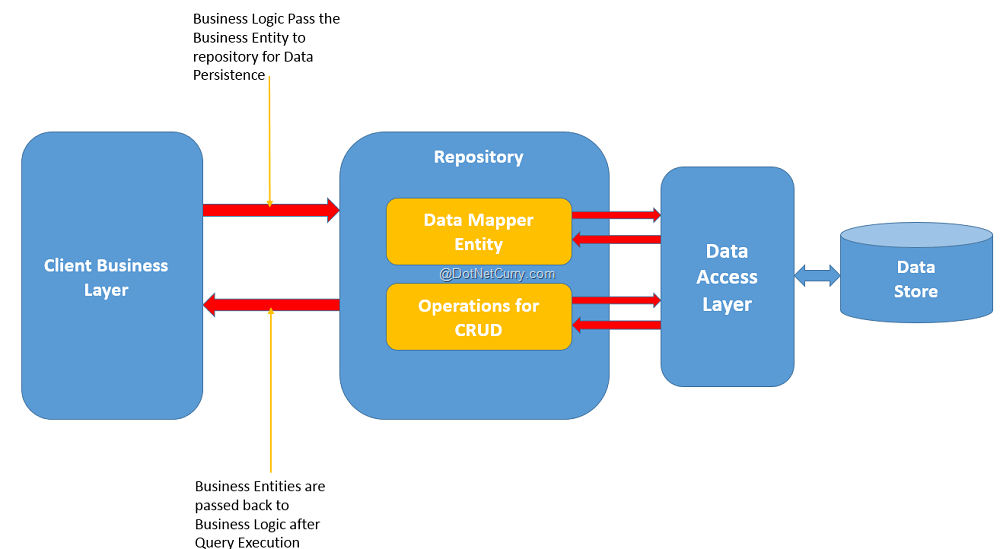
Typically in multi-layer applications, the business logic or Domain logic retrieves data from various sources, e.g. Database, Web/WCF services, etc. In this type of direct access, the business logic is tightly coupled with the data access code which may result in code maintenance issues; because if the Database or Service changes, then the dependency code also needs to be changed. So the point here is that how to minimize this maintenance effort?

* Implementing an isolation layer for the data access which can be effectively tested.
* Implement consistent data access logic irrespective of the data source locations.
* Implementing maintainable code by separating business logic from data access logic.
* Implementing an easy domain model development for complex business logic.



The repository layer isolates Business layer from the Data Access Layer. The Repository contains Data Mapper entity. This entity can be used as a model entity for providing schema of the data for performing CRUD operations, by using the CRUD operations defined in the repository. The Business layer performs Data Access using repository layer. If the application makes use of WCF/Web services for data access, then repository can be used to isolate these service references from the Business layer.

**Repository in case of ASP.NET MVC**

In case of ASP.NET MVC, we uses the Model layer for interacting with Data Access and Controller talks to the Model for performing the Data Access operations. In MVC, it is important to think of the repository pattern so that if the Model layers needs some breaking changes, then it should have minimum or no maintenance impact on the controller layer. In MVC, finally the Controller is responsible for exposing data to the View or accessing Http posted data from View and send it further to the Model. So here, if the Controller has tight-coupling dependency on the Model, then any change in the Model will impact the Controller and hence sometimes the View too. In this case, we can use the Repository pattern as shown in the following diagram:

