

[articles](#) [quick answers](#) [discussions](#) [features](#) [community](#) [help](#)

Search for articles, questions, tips

[Articles](#) » [Languages](#) » [C#](#) » [General](#)

# Mapping ViewModel to Model in ASP.NET MVC using Implicit Conversion Operator in C#

**Ehsan Sajjad**27 Oct 2015 [CPOL](#)Rate this:  4.87 (24 votes)

How to map ViewModel to Model or vice versa using implicit operator in C#

## Background

In ASP.NET MVC, we have three important things in which we are moving all the time which is Model, View and Controller. Sometimes, we want specific information of model to be passed from View to action, but if we use the Model classes that are mapped to our database tables make things messy, as all the model is round tripping from View to action or vice versa.

Consider the following model class which is mapped to the user table in my database.

[Hide](#) [Shrink](#)  [Copy Code](#)

```
namespace JIRA.Domain.Models
{
    public class User
    {
        public int UserID { get; set; }
    }
}
```

```
public string FirstName { get; set; }

public string LastName { get; set; }

public string UserName { get; set; }

public string Password { get; set; }

public bool IsActive { get; set; }

public bool IsDeleted { get; set; }

public DateTime CreatedAt { get; set; }

public int CreatedBy { get; set; }

public DateTime UpdatedAt { get; set; }

public int UpdatedBy { get; set; }

public string Email { get; set; }
}
```

Here is my database table:

[Hide](#) [Copy Code](#)

```
CREATE TABLE [dbo].[tblUser] (
    [UserID] INT IDENTITY (1, 1) NOT NULL,
    [FirstName] NVARCHAR (25) NULL,
    [LastName] NVARCHAR (25) NULL,
    [UserName] NVARCHAR (25) NULL,
    [Password] NVARCHAR (25) NULL,
    [IsActive] BIT NULL,
    [IsDeleted] BIT NULL,
    [CreatedBy] INT NULL,
    [CreatedAt] DATETIME NULL,
    [UpdatedBy] INT NULL,
    [UpdatedAt] DATETIME NULL,
    [Email] NVARCHAR (50) NULL,
    PRIMARY KEY CLUSTERED ([UserID] ASC)
);
```

So what happens normally is developers strongly type their view with the model class that is mapped with the table in our db which is not a good approach, as our View doesn't need all information of the table every time.

## Scenario

Now consider the scenario of Register/SignUp of user in which we have different fields from which some will map to the **User** class but as **User** is registering so some properties of **Model** are useless here which will be posted when user submits form and there are some additional properties that we may need but they are not mapped in the table. You can take an example when user registers we take Password from user two times for Confirming, in that case, we don't want to change our **Model** that represents our **Entity** in the database, so **ViewModel** comes in.

## ViewModels and Models

**ViewModels** are specific to the Views, we put information in View Model that we need on the particular View. Here is the snippet that is not a preferred way. So now, we will create a **ViewModel** for Register View which will have properties specific to that View that it needs to post and we will map the **ViewModel** properties to **Entity Model** that represents our table and will insert it in the database.

## Example

Following is the **ViewModel** for the current use case that I explained above:

[Hide](#) [Shrink](#) ▲ [Copy Code](#)

```
namespace JIRA.ViewModels
{
    public class RegisterViewModel
    {
        public int UserID { get; set; }

        [Required(ErrorMessage = "First Name is required")]
        public string FirstName { get; set; }

        [Required(ErrorMessage = "Last Name is Required")]
        public string LastName { get; set; }

        [Required(ErrorMessage = "Username is Required")]
        [RegularExpression(@"^[a-zA-Z0-9]+$",
            ErrorMessage = "user name must be combination of letters and numbers only.")]
        [Remote("UsernameExists", "Account",
            HttpMethod="POST", ErrorMessage="User name already registered.")]
        public string UserName { get; set; }

        [Required(ErrorMessage = "Password is Required")]
        public string Password { get; set; }

        [Required(ErrorMessage = "Password is Required")]
```

```

[System.Web.Mvc.Compare("Password",
ErrorMessage="Both Password fields must match.")]
public string ConfirmPassword { get; set; }

[Required(ErrorMessage = "Email Address is required")]
[EmailAddress(ErrorMessage = "Invalid Email Address")]
[Remote("EmailExists", "Account",
HttpMethod = "POST", ErrorMessage = "Email address already registered.")]
public string Email { get; set; }
}
}

```

Now, we will strongly type our View with the **RegisterViewModel** type which only contains properties that are related with the Register View:

Hide Shrink ▲ Copy Code

```

@model JIRA.ViewModels.RegisterViewModel

using (Html.BeginForm("SignUp", "Account",
FormMethod.Post, new { @class = "form-inline", role = "form" }))
{
    @Html.AntiForgeryToken()

    <div class="row">
        <div class="span8 offset5 aui-page-panel">

            <div>
                <h2>Sign Up</h2>
            </div>
            <fieldset style="margin-bottom: 40px;">

                <legend style="border-bottom: 1px solid gray;
font-size: 16px;">Basic Information</legend>

                <table width="100%" border="0"
cellspacing="0" cellpadding="0">
                    <tr id="tr_basic">
                        <td style="vertical-align: top;" width>
                            <div id="basicinfo"
                                style="width: 100%">
                                    <div style="height: auto !important;
overflow-y: visible;">
                                        <table cellpadding="3">

                                            <tbody>
                                                <tr>

