4 Ways To Create Form In ASP.NET MVC

In this chapter, you will learn:

***1.** How to create Forms in ASP.NET MVC?*
***2.** 4 Different ways to create MVC Forms.*
***3.** How to access Forms data in controllers?*

Forms are very essential and basic thing that every programmer has to learn. In this tutorial, I will teach you 4 Different Ways to Create ASP.NET MVC Forms with ease.

i. **Forms** - Weakly Typed (Synchronous)
ii. **Forms** - Strongly Typed (Synchronous)
iii. **Forms** - Strongly Typed AJAX (Asynchronous)
iv. **Forms** – HTML, AJAX and JQUERY

Setup

Step 1: Create a New ASP.NET MVC Project MvcForms. Go to **File 🡺 New 🡺 Project**.
If you don’t know how to create a new mvc project then see this chapter.

Create New ASP.NET MVC 5 Project (https://www.completecsharptutorial.com/asp-net-mvc5/create-first-asp-net-mvc5-project.php)

Step 2: Create a model class `StudentModel.cs`. *Right-click* on **Model 🡺 Add 🡺 Class**.

Step 3: Open StudentModel.cs and add the following code in it.

```
namespace MvcForms.Models
{
    public class StudentModel
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public bool Addon { get; set; }
    }
}
```

1. FORMS – WEAKLY TYPED

This is the easiest and quickest way to create forms in MVC.

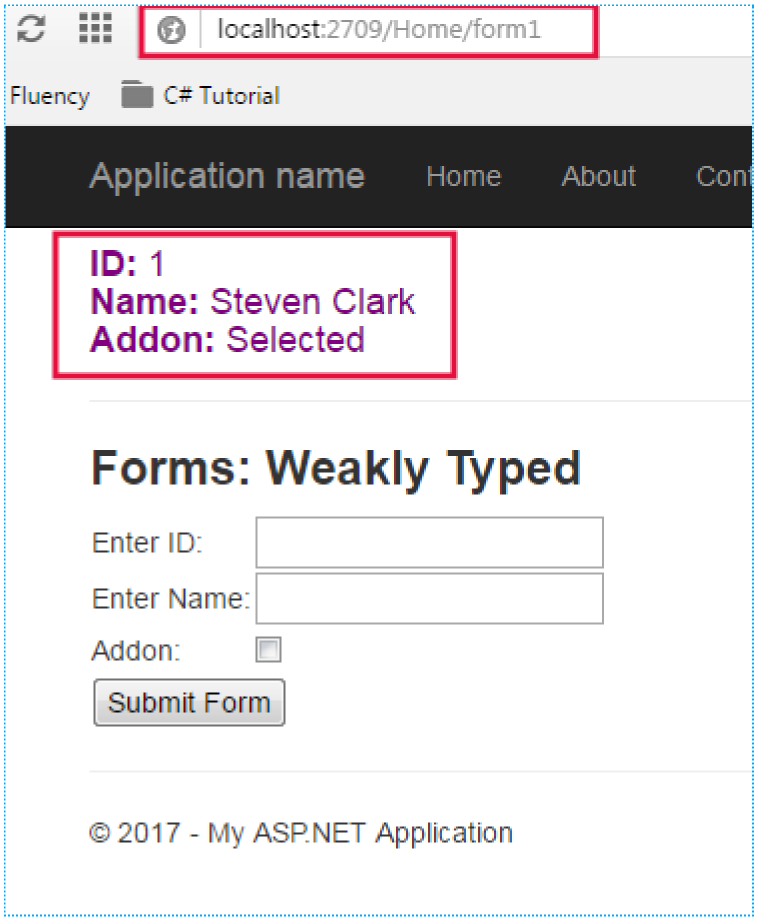
1. Go to **Views 🡺 Home 🡺** `Index.cshtml` and update it with following code.

```
<h4 style="color:purple">
    <b>ID:</b>    @ViewBag.ID <br />
    <b>Name:</b>  @ViewBag.Name <br />
    <b>Addon:</b> @ViewBag.Addon
</h4>
<hr />
<h3><b>Forms: Weakly Typed</b></h3>

<form action="form1" method="post">
    <table>
        <tr>
            <td>Enter ID: </td>
            <td><input type="text" name="txtId" /></td>
        </tr>
        <tr>
            <td>Enter Name: </td>
            <td><input type="text" name="txtName" /></td>
        </tr>
        <tr>
            <td>Addon: </td>
            <td><input type="checkbox" name="chkAddon" /></td>
        </tr>
        <tr>
            <td colspan="2"><input type="submit" value="Submit Form" /></td>
        </tr>
    </table>
</form>
```

2. Now, add an action method for this form in **`HomeController.cs`**

```
[HttpPost]
        public ActionResult form1(int txtId, string txtName, string chkAddon)
        {
            ViewBag.Id = txtId;
            ViewBag.Name = txtName;
            if (chkAddon != null)
                ViewBag.Addon = "Selected";
            else
                ViewBag.Addon = "Not Selected";

            return View("Index");
        }
```

Output:



ID: 1
Name: Steven Clark
Addon: Selected

Forms: Weakly Typed

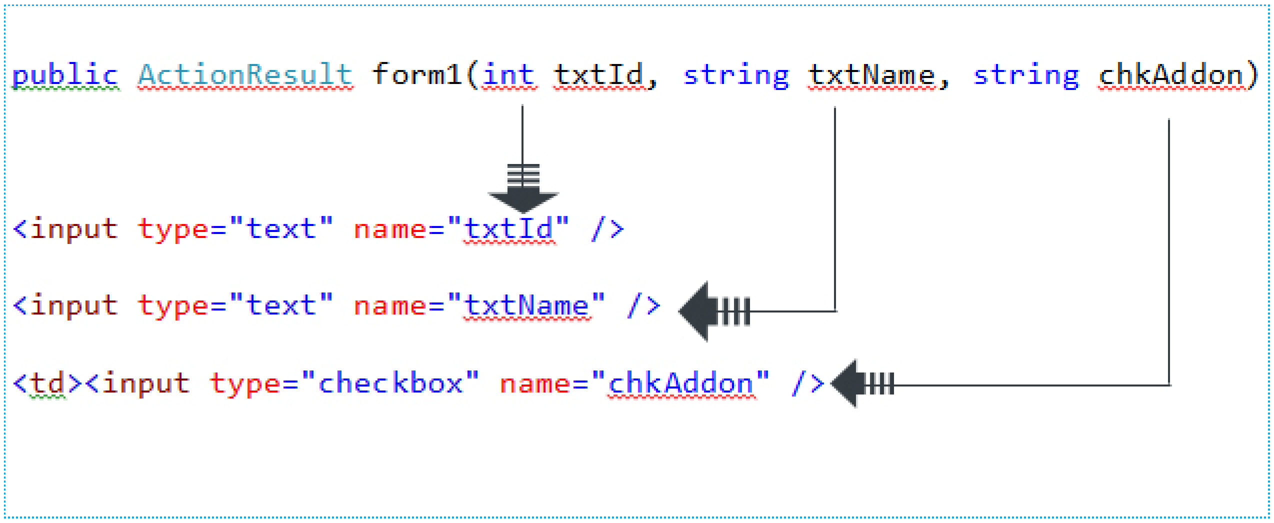
© 2017 - My ASP.NET Application

(https://www.completecsharptutorial.com/wp-content/uploads/2017/11/New-Picture-1-2.png)

Let's Understand

1. In the `<form action="form1" method=”post”>`, **form1** is Action Method that gets executed when forms sends data to **HomeController** using post method. In the next chapter, you will learn about post and get method in mvc.

