

## Login Form with Database connection example in ASP.Net MVC (/Articles/Login-Form-with-Database-connection-example-in-ASPNet-MVC.aspx)

21 Nov 2017 17 Jul 2018 Mudassar Khan (/Authors/Mudassar-Khan.aspx) 0 Comments 51012 Views  
 ASP.Net (/Categories/ASP.Net.aspx) Entity Framework (/Categories/Entity-Framework.aspx) MVC (/Categories/MVC.aspx)

Here Mudassar Ahmed Khan has explained with an example, how to implement simple User Login form using Database connection in ASP.Net MVC Razor. The User login form will be connected to Database using Custom Forms Authentication and Entity Framework in ASP.Net MVC Razor.

Download

Download Free Files API (<https://www.aspsnippets.com/Redirect.aspx?AdId=10567&RedirectUrl=https%3A%2F%2Fwww.e-iceblue.com%2FIntroduce%2Fspire-office-for-net-free.h>)

In this article I will explain with an example, how to implement simple User Login form using Database connection in ASP.Net MVC Razor. The User login form will be connected to Database using Custom Forms Authentication and Entity Framework in ASP.Net MVC Razor.



**Note:** This is the third article from the series, in my previous articles I have explained [Simple User Registration Form with Entity Framework Database in ASP.Net MVC](https://www.aspsnippets.com/Articles/Simple-User-Registration-Form-with-Entity-Framework-Database-in-ASPNet-MVC.aspx) (<https://www.aspsnippets.com/Articles/Simple-User-Registration-Form-with-Entity-Framework-Database-in-ASPNet-MVC.aspx>) and [ASP.Net MVC: Send user Confirmation email after Registration with Activation Link](https://www.aspsnippets.com/Articles/ASPNet-MVC-Send-user-Confirmation-email-after-Registration-with-Activation-Link.aspx) (<https://www.aspsnippets.com/Articles/ASPNet-MVC-Send-user-Confirmation-email-after-Registration-with-Activation-Link.aspx>).

### Configuring Bundles and enabling Client Side Validation

The User Login Form validation will be performed on Client Side using Model Data Annotations and jQuery.

Please refer the following article for complete information on how to configure Bundles and enable Client Side validation in ASP.Net MVC project.

[Using Bundles \(ScriptBundle\) in ASP.Net MVC Razor](https://www.aspsnippets.com/Articles/Using-Bundles-ScriptBundle-in-ASPNet-MVC-Razor.aspx) (<https://www.aspsnippets.com/Articles/Using-Bundles-ScriptBundle-in-ASPNet-MVC-Razor.aspx>).



**Note:** By default the validation done using Data Annotation attributes is Server Side. And hence to make it work Client Side, the Client Side validation must be enabled.

### Database

I am making use of the same database table [Users](https://www.aspsnippets.com/Articles/Simple-User-Registration-Form-with-Entity-Framework-Database-in-ASPNet-MVC.aspx) which was used in the article [Simple User Registration Form with Entity Framework Database in ASP.Net MVC](https://www.aspsnippets.com/Articles/Simple-User-Registration-Form-with-Entity-Framework-Database-in-ASPNet-MVC.aspx) (<https://www.aspsnippets.com/Articles/Simple-User-Registration-Form-with-Entity-Framework-Database-in-ASPNet-MVC.aspx>).

LENEVO-THINK\SQL...inDB - dbo.Users			
Column Name	Data Type	Allow Nulls	
UserId	int	<input type="checkbox"/>	
Username	nvarchar(20)	<input type="checkbox"/>	
Password	nvarchar(20)	<input type="checkbox"/>	
Email	nvarchar(30)	<input type="checkbox"/>	
CreatedDate	datetime	<input type="checkbox"/>	
LastLoginDate	datetime	<input checked="" type="checkbox"/>	



**Note:** You can download the database table SQL by clicking the download link below.

[Download SQL file](https://www.aspsnippets.com/DownloadFile.aspx?File=LoginDB_MVC.sql) ([https://www.aspsnippets.com/DownloadFile.aspx?File=LoginDB\\_MVC.sql](https://www.aspsnippets.com/DownloadFile.aspx?File=LoginDB_MVC.sql))

