Simple User Registration Form with Entity Framework Database in ASP.Net MVC (/Articles/Simple-User-Registration-Form-with-Entity-Framework-Database-in-ASPNet-MVC.aspx)

22 Nov 2016 Mudassar Khan (/Authors/Mudassar-Khan.aspx) 6 Comments 100219 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 build a simple user registration form that will allow user register to the website in ASP.Net MVC Razor. The Registration Form will save (insert) data to database using Entity Framework.
User will fill up the registration form with details such as username, password, email address, etc. and these details will be saved in the database table.
The registration form will also make sure that duplicate username and email addresses are not saved by verifying whether username and email address must not exists in the table.

Download

In this article I will explain with an example, how to build a simple user registration form that will allow user register to the website in ASP.Net MVC Razor. The Registration Form will save (insert) data to database using Entity Framework.

User will fill up the registration form with details such as username, password, email address, etc. and these details will be saved in the database table.

The registration form will also make sure that duplicate username and email addresses are not saved by verifying whether username and email address must not exists in the table.

Configuring Bundles and enabling Client Side Validation

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

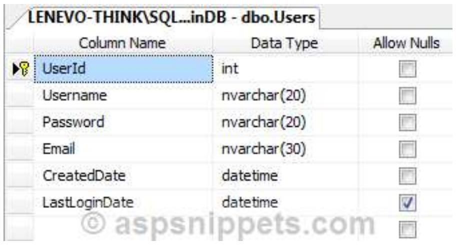
Please refer the following article for complete information on how to configure Bundles and enable Client Side validation in ASP.Net MVC project.

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

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

For this article I have created a new database named LoginDB which contains the following table named Users in it.



LENEVO-THINK\SQL...inDB - dbo.Users

Column Name	Data Type	Allow Nulls
UserId	int	☐
Username	nvarchar(20)	☐
Password	nvarchar(20)	☐
Email	nvarchar(30)	☐
CreatedDate	datetime	☐
LastLoginDate	datetime	☑
		☐

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)

Stored Procedure to insert the User details

The following stored procedure is used to insert the user details such as username, password and email address.

The stored procedure first checks whether the username supplied already exists, if yes then it will return negative 1 (-1) value.

Then the stored procedure checks whether the email address supplied already exists, if yes then it will return negative 2 (-2) value.

If both username and email address are valid then the record will be inserted and the auto-generated UserId will be returned by the stored procedure.

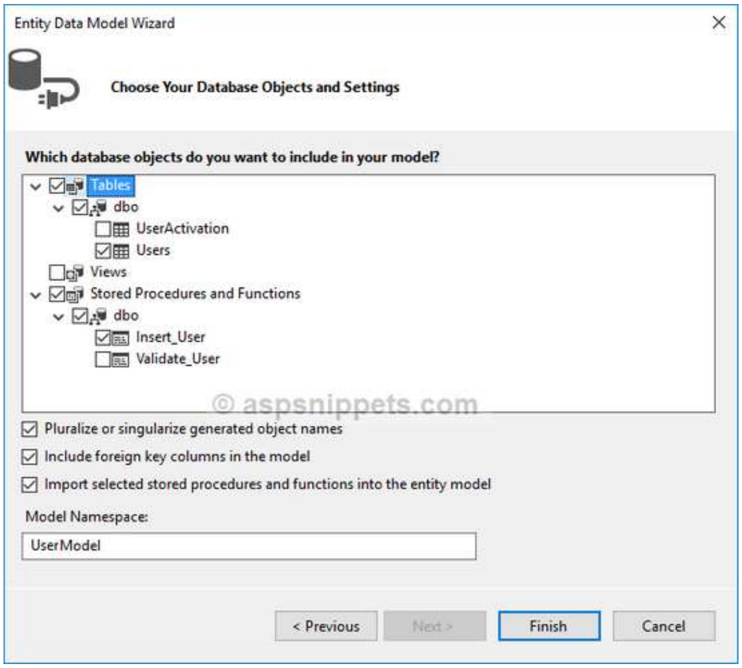
```
CREATE PROCEDURE [dbo].[Insert_User]
       @Username NVARCHAR(20),
AS
BEGIN
       SET NOCOUNT ON;
       IF EXISTS(SELECT UserId FROM Users WHERE Username = @Username)
       BEGIN
              SELECT -1 AS UserId -- Username exists.
       END
       ELSE IF EXISTS(SELECT UserId FROM Users WHERE Email = @Email)
       BEGIN
              SELECT -2 AS UserId -- Email exists.
       END
       ELSE
       BEGIN
              INSERT INTO [Users]
                        ([Username]
                        ,[Password]
                       ,[Email]
                        ,[CreatedDate])
              VALUES
                        (@Username
                        ,@Password
                        ,@Email
                        ,GETDATE())

              SELECT SCOPE_IDENTITY() AS UserId -- UserId
     END
END
```