2. In the `<input type="text" name="txtId" />`, the property `name=”txtId”` must be same as parameter name in form 1 action method.



```
public ActionResult form1(int txtId, string txtName, string chkAddon)

<input type="text" name="txtId" />
<input type="text" name="txtName" />
<td><input type="checkbox" name="chkAddon" />
```

(https://www.completecsharptutorial.com/wp-content/uploads/2017/11/weakly-typed-form.jpg)

3. CheckBox sends `"on"` if it selected otherwise sends `null`.

Advantage and Disadvantage of Weakly Typed Form

Advantage:

1. It is easy to create a form using Weakly Typed mechanism
2. Mostly used when you need to create a form with one or two input items.

Disadvantage:

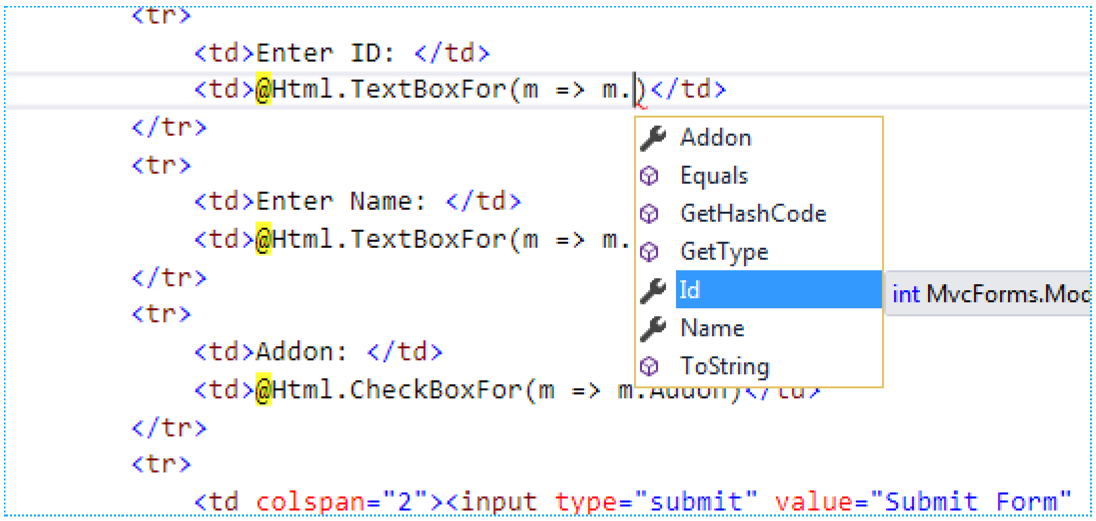
1. Because, it is not strongly typed so IntelliSense doesn't help you.
2. Have higher chance of getting exception and runtime error messages.

3. Very difficult to manage when forms have multiple input items and controls.

4. It is very clumsy when you need to add or remove some input items.

2. FORMS : STRONGLY TYPED

In this method, we send *objects (model)* instead of sending each item as parameter. It is easy to maintain because you don't need to remember each input item and *IntelliSense* will show you automatically the each item.



```
        <tr>
            <td>Enter ID: </td>
            <td>@Html.TextBoxFor(m => m.)</td>
        </tr>
        <tr>
            <td>Enter Name: </td>
            <td>@Html.TextBoxFor(m => m.
        </tr>
        <tr>
            <td>Addon: </td>
            <td>@Html.CheckBoxFor(m => m.Addon)</td>
        </tr>
        <tr>
            <td colspan="2"><input type="submit" value="Submit Form"
```

(https://www.completecsharptutorial.com/wp-content/uploads/2017/11/New-Picture-3-2.png)

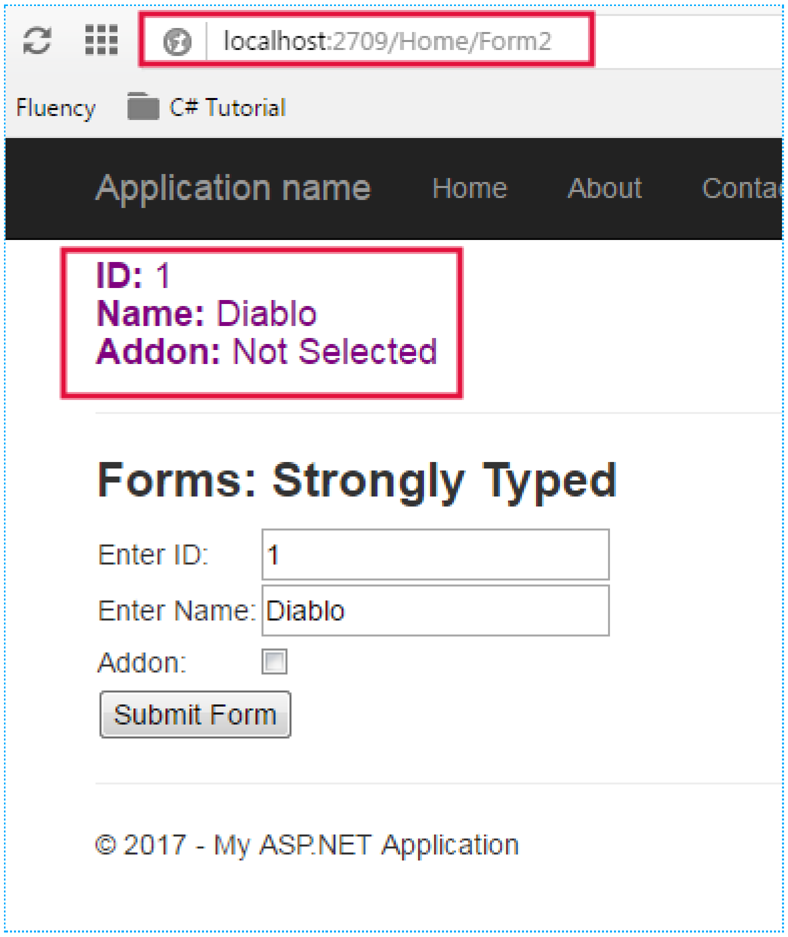
> **Step 1:** Go to `Index.cshtml` and update the code like this.

```
@model MvcForms.Models.StudentModel
<h4 style="color:purple">
    <b>ID:</b>    @ViewBag.ID <br />
    <b>Name:</b>  @ViewBag.Name <br />
    <b>Addon:</b> @ViewBag.Addon
</h4>
<hr />
<h3><b>Forms: Strongly Typed</b></h3>

@using (Html.BeginForm("Form2", "Home", FormMethod.Post))
{
    <table>
        <tr>
            <td>Enter ID: </td>
            <td>@Html.TextBoxFor(m => m.Id)</td>
        </tr>
        <tr>
            <td>Enter Name: </td>
            <td>@Html.TextBoxFor(m => m.Name)</td>
        </tr>
        <tr>
            <td>Addon: </td>
            <td>@Html.CheckBoxFor(m => m.Addon)</td>
        </tr>
        <tr>
            <td colspan="2"><input type="submit" value="Submit Form" /></td>
        </tr>
    </table>
```