### Stored Procedure to Validate the User Credentials

The following stored procedure is used to validate the user credentials, this stored procedure first checks whether the username and password are correct else returns -1. If the username and password are correct but the user has not been activated then the code returned is -2.

```

@Username NVARCHAR(20),
@Password NVARCHAR(20)
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @UserId INT, @LastLoginDate DATETIME

    SELECT @UserId = UserId, @LastLoginDate = LastLoginDate
    FROM Users WHERE Username = @Username AND [Password] = @Password

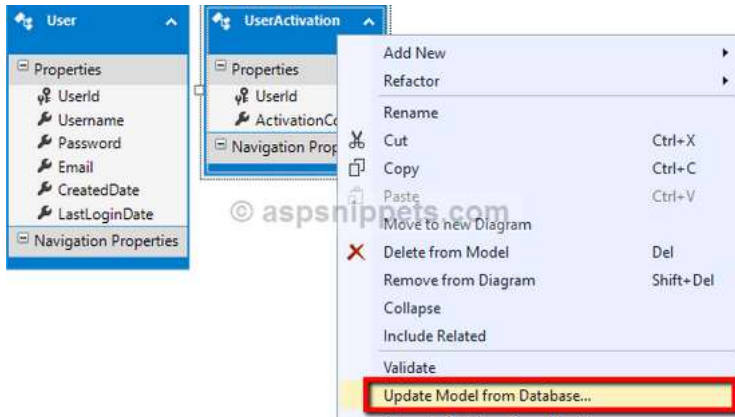
    IF @UserId IS NOT NULL
    BEGIN
        IF NOT EXISTS(SELECT UserId FROM UserActivation WHERE UserId = @UserId)
        BEGIN
            UPDATE Users
            SET LastLoginDate = GETDATE()
            WHERE UserId = @UserId
            SELECT @UserId [UserId] -- User Valid
        END
    ELSE
    BEGIN
        SELECT -2 -- User not activated.
    END
    ELSE
    BEGIN
        SELECT -1 -- User invalid.
    END
END
END

```

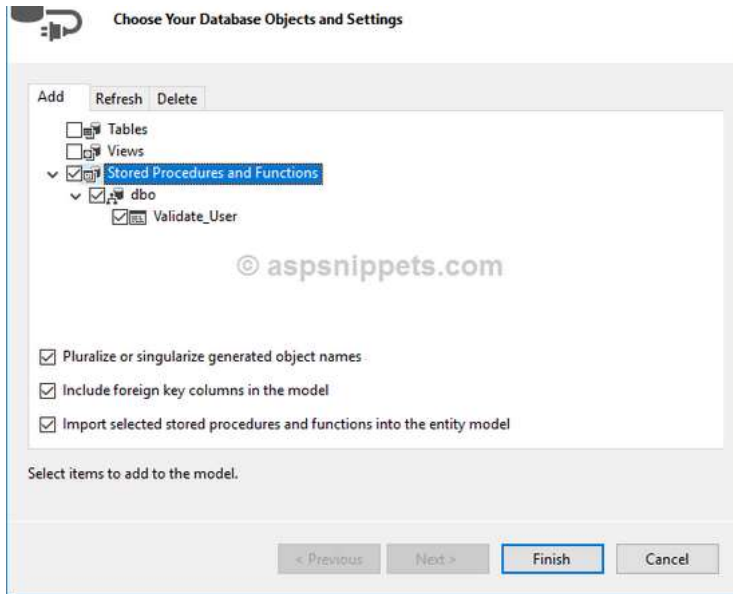
### **Adding new Stored Procedure to the Entity Framework Data Model**

The Entity Framework has been already configured in the [User Registration article \(https://www.aspsnippets.com/Articles/Simple-User-Registration-Form-with-Entity-Framework-Database-in-ASPNet-MVC.aspx\)](https://www.aspsnippets.com/Articles/Simple-User-Registration-Form-with-Entity-Framework-Database-in-ASPNet-MVC.aspx) and hence we will continue with further step i.e. adding the new Stored Procedure (discussed earlier) to the existing Entity Framework Data Model.

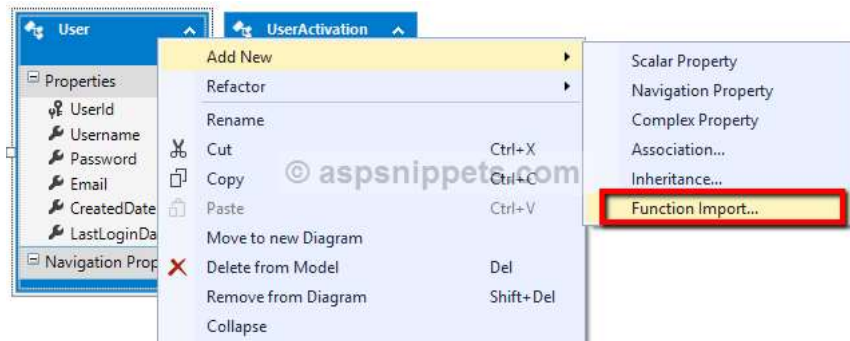
To do so, open the [User Data Model](#) and then Right click on the [User Table](#) and click on the [Update Model from Database](#) option from the Context menu.



The above action will open the [Update Wizard](#) dialog window, where you will need to select the newly added Stored Procedure and click Finish button.



Once the Stored Procedure is added you will again need to Right click on the User Table and click on Add New option and then Function Import option from the Context menu.



The above action will open the Add Function Import Dialog window. Here you will need to

1. Function Import Name: Specify the name of the method which will be used to execute the Stored Procedure.
2. Stored Procedure / Function Name: Select the Stored Procedure / Function to be imported.
3. Returns a Collection Of: The Stored Procedure used in this article returns a Scalar value and hence the same is selected.

Finally once all the above is done, simply click the OK button.

Function Import Name:

☐ Function Import is composable


Stored Procedure / Function Name:

Returns a Collection Of

☐ None

☒ Scalars:

☐ Complex:

☐ Entities:  

Stored Procedure / Function Column Information

Name	EDM Type	Db Type	Nullable	MaxLength	Precision	Scale
UserId	Int32	int	true			

### Model

The Model class `User.cs` remains the same as the [User Registration article \(https://www.aspsnippets.com/Articles/Simple-User-Registration-Form-with-Entity-Framework-Dat-abase-in-ASPNet-MVC.aspx\)](https://www.aspsnippets.com/Articles/Simple-User-Registration-Form-with-Entity-Framework-Dat-abase-in-ASPNet-MVC.aspx) except one change i.e. a new property **RememberMe** has been added.

```
namespace User_Login_MVC
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;

    public partial class User
    {
        public int UserId { get; set; }

        [Required(ErrorMessage = "Required.")]
        public string Username { get; set; }

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

        [Required(ErrorMessage = "Required.")]
        [Compare("Password", ErrorMessage = "Passwords do not match.")]
        public string ConfirmPassword { get; set; }

        [Required(ErrorMessage = "Required.")]
        [EmailAddress(ErrorMessage = "Invalid email address.")]
        public string Email { get; set; }

        public System.DateTime CreatedDate { get; set; }

        public Nullable<System.DateTime> LastLoginDate { get; set; }

        public bool RememberMe { get; set; }
    }
}
```



les:

[ASP.Net MVC: Client Side validations using Data Annotation attributes and jQuery \(https://www.aspsnippets.com/Articles/ASPNet-MVC-Client-Side-validations-using-Data-Annotation-attributes-and-jQuery.aspx\)](https://www.aspsnippets.com/Articles/ASPNet-MVC-Client-Side-validations-using-Data-Annotation-attributes-and-jQuery.aspx)

[Client Side Password and Confirm Password validation in ASP.Net MVC using Data Annotations and jQuery \(https://www.aspsnippets.com/Articles/Client-Side-Password-and-Confirm-Password-validation-in-ASPNet-MVC-using-Data-Annotations-and-jQuery.aspx\)](https://www.aspsnippets.com/Articles/Client-Side-Password-and-Confirm-Password-validation-in-ASPNet-MVC-using-Data-Annotations-and-jQuery.aspx)

[ASP.Net MVC: Client Side Email Validation using Data Annotation attributes and jQuery \(https://www.aspsnippets.com/Articles/ASPNet-MVC-Client-Side-Email-Validation-using-Data-Annotation-attributes-and-jQuery.aspx\)](https://www.aspsnippets.com/Articles/ASPNet-MVC-Client-Side-Email-Validation-using-Data-Annotation-attributes-and-jQuery.aspx)

## Namespaces

You will need to import the following namespace.

```
using System.Web.Security;
```

## Controller

The Controller consists of four Action methods.

### Action method for handling GET operation for Login

Inside this Action method, simply the View is returned. This Action method is decorated with AllowAnonymous Data Annotation which signifies Form Based authentication that this method can be accessed without authentication.

### Action method for handling GET operation for Profile

Inside this Action method, simply the View is returned. This Action method is decorated with Authorize Data Annotation which signifies Form Based authentication that this method requires authentication to be accessed.

### Action method for handling POST operation for Login

Inside this Action method, the ValidateUser method is called which executes the Stored Procedure that validates the User's credentials.

The status returned from the Stored Procedure is captured and if the value is not -1 (Username or password incorrect) or -2 (Account not activated) then the user is redirected to the Profile View after setting the Forms Authentication Cookie.

For the status -1 and -2, the message is displayed to the user using ViewBag object.

### Action method for handling POST operation for Logout

Inside this Action method, the Signout method of Forms Authentication is called which clears the Forms Authentication Cookie and the user is redirected to the Index View.

```
public class HomeController : Controller
{
    [AllowAnonymous]
    public ActionResult Index()
    {
        return View();
    }

    [Authorize]
    public ActionResult Profile()
    {
        return View();
    }

    [HttpPost]
    [AllowAnonymous]
    public ActionResult Index(User user)
    {
        UsersEntities usersEntities = new UsersEntities();
        int? userId = usersEntities.ValidateUser(user.Username, user.Password).FirstOrDefault();

        string message = string.Empty;
        switch (userId.Value)
        {

```

```

        case -2:
            message = "Account has not been activated.";
            break;
        default:
            FormsAuthentication.SetAuthCookie(user.Username, user.RememberMe);
            return RedirectToAction("Profile");
    }

    ViewBag.Message = message;
    return View(user);
}

[HttpPost]
[Authorize]
public ActionResult Logout()
{
    FormsAuthentication.SignOut();
    return RedirectToAction("Index");
}
}

```

## Views

Index

Inside the View, in the very first line the User Model class is declared as Model for the View.

The View consists of an HTML Form which has been created using the Html.BeginForm method with the following parameters.

ActionName – Name of the Action. In this case the name is Index.

ControllerName – Name of the Controller. In this case the name is Home.

FormMethod – It specifies the Form Method i.e. GET or POST. In this case it will be set to POST.

Inside the View, the following four HTML Helper functions are used:-

1. `Html.TextBoxFor` – Creating a `TextBox` for the Model property.
2. `Html.PasswordFor` – Creating a Password `TextBox` for the Model property.
3. `Html.ValidationMessageFor` – Displaying the Validation message for the property.
4. `Html.CheckBoxFor` – Creating a `CheckBox` for the Model property.

There is also Submit button which when clicked, the Form gets submitted.

The jQuery and the jQuery Validation script bundles are rendered at the end of the Model using the `Scripts.Render` function.

ViewBag's Message object is checked for NULL and if it is not NULL then the string message is displayed using JavaScript Alert Message Box.

```
@model User_Login_MVC.User

@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width"/>
    <title>Index</title>
    <style type="text/css">
        body {
            font-family: Arial;
            font-size: 10pt;
        }

        table {
            border: 1px solid #ccc;
            border-collapse: collapse;
        }
    </style>

```

```

        background-color: #ccc;
        color: #333;
        font-weight: bold;
    }

    table th, table td {
        padding: 5px;
        border: 1px solid #ccc;
    }

    .error {
        color: red;
    }
</style>
</head>
<body>
    @using (Html.BeginForm("Index", "Home", FormMethod.Post))
    {
        <table border="0" cellpadding="0" cellspacing="0">
            <tr>
                <th colspan="3">
                    Login
                </th>
            </tr>
            <tr>
                <td>
                    Username
                </td>
                <td>
                    @Html.TextBoxFor(m => m.Username)
                </td>
                <td>
                    @Html.ValidationMessageFor(m => m.Username, "", new { @class = "error" })
                </td>
            </tr>
            <tr>
                <td>
                    Password
                </td>
                <td>
                    @Html.PasswordFor(m => m.Password)
                </td>
                <td>
                    @Html.ValidationMessageFor(m => m.Password, "", new { @class = "error" })
                </td>
            </tr>
            <tr>
                <td>
                    Remember Me
                </td>
                <td>
                    @Html.CheckBoxFor(m => m.RememberMe)
                </td>
                <td>
                </td>
            </tr>
            <tr>
                <td></td>
                <td>
                    <input type="submit" value="Submit"/>
                </td>
                <td></td>
            </tr>
        </table>
    }

```

```

    if (@ViewBag.Message != null)
    {
        <script type="text/javascript">
            $(function () {
                alert("@ViewBag.Message")
            });
        </script>
    }
</body>
</html>

```

### Profile

The Profile View displays the name of the Current Logged in User and it also consists of an HTML Form with an HTML Anchor link for Logout functionality.

When the Logout link is clicked, the Form gets submitted and the Logout Action method gets called.

```

@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width"/>
    <title>Profile</title>
    <style type="text/css">
        body {
            font-family: Arial;
            font-size: 10pt;
        }
    </style>
</head>
<body>
    <div>
        Welcome
        <b>@HttpContext.Current.User.Identity.Name</b>
        <br/>
        <br/>
        @using (Html.BeginForm("Logout", "Home", FormMethod.Post))
        {
            <a href="javascript:;" onclick="document.forms[0].submit();">Logout</a>
        }
    </div>
</body>
</html>

```

### Web.Config Configuration

You will need to add the following configuration in the Web.Config file in the <system.web> section.

```

<authentication mode="Forms">
    <forms defaultUrl="/Home/Profile" loginUrl="/Home/Index" slidingExpiration="true" timeout="2880"></forms>
</authentication>

```

### Screenshots





## Downloads

[Download](#)

Download Free Files API ([https://www.aspsnippets.com/Redirect.aspx?AdId=10567&RedirectUrl=https%3A%2F%2Fwww.e-iceblue.com%2FIntroduce%2Fspire-office-for-net-free.html%](https://www.aspsnippets.com/Redirect.aspx?AdId=10567&RedirectUrl=https%3A%2F%2Fwww.e-iceblue.com%2FIntroduce%2Fspire-office-for-net-free.html%2F))



## Related Articles

### Save and Retrieve Files from SQL Server Database using ASP.Net (/Articles/Save-and-Retrieve-Files-from-SQL-Server-Database-using-ASPNet.aspx)

Here Mudassar Ahmed Khan has explained how to save file as Binary data inside SQL Server database and also how to retrieve them reconvert to their respective formats in ASP.Net using C# and VB.Net.



## Comments

No comments have been added to this article.



## Add Comments



You can add your comment about this article using the form below. Make sure you provide a valid email address else you won't be notified when the author replies to your comment

Please note that all comments are moderated and will be deleted if they are

- Not relevant to the article
- Spam
- Advertising campaigns or links to other sites
- Abusive content.

**Please do not post code, scripts or snippets.**

Name

Email

Comment

Security code:

15 + 4 =



 Add Comment

## What our readers say

“

**Mo ARKAM**


Hi Sir,

i learn lot of information.

Thank You.



”

© 2021 [www.aspsnippets.com](https://www.aspsnippets.com/) (<https://www.aspsnippets.com/>) All rights reserved | [Privacy Policy](https://aspsnippets.com/PrivacyPolicy.aspx) (<https://aspsnippets.com/PrivacyPolicy.aspx>) | Powered by  
Excelasoft Solutions (<http://www.excelasoft.com/>) 

 (<https://www.facebook.com/pages/ASPSnippets/306994206006035>)   
(<https://plus.google.com/110371172807820981480>)   
(<https://twitter.com/aspsnippets>)  (/Rss.ashx)