Car Search Assignment

Author: Dinh Van Vu
Date: 23/064/2015

1. Design Decisions

ASP.NET MVC 4.0 framework was chosen due to the requirements of the assignment. When designing the data access layer, I used Navicat to analyze the Amazone AWS database and Soap UI to select the correct API from the the Webservice. Probing Webservice shows that the web service can be added as a service reference to project as usual. However, the table in Amazone AWS database has no primary key. While this is a bad practice, it also makes it impossible to use Entity Framework in the project. All queries will return the same record.

Therefore, I used ADO.NET with plain SQL query for data access. As data in the table can be null, I use a Nullable data reader to handle null data. Reference:
http://www.codeproject.com/Articles/12778/C-Nullable-Data-Readers

To setup the required services, I used IoC Dependency Injection (DI). For example, the backbone Search service of the application depends on two other services:

1. An Web service to return a search criteria in the car database based on the car registration number.
2. A database search service to return the car information based on the above search criteria.

Both services are injected into the backbone Search service. The database search service need a connection string to establish the connection to database, thus its constructor is also injected during the bootstrapping phase. The Unity container was chosen to do the DI.

Another requirement of the assignment is the format of displayed search result. To satisfy this requirement, I use fluent Builder to format the display object with input from other source, e.g. database.

Finally, as the application relies heavily on the remote services, the application need an exception handler. While it is feasible to implement a exception handler mechanism at application level, I chose to override the *OnException* handler of the controller **SearchController.** The reason is I want to enabkle user to use the same form for another search after an exception. This form includes a partial view that displays the search result in normal case. In case of the exception, the partial view will show the error message. The partial view also allows us to display search result in flexible position in a web page.

Any design must satisfy at least two criteria:
1. Isolation of concerns.
2. Code reuse.

For the 1st criteria, the implementation of search business rule is encapsulated in method *buildSQLCommand* of class **CarRepository**. For the 2nd criteria, thanks to the DI, we can easily inject another search service not ADO.NET into the **SearchService** class.

In the future work, we need to implement Unit tests and factory to select different search services.