Step 2: Go to `HomeController.cs` and add the following action method.

```
[HttpPost]
        public ActionResult Form2(Models.StudentModel sm)
        {
            ViewBag.Id = sm.Id;
            ViewBag.Name = sm.Name;
            if (sm.Addon == true)
                ViewBag.Addon = "Selected";
            else
                ViewBag.Addon = "Not Selected";

            return View("Index");
        }
```

Output



ID: 1
Name: Diablo
Addon: Not Selected

Forms: Strongly Typed

Enter ID: 1
Enter Name: Diablo
Addon:

© 2017 - My ASP.NET Application

(https://www.completecsharptutorial.com/wp-content/uploads/2017/11/New-Picture-4-2.png)

Let's understand it

In the Form

1. `@using (Html.BeginForm("Form2", "Home", FormMethod.Post))` is used for creating strongly typed forms. It has 3 parameters that denotes:

i. Form2: It is Action Method Name

ii. Home: It is Controller Name

iii. FormMethod.Post: It denotes that all the data will be submitted to controller using Post method.

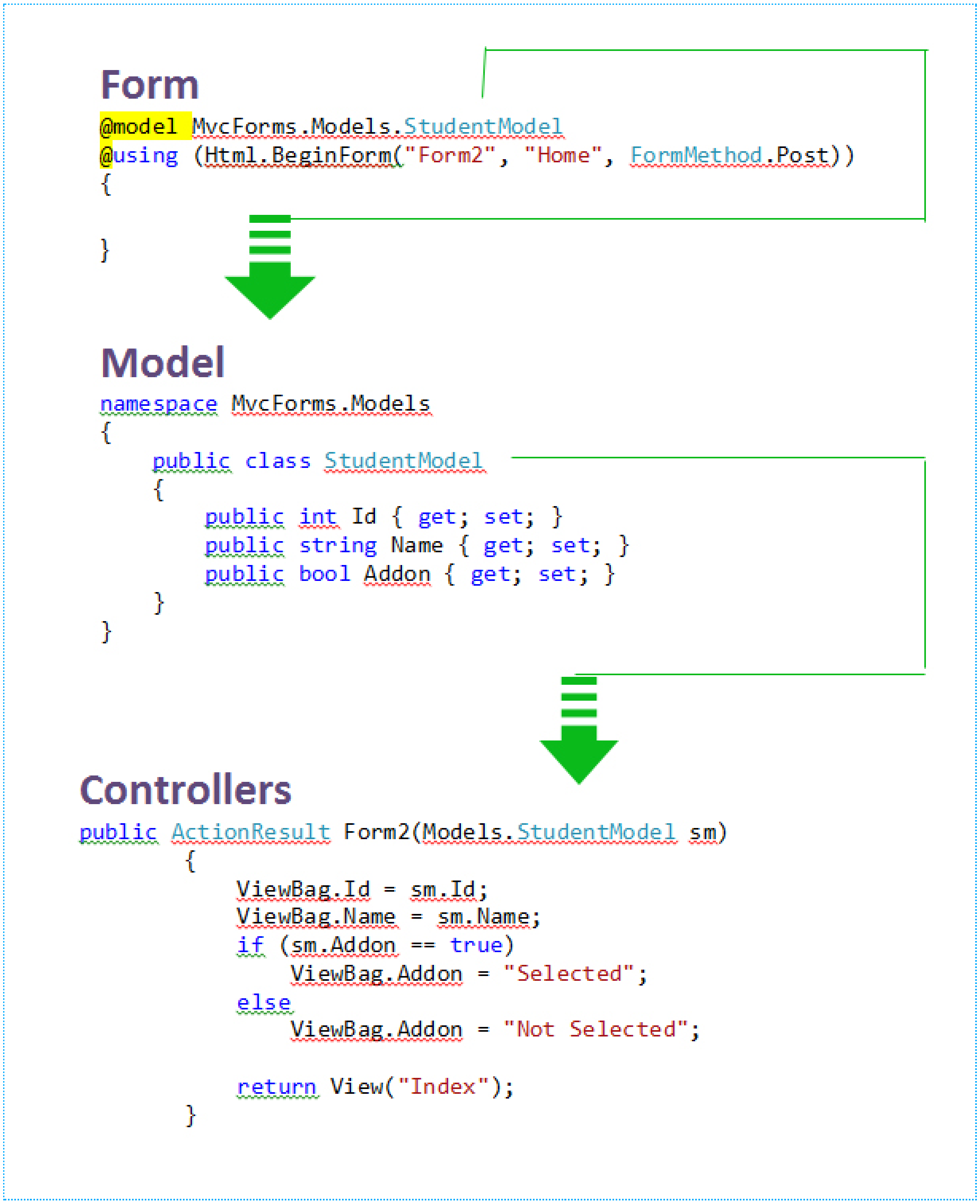
iv. `@Html.TextBoxFor(m => m.Id)` This is Html Helper. I have created textbox using mvc htmo helper and it is strongly bounded with Id.

v. `m => m.Id` is a lambda expression. It means that m is an instance of StudentModel.

