# Repository Pattern

According to MSDN, a **repository separates the business logic from the interactions with the underlying data source** or Web service. The separation between the data and business tiers has three benefits:

1. It centralizes the data logic or Web service access logic.
2. It provides a substitution point for the unit tests.
3. It provides a flexible architecture that can be adapted as the overall design of the application evolves.

a repository basically works as a **mediator between our business logic layer and our data access layer of the application. Sometimes**, it would be troublesome to expose the data access mechanism directly to business logic layer, it may result in redundant code for accessing data for similar entities or it may result in a code that is hard to test or understand.

The repository makes queries to the data source for the data, thereafter maps the data from the data source to a business entity/domain object, finally and persists the changes in the business entity to the data source.

public MyController()

{

this.userRepository = new UserRepository(new MVCEntities());

}

# ****Inversion of Control (IoC)****

**Inversion of Control (IoC)** means that **objects do not create other objects** on which they rely to do their work. Instead, they get the objects that they need from an outside source (for example, an xml configuration file).

The advantages of using Dependency Injection pattern and Inversion of Control are the following:

* Reduces class coupling
* Increases code reusing
* Improves code maintainability
* Improves application testing

dependency injection

Dependency injection is a software design pattern that allows the removal of hard-coded dependencies and makes it possible to change them, whether at run-time or compile-time

DI is one way in which IoC can be applied

The Dependency Injection pattern is a particular implementation of Inversion of Control. **Inversion of Control**

 DI and Service Locator patterns are specialized versions of the IoC pattern or you can say DI and Service Locator are the ways of implementing IoC

DI is a software design pattern that allow us to develop loosely coupled code