Phase 1

Phase 1 should be to create the `VehicleDao` and to create the methods required to allow a user to search the database for vehicles:

1. By price range
2. By make/model
3. By year range
4. By color
5. By mileage range
6. By type

Phase 2

Phase 2 should be the to add and remove vehicles from the database

Phase 3

Phase 3 involves creating the SalesDao and LeaseDao classes and update your dealership logic to save the sales/lease information to the database.

Delete all `.csv` files from this project and verify that the project still works as expected.

Bonus Ideas

If you finish early (and if you created an Admin interface before) modify your `AdminUserInterface` to pull all sales and lease information from the database.

What Makes a Good Workshop Project?

- **You should:**
  - Have a clean and intuitive user interface (give the user clear instructions on each screen)
  - Implement the ability for a customer to add/remove items to a cart and also to purchase the items in the cart

- **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

- **Make sure that:**
  - Your code is free of errors and that it compiles and runs
  - Include a script for the Car Dealership database

- **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.