In the **Form2** Action Method in `HomeController`

```
public ActionResult Form2(Models.StudentModel sm)
```

In this Action Method, I have passed the object of `StudentModel` class.



```
@model MvcForms.Models.StudentModel
@using (Html.BeginForm("Form2", "Home", FormMethod.Post))
{

}
```

```
namespace MvcForms.Models
{
    public class StudentModel
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public bool Addon { get; set; }
    }
}
```

```
public ActionResult Form2(Models.StudentModel sm)
        {
            ViewBag.Id = sm.Id;
            ViewBag.Name = sm.Name;
            if (sm.Addon == true)
                ViewBag.Addon = "Selected";
            else
                ViewBag.Addon = "Not Selected";

            return View("Index");
        }
```

(https://www.completecsharptutorial.com/wp-content/uploads/2017/11/strongly-typed-form.jpg)

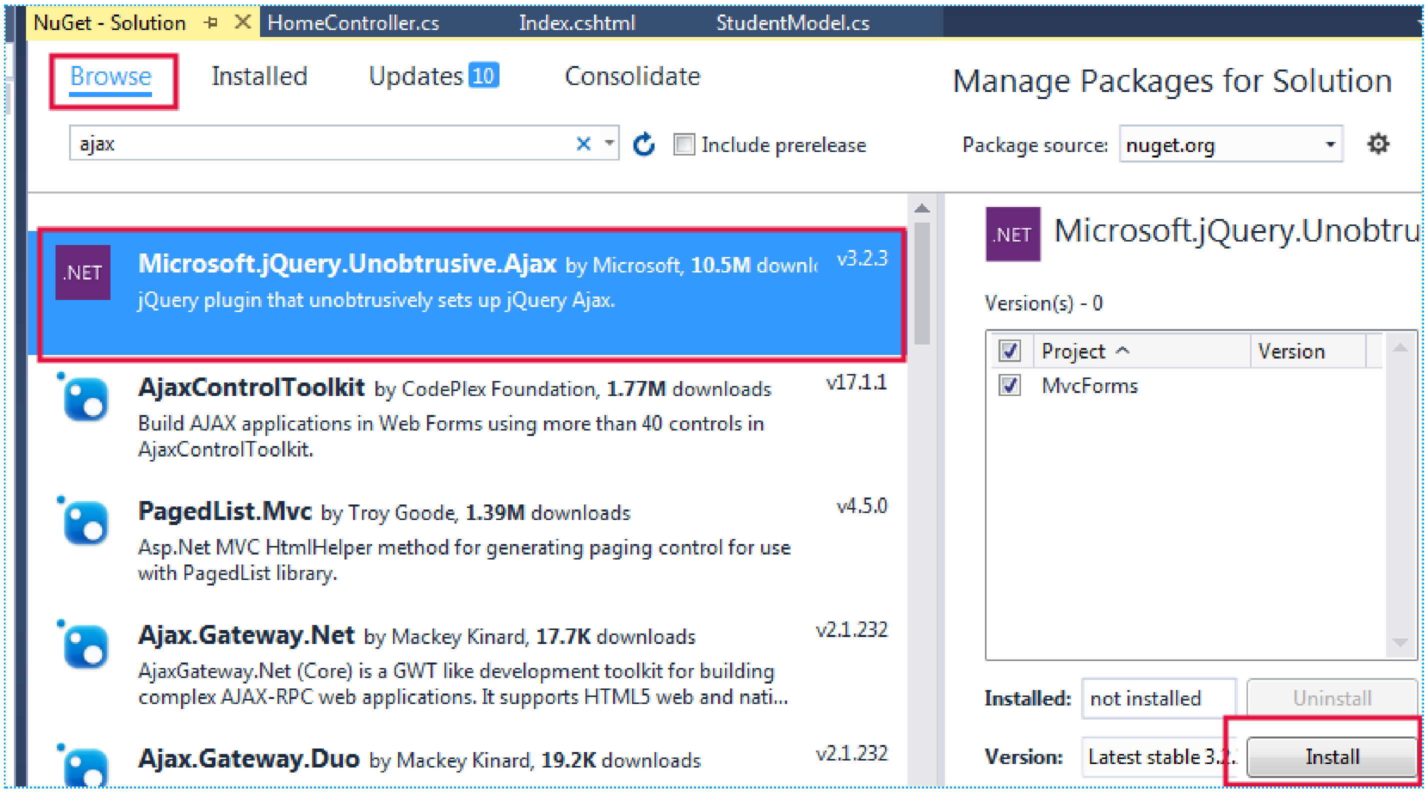
3. FORMS - STRONGLY TYPED AJAX (ASYNCHRONOUS)

Asynchronous AJAX form is a very magical way to submit data to the controller without happening page load. Asynchronous AJAX Forms simply post back the data to the controllers and update the only that part of the page, which has to display output.

To make this happen, we will use ***JQuery-Unobstrusive-AJAX***. This is a great feature which is launched in MVC 3. It helps you to create AJAX Form without writing bunch of javascript code. Before creating Asynchronous AJAX Form you need to add **JQuery-Unobstrusive-AJAX** in your project. Adding is very easy, and just follows the steps.

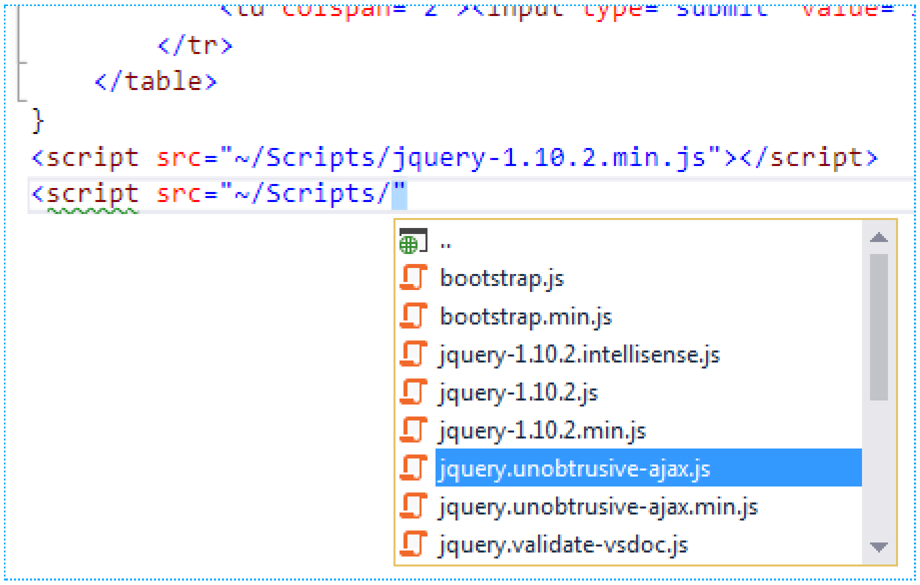
Adding JQuery-Unobstrusive-AJAX

> **Step 1: Right-click** on your Project name in Solution Explorer and click **Manage Nuget Packages…**
>
> **Step 2:** Go to **Browse** and search for **ajax**. Find and Install **Microsoft-JQuery-Unobstrusive-Ajax**.


NuGet - Solution
HomeController.cs
Index.cshtml
StudentModel.cs
Browse
Installed
Updates 10
Consolidate
Manage Packages for Solution
ajax
Include prerelease
Package source: nuget.org
Microsoft.jQuery.Unobtrusive.Ajax by Microsoft, 10.5M downl v3.2.3
jQuery plugin that unobtrusively sets up jQuery Ajax.
AjaxControlToolkit by CodePlex Foundation, 1.77M downloads v17.1.1
Build AJAX applications in Web Forms using more than 40 controls in AjaxControlToolkit.
PagedList.Mvc by Troy Goode, 1.39M downloads v4.5.0
Asp.Net MVC HtmlHelper method for generating paging control for use with PagedList library.
Ajax.Gateway.Net by Mackey Kinard, 17.7K downloads v2.1.232
AjaxGateway.Net (Core) is a GWT like development toolkit for building complex AJAX-RPC web applications. It supports HTML5 web and nati...
Ajax.Gateway.Duo by Mackey Kinard, 19.2K downloads v2.1.232
Microsoft.jQuery.Unobtru
Version(s) - 0
Project
Version
MvcForms
Installed: not installed
Uninstall
Version: Latest stable 3.2.
Install

(https://www.completecsharptutorial.com/wp-content/uploads/2017/11/New-Picture-7.png)

> **Step 3:** After installing this you can see it in Script folder.



```
        </tr>
    </table>
}
<script src="~/Scripts/jquery-1.10.2.min.js"></script>
<script src="~/Scripts/"
```

(https://www.completecsharptutorial.com/wp-content/uploads/2017/11/New-Picture-8.png)
Now, your project is ready to use **JavaScript** and **AJAX**.

Create Forms and Controller.

> **Step 1:** Create Form in `Index.cshtml`

```
@model MvcForms.Models.StudentModel
<script src="@Url.Content("~/Scripts/jquery-1.10.2.min.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.unobtrusive-ajax.js")" type="text/javascript"></script>

<h4 id="id1" style="color:purple"></h4>
<hr />
<h3><b>Forms - Strongly Typed AJAX (Asynchronous)</b></h3>
    @using (Ajax.BeginForm("Form3", "Home", new AjaxOptions
    {
        HttpMethod = "POST",
        UpdateTargetId = "id1",
        LoadingElementId = "LoadingImage",
        OnSuccess = "onSuccess_Message",
        OnFailure="onFailure_Message"

    }))
    {
        <table>
            <tr>
                <td>Enter ID: </td>
                <td>@Html.TextBoxFor(m => m.Id)</td>
            </tr>
            <tr>
                <td>Enter Name: </td>
                <td>@Html.TextBoxFor(m => m.Name)</td>
            </tr>
            <tr>
                <td>Addon: </td>
                <td>@Html.CheckBoxFor(m => m.Addon)</td>
            </tr>
            <tr>
                <td colspan="2"><input type="submit" value="Submit Form" /></td>
            </tr>
        </table>
        <div id="LoadingImage" style="display:none">Loading...</div>
        <div id="onSuccess_Message"></div>
        <div id="onFailure_Message"></div>
    }
```

Let’s understand this code:

1. Add these two scripts in the project.

```
<script src="@Url.Content("~/Scripts/jquery-1.10.2.min.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.unobtrusive-ajax.js")" type="text/javascript"></script>
```

You must check the correct version of javascript installed on your project. **Jquery-1.xx.x**.

2.

```
@using (Ajax.BeginForm("Form3", "Home", new AjaxOptions
    {
        HttpMethod = "POST",
        UpdateTargetId = "id1",
        LoadingElementId = "LoadingImage",
        OnSuccess = "onSuccess_Message",
        OnFailure="onFailure_Message"

    }))
```

a. **`Ajax.BeginForm`** is used for creating Asynchronous AJAX Forms.

b. **`Form3`** is an Action method.

c. **`Home`** is a Controller name.

d. **`HttpMethod = “POST”`** denotes that data will be sent to server using POST method.

e. **`UpdateTargetId`** updates the area which will get updated and display output. In my program, **`<h4 id="id1" style="color:purple"></h4>`** will be updated and display output.

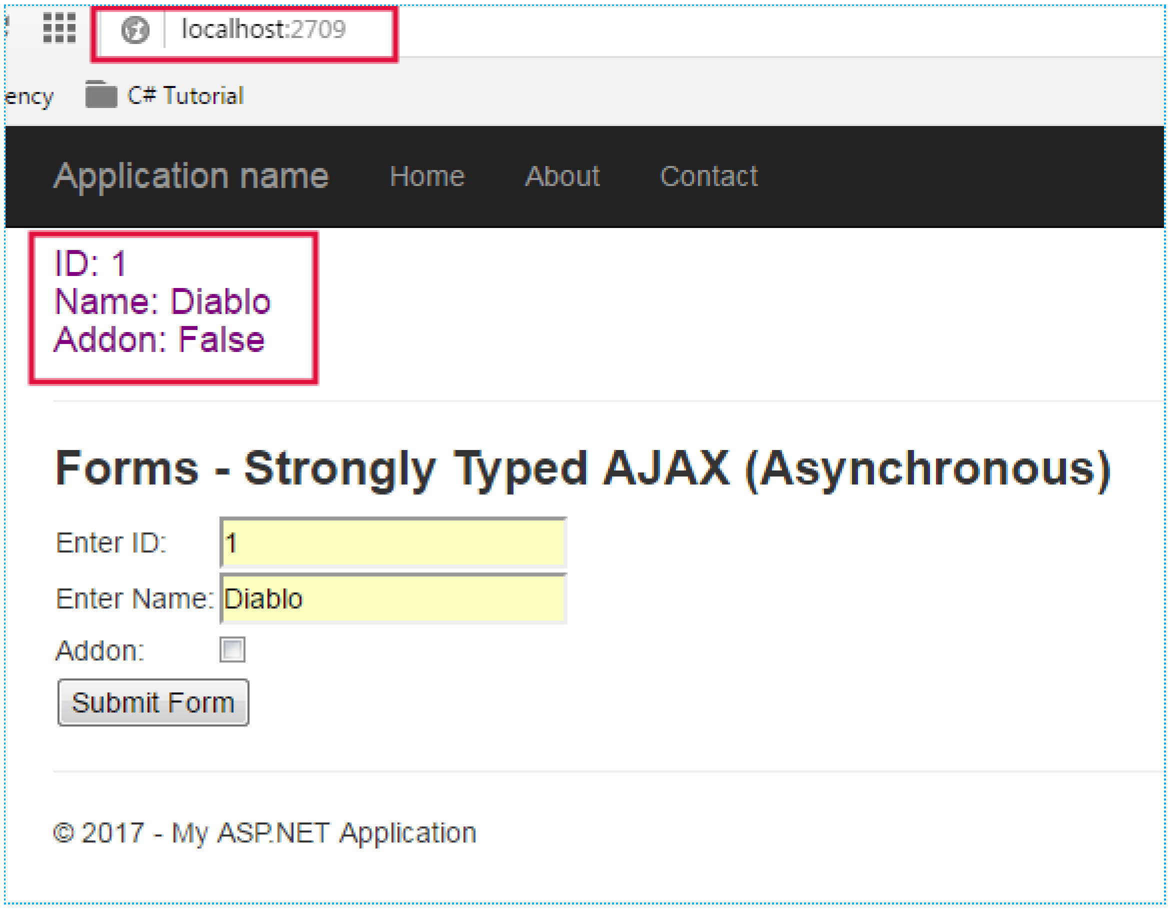
f. **`LoadingElementId`** display the loading image or loading message meanwhile AJAX is posting and retrieving data from models or controllers.

g. **`OnSuccess`** works when task completed successfully.

h. **`OnFailure`** works when task gets failed.

Step 2: Go to **`HomeController`** and add the following action method.

```
[HttpPost]
        public ActionResult Form3(Models.StudentModel sm)
        {
            if(ModelState.IsValid)
            {
                System.Text.StringBuilder sb = new System.Text.StringBuilder();
                sb.Append("ID: " + sm.Id + "<br />");
                sb.Append("Name: " + sm.Name + "<br />");
                sb.Append("Addon: " + sm.Addon + "<br />");
                return Content(sb.ToString());
            }
            else
            {
                return View("Index");
            }
        }
```



ID: 1
Name: Diablo
Addon: False

Forms - Strongly Typed AJAX (Asynchronous)

Enter ID: 1
Enter Name: Diablo
Addon:

(https://www.completecsharptutorial.com/wp-content/uploads/2017/11/New-Picture-9.png)

4. PURE HTML FORMS WITH AJAX AND JQUERY

In this method, you can not only send data from input controls but can also use html elements like <p>…</p>, <span>…</span> to send data to controllers. This is pure JQuery and AJAX query.

Create Forms

> **Step 1.** Go to `Index.cshtml` and create form like this.

```
<h3><b>Forms - Pure HTML and JQUERY</b></h3>

    <table>
        <tr>
            <td>Enter ID: </td>
            <td><input type="text" id="Id" /></td>
        </tr>
        <tr>
            <td>Enter Name: </td>
            <td><input type="text" id="Name" /></td>
        </tr>
        <tr>
            <td>Addon: </td>
            <td><input type="checkbox" id="Addon" /></td>
        </tr>
        <tr>
            <td colspan="2"><button onclick="submit()">Submit Form</button></td>
        </tr>
    </table>

<h4 style="color:purple" id="output"></h4>

<script src="~/Scripts/jquery-1.10.2.min.js" type="text/javascript"></script>
<script>
    function submit(){
        var data = {
            Id: $('#Id').val(),
            Name: $('#Name').val(),
            Addon: $('#Addon').is(':checked')
        };

        $.post("/Home/Form4", { sm: data }, function () { alert('Successfully Saved') });
    }
</script>
```

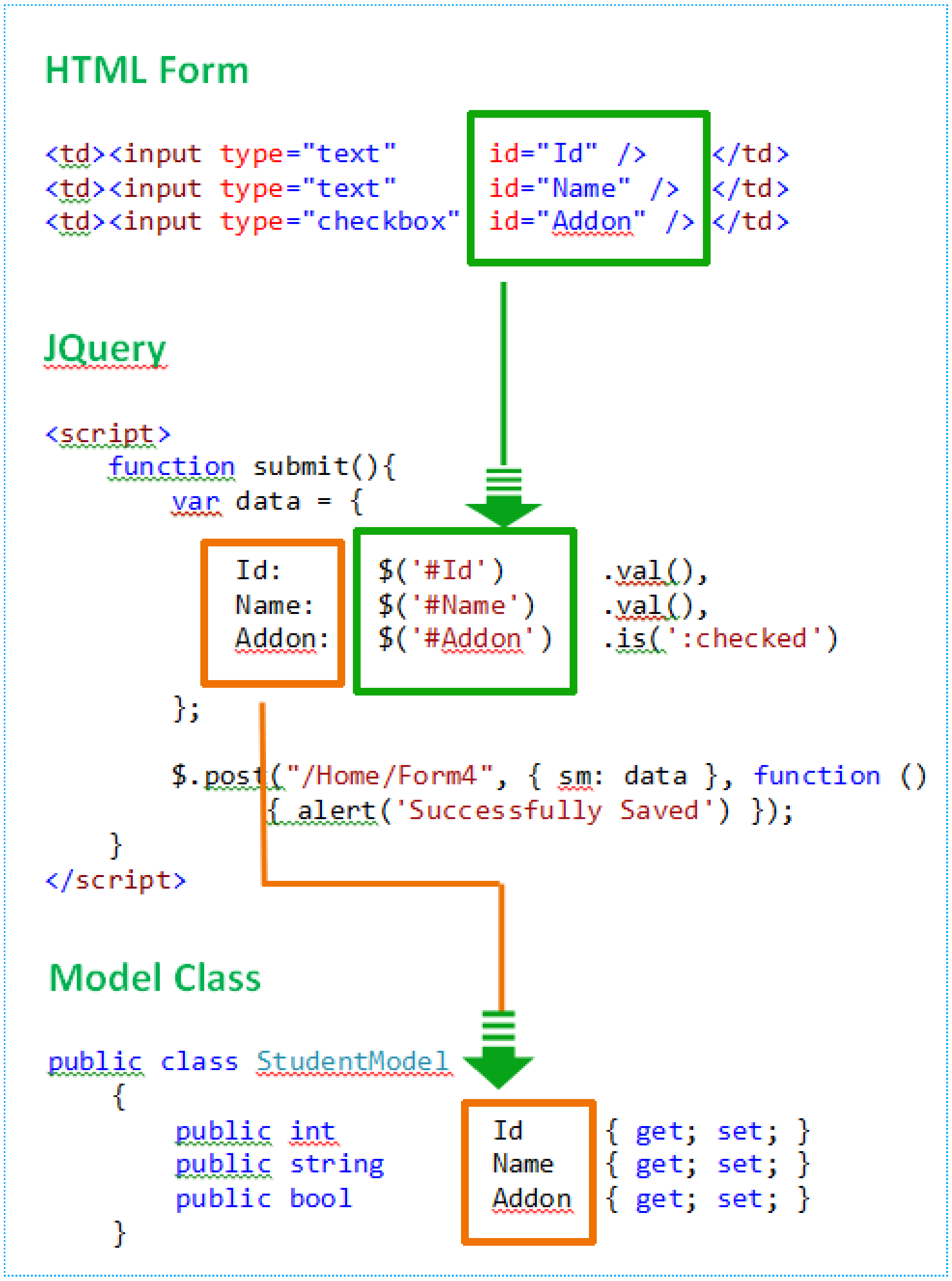
Step 2: Go to **`HomeController`** and add following action method.

```
[HttpPost]
        public ActionResult Form4(StudentModel sm)
        {
            string value = "ID: "+ Convert.ToString(sm.Id)
                + "<br />Name: " + sm.Name
                + "<br />Addon: " + Convert.ToString(sm.Addon);

            string s = "$('#output').html('" + value + "');";
            return JavaScript(s);
        }
```