2. Class Diagram

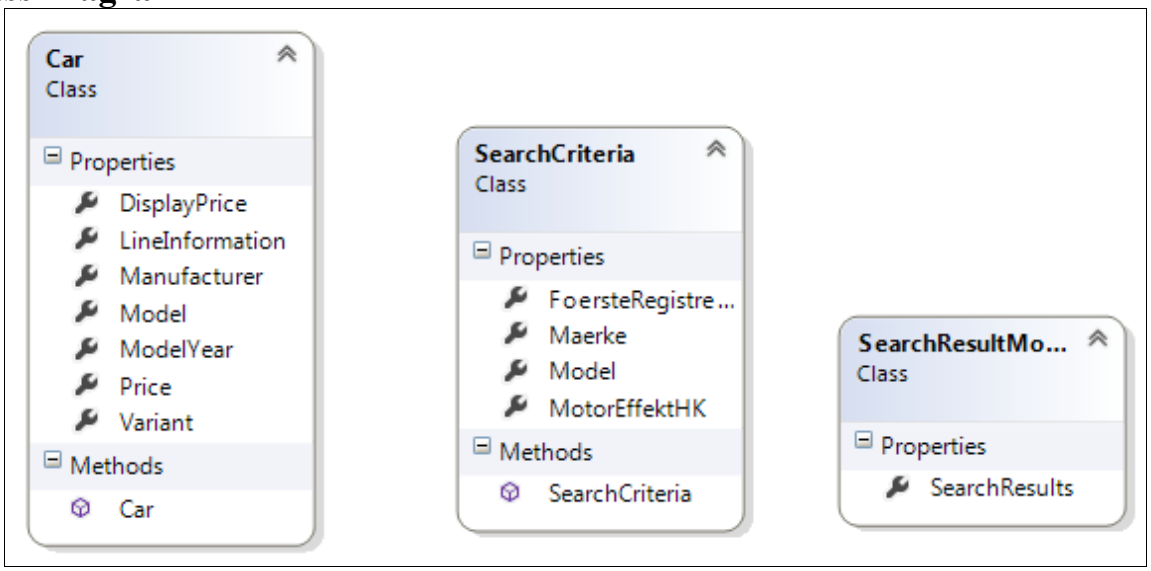

Figure 1: Models

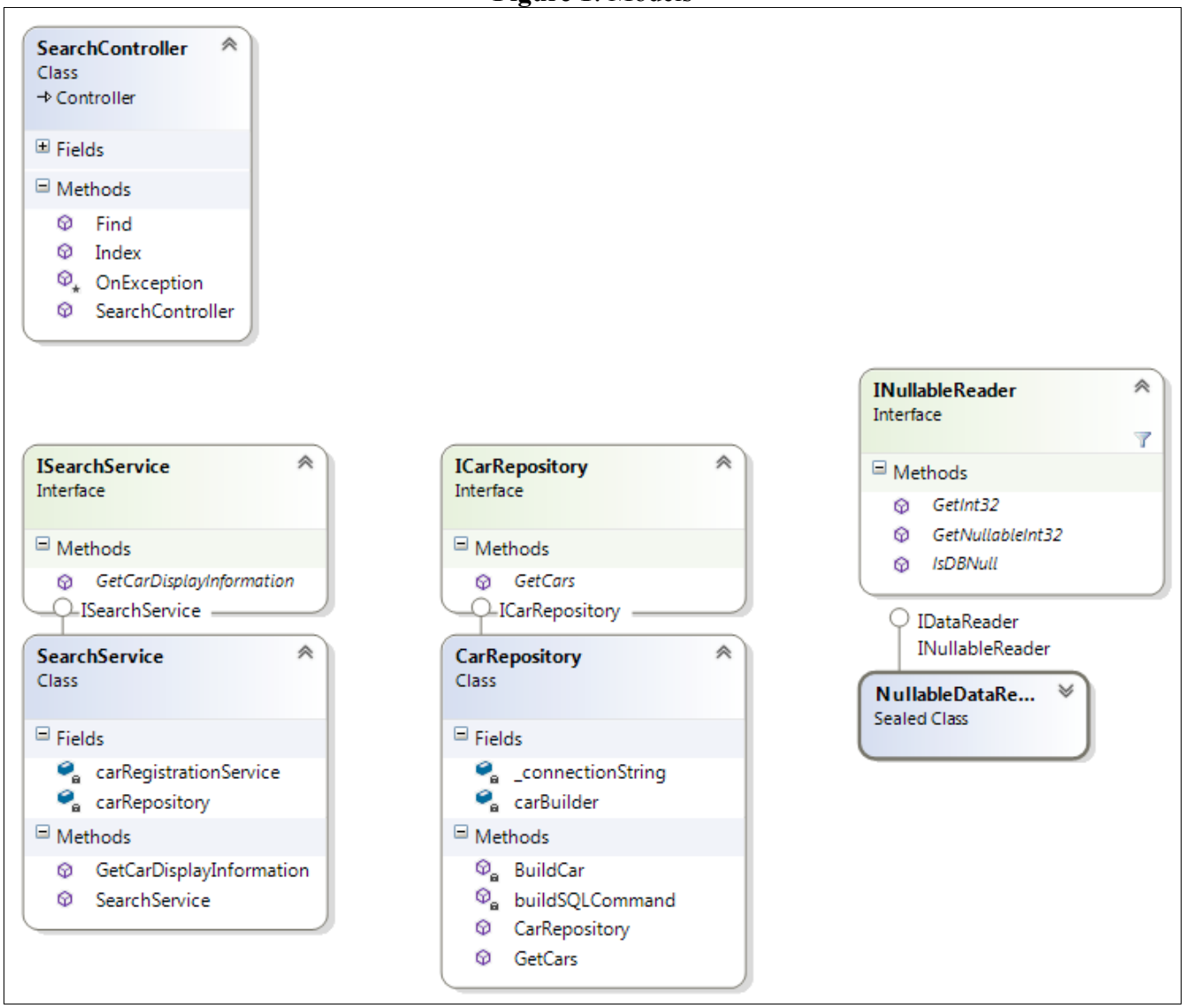

Figure 2: Controller and Services