Configuring Entity Framework

You will need to configure the Entity Framework in order to connect to the database.

> **Note**: The complete details of configuring and using Entity Framework in ASP.Net MVC are provided in my article ASP.Net MVC: Entity Framework Database First Approach example (https://www.aspsnippets.com/Articles/ASPNet-MVC-Entity-Framework-Database-First-Approach-example.aspx).

Once you reach the Database Objects selection step, you will need to select the Users table and the **Insert_User** Stored Procedure as shown below.



Which database objects do you want to include in your model?

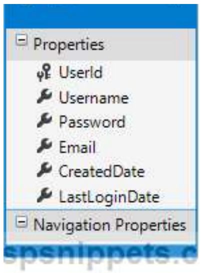
- ☑ Tables
  - ☑ dbo
    - □ UserActivation
    - ☑ Users
- □ Views
- ☑ Stored Procedures and Functions
  - ☑ dbo
    - ☑ Insert_User
    - □ Validate_User

☑ Pluralize or singularize generated object names

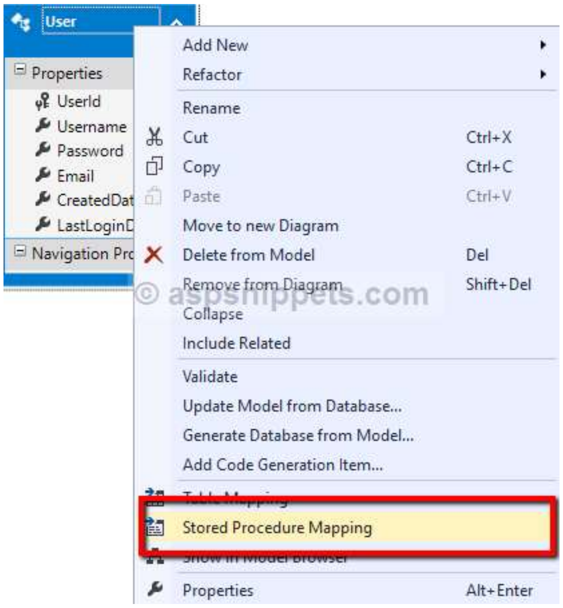
☑ Include foreign key columns in the model

☑ Import selected stored procedures and functions into the entity model

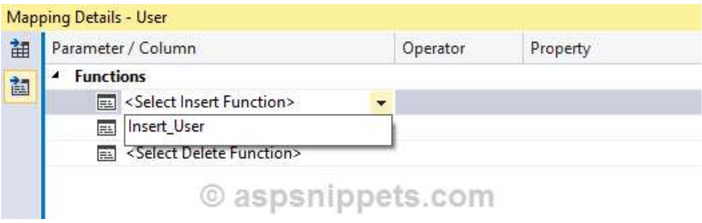
Model Namespace: UserModel



The next step is to map the **Insert_User** Stored Procedure with the Insert operation of Entity Framework. In order to do so, you will need to Right Click, the Entity Model and select Stored Procedure Mapping as shown below.

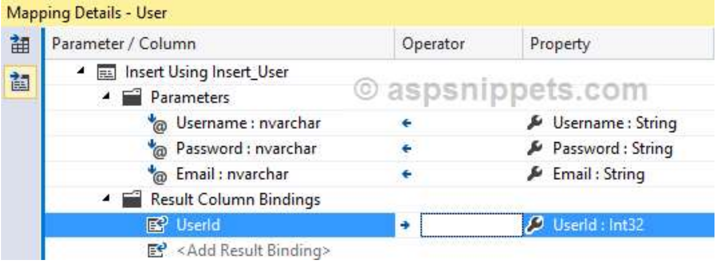


Once the Mapping Details window is open, you need to click the <Select Insert Function> Dropdown and select the **Insert_User** Stored Procedure.



Finally from the Result Column Bindings, click on Add Result Binding and type in the value UserId and then click on the next Cell in the Operator column and automatically it will be display the UserId property name in the corresponding Cell of the Property column.

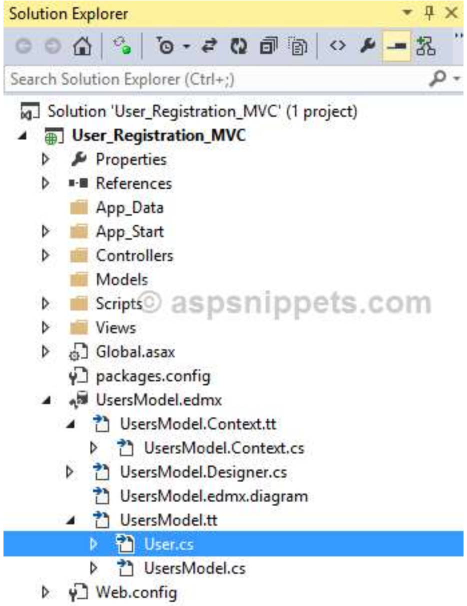
Note: The ALIAS for the returning Column in the **Insert_User** Stored Procedure is UserId and hence the same is used while mapping the returning value to the property.


Mapping Details - User

Parameter / Column	Operator	Property
Insert Using Insert_User		
Parameters		
Username : nvarchar	←	Username : String
Password : nvarchar	←	Password : String
Email : nvarchar	←	Email : String
Result Column Bindings		
UserId	→	UserId : Int32
<Add Result Binding>		

Model

There is no need of Model class for this project as we will be using the Entity Framework Model class which is automatically generated. You will find it in the Solution Explorer as shown below.



Solution 'User_Registration_MVC' (1 project)
User_Registration_MVC
Properties
References
App_Data
App_Start
Controllers
Models
Scripts
Views
Global.asax
packages.config
UsersModel.edmx
UsersModel.Context.tt
UsersModel.Context.cs
UsersModel.Designer.cs
UsersModel.edmx.diagram
UsersModel.tt
User.cs
UsersModel.cs
Web.config

Now you will need to open the Model class User.cs and it will have the following contents.

```
namespace User_Registration_MVC
{
    using System;
    using System.Collections.Generic;

    public partial class User
    {
        public int UserId { get; set; }
        public string Username { get; set; }
        public string Password { get; set; }
        public string Email { get; set; }
        public System.DateTime CreatedDate { get; set; }
        public Nullable<System.DateTime> LastLoginDate { get; set; }
    }
}
```

Now you will need to add the validation Data Annotations and also one additional property ConfirmPassword as shown below.

> **Note**: You will need to keep a copy of this class before you regenerate Entity Framework model or make changes to it as all your changes will be overwritten.

```
namespace User_Registration_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.")]
        [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; }
    }
}
```

Note: For explanation of the various Data Annotations used for Required, Email and Confirm Password validations, please refer my articles:

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)

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)

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)

Controller

The Controller consists of two Action methods.

Action method for handling GET operation

Inside this Action method, simply the View is returned.

Action method for handling POST operation

This action method handles the POST operation and when the form is submitted, the object of the User model class is sent to this method.

The received User Model class object is inserted into the database using the Entity Framework and the value returned from the **Insert_User** Stored Procedure is captured in the UserId property of the User Model class object.

As discussed earlier, the **Insert_User** Stored Procedure will return negative 1 (-1), negative 2 (-2) or UserId of the inserted record.

Based on the returned values, a string message is set in the ViewBag object which will be later on displayed in View using JavaScript Alert Message Box.

```
public class HomeController : Controller
{
    // GET: Registration
    public ActionResult Index()
    {
        return View();
    }

    [HttpPost]
    public ActionResult Index(User user)
    {
        UsersEntities usersEntities = new UsersEntities();
        usersEntities.Users.Add(user);
        usersEntities.SaveChanges();
        string message = string.Empty;
        switch (user.UserId)
        {
            case -1:
                message = "Username already exists.\\nPlease choose a different username.";
                break;
            case -2:
                message = "Supplied email address has already been used.";
                break;
                break;
        }
        ViewBag.Message = message;

        return View(user);
    }
}
```

View

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 three 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.

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.

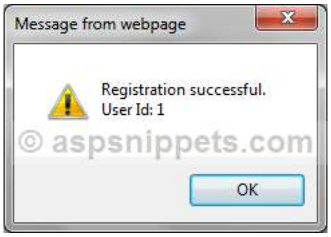
```
@model User_Registration_MVC.User

@{
    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;
        }

        table th {
            background-color: #F7F7F7;
            color: #333;
            font-weight: bold;
        }

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

        .error {
            color: red;
```

```
</head>
<body>
    @using (Html.BeginForm("Index", "Home", FormMethod.Post))
    {
        <table border="0" cellpadding="0" cellspacing="0">
            <tr>
                <th colspan="3">
                    Registration
                </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>
                    Confirm Password
                </td>
                <td>
                    @Html.PasswordFor(m => m.ConfirmPassword)
                </td>
                <td>
                    @Html.ValidationMessageFor(m => m.ConfirmPassword, "", new { @class = "error" })
                </td>
            </tr>
            <tr>
                <td>
                    Email
                </td>
                <td>
                    @Html.TextBoxFor(m => m.Email)
                </td>
                <td>
                    @Html.ValidationMessageFor(m => m.Email, "", new { @class = "error" })
                </td>
            </tr>
            <tr>
                <td></td>
                <td>
                    <input type="submit" value="Submit"/>
                </td>
                <td></td>
            </tr>
        </table>
    }
    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/jqueryval")
    @if (@ViewBag.Message != null)
```

```
            $(function () {
                alert("@ViewBag.Message")
            });
        </script>
    }
</body>
</html>
```

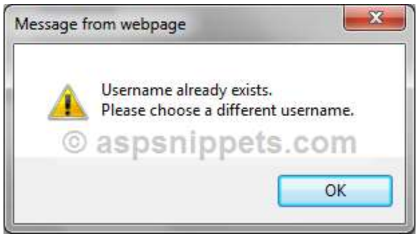
Screenshots

Message Box when registration is successful



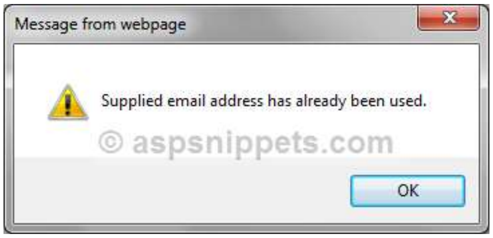
Registration successful.
User Id: 1

Message Box when username already exists



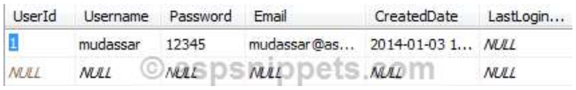
Username already exists.
Please choose a different username.

Message Box when email address already exists



Supplied email address has already been used.

User record inserted in table



UserId	Username	Password	Email	CreatedDate	LastLogin...
1	mudassar	12345	mudassar@as...	2014-01-03 1...	NULL
NULL	NULL	NULL	NULL	NULL	NULL

Downloads

Download

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 relavant to the article
- Spam
- Advertising campaigns or links to other sites
- Abusive content.

Please do not post code, scripts or snippets.

Name

Name

Email

Email

Comment

Comment

Security code: 13 + 1 = □

□ I declare, I accept the site's **Privacy Policy (/PrivacyPolicy.aspx)**.

Add Comment

What our readers say

Mo ARKAM
Hi Sir,
i learn lot of information.
Thank You.

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