Let's understand this code:

1. Created a form using pure html control and call a **`submit()`** function in button **`onclick`** event.

2. Must map the variable and member according to your models and controllers member. See the picture below.



```
<td><input type="text"     id="Id" />      </td>
<td><input type="text"     id="Name" />    </td>
<td><input type="checkbox" id="Addon" />   </td>
```

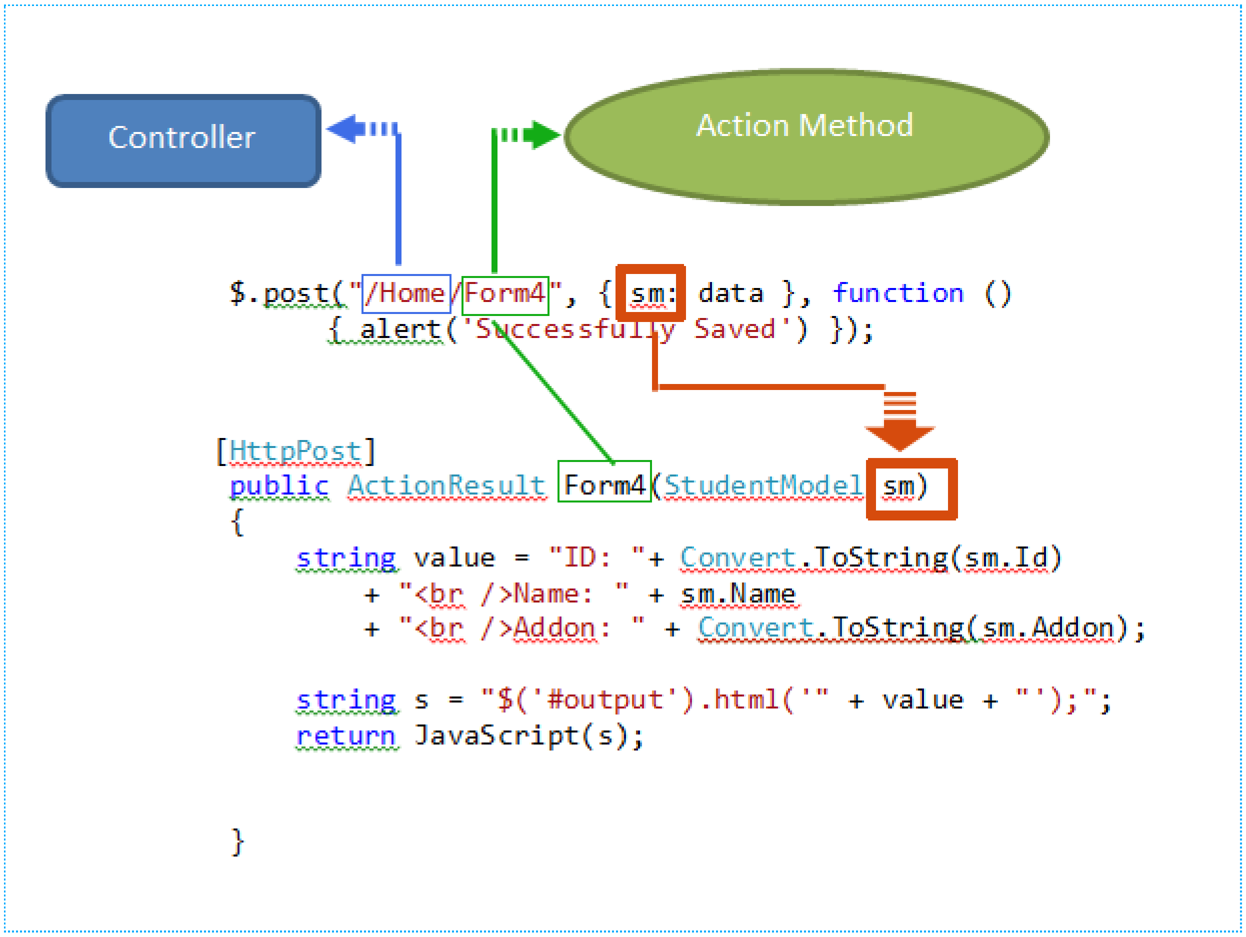
```
<script>
    function submit(){
        var data = {
            Id:     $('#Id')     .val(),
            Name:   $('#Name')   .val(),
            Addon:  $('#Addon')  .is(':checked')
        };

        $.post("/Home/Form4", { sm: data }, function ()
        { alert('Successfully Saved') });
    }
</script>
```

```
public class StudentModel
    {
        public int       Id     { get; set; }
        public string    Name   { get; set; }
        public bool      Addon  { get; set; }
    }
```

(https://www.completecsharptutorial.com/wp-content/uploads/2017/11/New-Picture-11.png) **3.** **`$.post()`** method post form data to controller.

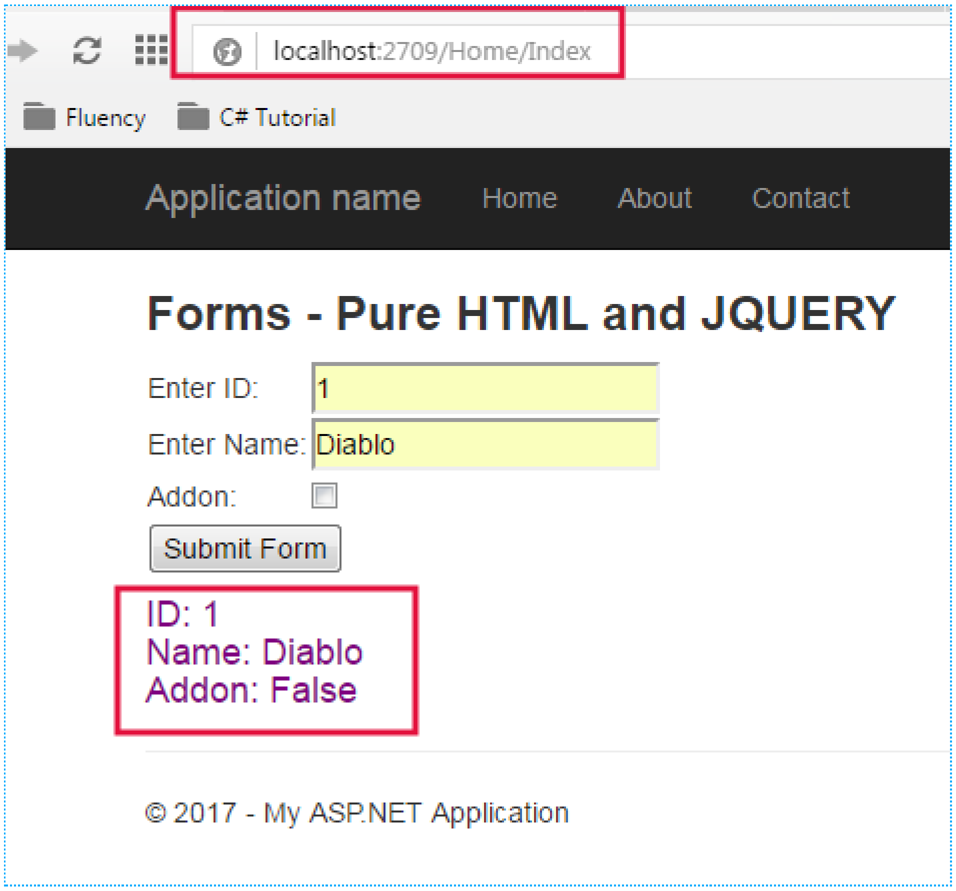
a. **`/Home/Form4`** - **Home** is controller name and **Form4** is an action method.

