

Software Engineering Stack Exchange is a question and answer site for professionals, academics, and students working within the systems development life cycle. Join them; it only takes a minute:

[Sign up](#)

Here's how it works:

Anybody can ask a question

Anybody can answer

The best answers are voted up  
and rise to the top



SOFTWARE ENGINEERING

## Is it right to have dependencies to my Viewmodel from data acces layer and view layer

### Context

- 1 In my application I use the MVVM pattern. I'll have a ViewModel that can contains information of a student. This viewmodel is used to

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



*My project contains out of the following elements:*

1. Views (UI)
2. Controller API (responsible for responding to requests from the view)
3. Models (logic)
4. ViewModels(Containing information for the views)
5. Repository (repository pattern for communicating whit database)

## Senario

When the end user creates for example a student it will fill in some data and and press button to safe the student. Now the information will be send to the server whit a ajax call. The Controller API now catches the HTTP request and cast it to the view model.

### Controller API

```
[HttpPut]
public JsonResult Put([FromBody] StudentViewModel studentViewModel)
{
    //...
}
```

*This is the process in steps:*

1. The view makes an ajax request to the Controller Api. This request has all the property's of the StudentViewModel so it can be parsed in the Controller Api.
2. The Controller Api takes in the information and will parse it to the StudentViewModel
3. The Controller Api asks the repository to create a new student from the StudentViewmodel.
4. The repository will create a new student from the information in the StudentViewModel

I have my data acces layer wrapped in repository pattern. It will look something like this:

### Controller API

```
[HttpPut]
public JsonResult Put([FromBody] StudentViewModel studentViewModel)
{
    r
```

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

```
        db.Students.CreateNew(studentViewModel);  
    }  
}
```

But for some reason I do not like this approach. Because I now have my View layer and data layer depend on the same viewmodel. The view is heavily bound to the StudentViewModel because the data it provide should be convertible and this same VieModel is used for the database to create a new user.

## Question

Is there a better way for this scenario? Or is this just fine.

[c#](#)[design-patterns](#)[asp.net](#)[mvvm](#)[dependencies](#)

edited Oct 28 '17 at 17:57

asked Oct 28 '17 at 17:31



[Timon Post](#)

108 4

## 1 Answer



ViewModel objects and Model objects are not the same thing.

2

The purpose of a Model object is to model something in your domain. The purpose of a ViewModel object is to model something in your view.



Do you see the difference?




The reason this doesn't look right to you is that the Model shouldn't know anything about your ViewModels. Your repository should only know about Models.

There are a couple of ways you can fix this. One way is to build a Model object from the studentViewModel *in your controller*, and pass that to your repository. The other way to do it is to add a *service layer* that accepts ViewModel objects and translates them into Model objects for your repository.

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

---

I know the difference between the view model and model. But if I follow your advice I would end up whit 3 classes containing almost the same data. 1. The database `Student model` (entity framework model) 2. The `Student model` whits I pass to the repository. 3. And the `Student ViewModel` right? Or should I just use my database model (entity framework model) and pass that to the repository. — [Timon Post](#) Oct 28 '17 at 17:49 

---

You have to think of it more in terms of CRUD and business processes. Your Repository is about CRUD. Your Service Layer is about business processes. If you're just passing CRUD through to the database (i.e. your ViewModel and Model are identical), then you're right; you probably don't need any more abstraction. — [Robert Harvey](#) ♦ Oct 28 '17 at 17:54

---