Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application

The Repository and Unit of Work Patterns

The repository and unit of work patterns are intended to create an abstraction layer between the data access layer and the business logic layer of an application. Implementing these patterns can help insulate your application from changes in the data store and can facilitate automated unit testing or test-driven development (TDD).

In this tutorial you'll implement a repository class for each entity type. For the Student entity type you'll create a repository interface and a repository class. When you instantiate the repository in your controller, you'll use the interface so that the controller will accept a reference to any object that implements the repository interface. When the controller runs under a web server, it receives a repository that works with the Entity Framework. When the controller runs under a unit test class, it receives a repository that works with data stored in a way that you can easily manipulate for testing, such as an in-memory collection.

Later in the tutorial you'll use multiple repositories and a unit of work class for the Course and Department entity types in the Course controller. The unit of work class coordinates the work of multiple repositories by creating a single database context class shared by all of them. If you wanted to be able to perform automated unit testing, you'd create and use interfaces for these classes in the same way you did for the Student repository. However, to keep the tutorial simple, you'll create and use these classes without interfaces.

The following illustration shows one way to conceptualize the relationships between the controller and context classes compared to not using the repository or unit of work pattern at all.