```

```

        <td width="150">
            @Html.LabelFor
            (model => model.FirstName,
            new { @class = "sr-only" })
        </td>
        <td>
            @Html.TextBoxFor
            (model => model.FirstName,
            new { @class =
            "form-control input-sm" })
            @Html.MyValidationMessageFor
            (model => model.FirstName)
        </td>
    </tr>
    <tr>
        <td>
            @Html.LabelFor
            (model => model.LastName)
        </td>
        <td>
            @Html.TextBoxFor
            (model => model.LastName,
            new { @class =
            "input-xsmall" })
            @Html.MyValidationMessageFor
            (model => model.LastName)
        </td>
    </tr>
    @*<tr>
        <td>
            @Html.LabelFor(model => model.Email)
        </td>
        <td>
            @Html.TextBoxFor(model => model.Email,
            new { @class = "required" })
            @Html.MyValidationMessageFor
            (model => model.Email)
        </td>
    </tr>*@

    <tr>
    </tr>

    </tbody>
</table>

</div>
</td>

```

```

    </tr>

</table>

<legend style="border-bottom: 1px solid gray;
font-size: 16px;">Account Information</legend>
<table cellpadding="5">
    <tr>
        <td width="150">
            @Html.LabelFor(model => model.UserName)
        </td>
        <td>
            @Html.TextBoxFor(model => model.UserName)
            @Html.MyValidationMessageFor(model => model.UserName)
        </td>
        <td id="tdValidate">
            </td>

    </tr>
    <tr>
        <td>
            @Html.LabelFor(model => model.Password)
        </td>
        <td>
            @Html.PasswordFor(model => model.Password)
            @Html.MyValidationMessageFor(model => model.Password)
        </td>
    </tr>
    <tr>
        <td>
            @Html.LabelFor(m => m.ConfirmPassword,
            new { @class = "control-label" })
        </td>
        <td>
            @Html.PasswordFor(model => model.ConfirmPassword)
            @Html.MyValidationMessageFor(model => model.ConfirmPassword)

        </td>
    </tr>
    <tr>
        <td>
            @Html.LabelFor(m => m.Email,
            new { @class = "control-label" })
        </td>

```

```

        <td>
            @Html.TextBoxFor(model => model.Email)
            @Html.MyValidationMessageFor(model => model.Email)
        </td>

    </tr>
    <tr>
    <td>
        <p>
            <div class="control-group">
                <div class="controls">
                    <input id="btnRegister"

                        type="submit"

                        class="btn btn-primary"

                        value="Register" />
                </div>
            </div>
        </p>
    </td>
    <td></td>
    </tr>
</table>

</fieldset>
</div>
</div>
}

```

and our action would look like:

Hide Copy Code

```

[HttpPost]
[AllowAnonymous]
public ActionResult SignUp(RegisterViewModel registerVM)
{
    if (ModelState.IsValid)
    {
        // save to database
    }
    return View(registerVM);
}

```

Our service method takes object of type **User** as input which is our Domain or Entity Model so we will have to convert **RegisterViewModel** object to **User** object, a quick and dirty way is to create an instance of type **User** and map it to **RegisterViewModel** before calling service method.

Here it is:

[Hide](#) [Copy Code](#)

```
[HttpPost]
[AllowAnonymous]
public ActionResult SignUp(RegisterViewModel registerVM)
{
    if (ModelState.IsValid)
    {
        User user = new User
        {
            FirstName = registerVM.FirstName,
            LastName = registerVM.LastName,
            UserName = registerVM.UserName,
            Email = registerVM.Email,
            Password = registerVM.Password
        };

        var result = authenticateService.RegisterUser(user);
    }
}
```

The above code will obviously work but it will cause code redundancy and in result it will be violation of **DRY** principle, because in future there is a possibility of mapping **ViewModel** instance to **Model** instance or vice versa and we will end up writing the same code again and again at different places where it is needed which is not good.

## Implicit Operator in C#

Here comes in the **implicit operator feature** provide by C#, we will write our operator for **RegisterViewModel** and **User** class so that they can be implicitly converted to each other wherever needed. We will have to modify the implementation of **RegisterViewModel** for this:

We will have to add these two operators in the **RegisterViewModel** class:

[Hide](#) [Copy Code](#)

```
public static implicit operator RegisterViewModel(User user)
{
    return new RegisterViewModel
    {
        UserID = user.UserID,
        FirstName = user.FirstName,
        UserName = user.UserName,
        Password = user.Password,
        ConfirmPassword = user.Password,
        Email = user.Email
    };
}
```



```
}

public static implicit operator User(RegisterViewModel vm)
{
    return new User
    {
        FirstName = vm.FirstName,
        LastName = vm.LastName,
        UserName = vm.UserName,
        Email = vm.Email,
        Password = vm.Password
    };
}
```

So we have written the implicit conversion between these two types at one place and we will be reusing it everywhere we need.