b. **`sm: data`** – **`sm`** is an object of **`StudenModel`** in **Form4** Action Method.



```
$.post("/Home/Form4", { sm: data }, function ()
    { alert('Successfully Saved') });


[HttpPost]
public ActionResult Form4(StudentModel sm)
{
    string value = "ID: "+ Convert.ToString(sm.Id)
        + "<br />Name: " + sm.Name
        + "<br />Addon: " + Convert.ToString(sm.Addon);

    string s = "$('#output').html('" + value + "');";
    return JavaScript(s);


}
```

(https://www.completecsharptutorial.com/wp-content/uploads/2017/11/New-Picture-12-copy.png)



Forms - Pure HTML and JQUERY

Enter ID:	1
Enter Name:	Diablo
Addon:	☐

```
ID: 1
Name: Diablo
Addon: False
```

© 2017 - My ASP.NET Application

(https://www.completecsharptutorial.com/wp-content/uploads/2017/11/New-Picture-10.png)

SUMMARY

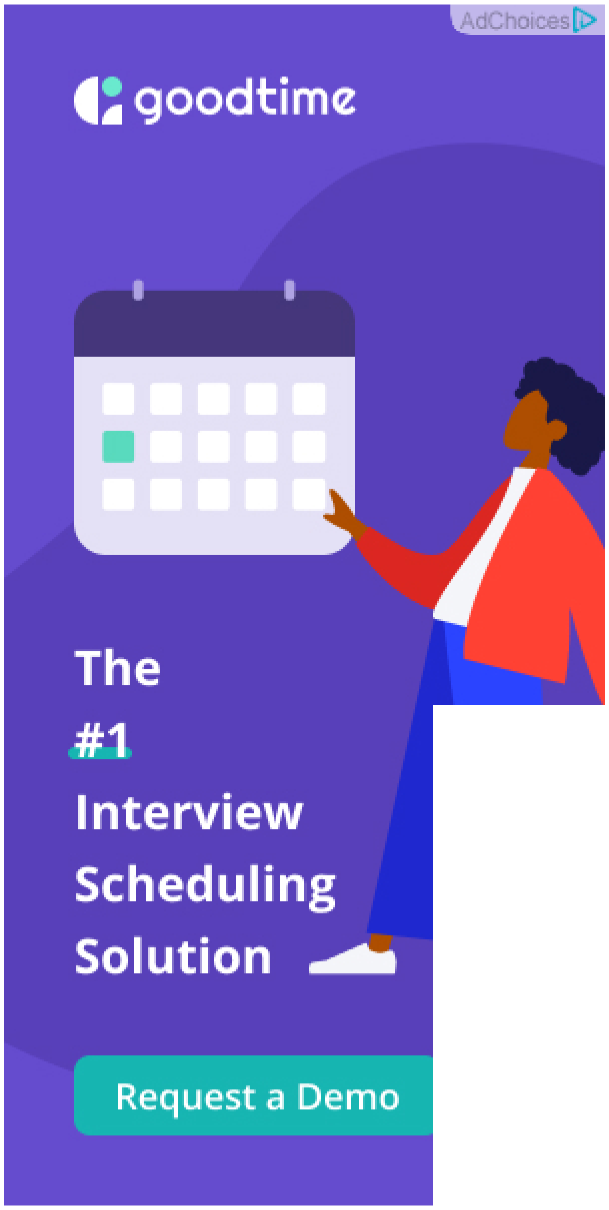
In this tutorial, you learned 4 different ways to create form and submit data to the controller. All these 4 ways used widely in MVC and I hope now you will be able to create a form in ASP.NET MVC. In the next chapter, you will learn FormCollection object in details with programming example. FormCollection objects make a job much easier when collecting form data into the controller.

MORE ARTICLES

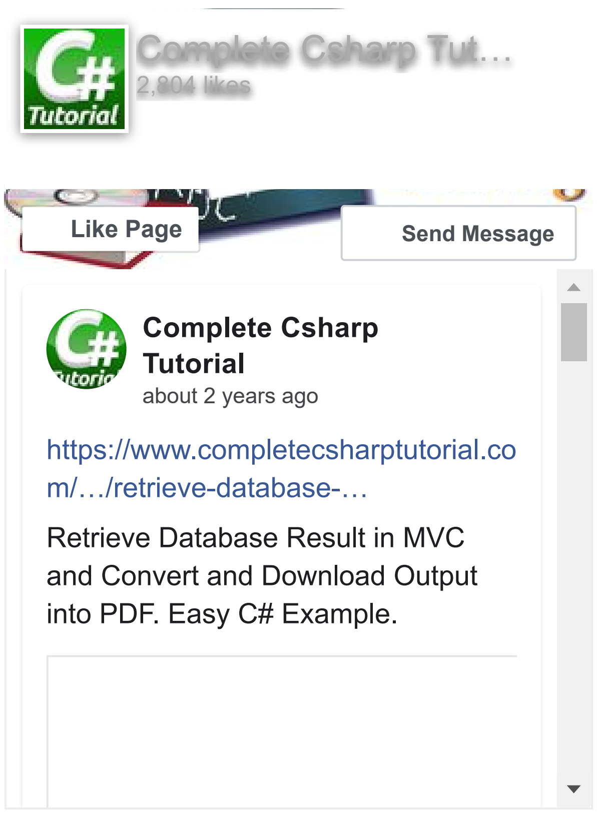
Previous Article (https://www.completecsharptutorial.com/asp-net-mvc5/render-action-and-navigation-property-in-asp-net-mvc.php)

Next Article (https://www.completecsharptutorial.com/asp-net-mvc5/formcollection-object-in-asp-net-mvc-with-example.php)

SHARE YOUR THOUGHT


AdChoices
goodtime
The #1 Interview Scheduling Solution
Request a Demo

Like us



Complete Csharp Tutorial
about 2 years ago

https://www.completecsharptutorial.com/…/retrieve-database-…

Retrieve Database Result in MVC and Convert and Download Output into PDF. Easy C# Example.