Yes, I mean that, now these two types can be implicitly convertible / cast-able into each other.

My action would look like this now:

[Hide](#) [Copy Code](#)

```
[HttpPost]
[AllowAnonymous]
public ActionResult SignUp(RegisterViewModel registerVM)
{
    if (ModelState.IsValid)
    {
        var result = authenticateService.RegisterUser(registerVM); // implicit
                                // conversion from RegisterViewModel to User Model

        RegisterViewModel vm = result; // see implicit conversion
                                // from User model to RegisterViewModel

        return View(vm);
    }
    return View(registerVM);
}
```

## Summary

By implementing implicit operator for **Model** to **ViewModel** mapping and vice versa, we can see that our action is cleaner now, as conversion is being moved to a central place due to which our code is reusable as well, and we are trying to follow **DRY** principle to some extent.

You can read more about [implicit operator at MSDN here](#).

## License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOl\)](#)

## Share

## About the Author



### Ehsan Sajjad



Software Developer  
Pakistan 🇵🇰

Ehsan Sajjad is a Microsoft Certified Professional, Microsoft Certified C# Specialist and he is also among the top users on [StackOverflow](#) from Pakistan with 50k+ reputation at time of writing this and counting.

He is a passionate software developer with around 5 years of professional experience in Microsoft Technologies both web and desktop applications and always open to learn new th...

[show more](#)

## Comments and Discussions

You must [Sign In](#) to use this message board.



Spacing

Relaxed ▼

Layout

Normal ▼




























Per page

25 ▼

Update

First Prev Next

<b>How is this better than...</b>	<b>#realJSOP</b>	<b>18-Oct-17 3:53</b>
Re: How is this better than...	Ehsan Sajjad	26-Dec-17 6:19
<b>my vote of 5</b>	<b>Ellix4u</b>	<b>22-Aug-17 5:19</b>
Re: my vote of 5	Ehsan Sajjad	22-Aug-17 23:51
<b>what about nested entities inside viewmodel</b>	<b>longnights</b>	<b>27-Feb-17 17:45</b>
Re: what about nested entities inside viewmodel	Ehsan Sajjad	27-Feb-17 19:43
<b>authenticateService.RegisterUser?</b>	<b>Member 3447522</b>	<b>6-Dec-16 7:28</b>
Re: authenticateService.RegisterUser?	Ehsan Sajjad	6-Dec-16 10:13
<b>How about Update?</b>	<b>andy_chou</b>	<b>22-Mar-16 6:34</b>
Re: How about Update?	Ehsan Sajjad	23-Mar-16 2:03
Re: How about Update?	andy_chou	23-Mar-16 5:59
Re: How about Update?	Ehsan Sajjad	23-Mar-16 6:06
<b>My vote of 5</b>	<b>Member 12352829</b>	<b>25-Feb-16 19:22</b>
Re: My vote of 5	Ehsan Sajjad	26-Feb-16 7:19
Re: My vote of 5	Member 3447522	6-Dec-16 7:07
<b>Data loss? EF error?</b>	<b>Alan MIERS</b>	<b>30-Oct-15 7:05</b>

 <a href="#">Re: Data loss? EF error?</a> 	 Ehsan Sajjad	30-Oct-15 7:20
 <b>My vote of 2</b> 	 shatl	<b>29-Oct-15 5:41</b>
 <a href="#">Re: My vote of 2</a> 	 Ehsan Sajjad	29-Oct-15 6:29
 <b>My vote of 5</b> 	 George McCormick	<b>29-Oct-15 5:15</b>
 <a href="#">Re: My vote of 5</a> 	 Ehsan Sajjad	29-Oct-15 6:27
 <a href="#">Re: My vote of 5</a> 	 George McCormick	29-Oct-15 6:32
 <b>How did I not know about this?</b> 	 Jim Harte	<b>29-Oct-15 0:51</b>
 <a href="#">Re: How did I not know about this?</a> 	 Ehsan Sajjad	29-Oct-15 6:40
 <b>My vote of 4</b> 	 Gaston Verelst	<b>28-Oct-15 21:01</b>

Last Visit: 16-Mar-20 6:05   Last Update: 18-Mar-20 7:23

[Refresh](#)[1](#) [2](#) [Next »](#)
[General](#)
[News](#)
[Suggestion](#)
[Question](#)
[Bug](#)
[Answer](#)
[Joke](#)
[Praise](#)
[Rant](#)
[Admin](#)

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.

[Permalink](#)[Advertise](#)[Privacy](#)[Cookies](#)[Terms of Use](#)Layout: [fixed](#) | [fluid](#)Article Copyright 2015 by Ehsan Sajjad  
Everything else Copyright © [CodeProject](#), 1999-2020

Web05 2.8.200307.1