Consume Web API Get method in ASP.NET MVC

We created Web API and implemented various Get methods to handle different HTTP GET requests in the Implement Get Method section. Here we will consume one of those Get methods named GetAllStudents() shown below.

Example: Sample Web API

public class StudentController : ApiController

{

public StudentController()

{

}

public IHttpActionResult GetAllStudents(bool includeAddress = false)

{

IList<StudentViewModel> students = null;

using (var ctx = new SchoolDBEntities())

{

students = ctx.Students.Include("StudentAddress").Select(s => new StudentViewModel()

{

Id = s.StudentID,

FirstName = s.FirstName,

LastName = s.LastName,

Address = s.StudentAddress == null || includeAddress == false ? null : new AddressViewModel()

{

StudentId = s.StudentAddress.StudentID,

Address1 = s.StudentAddress.Address1,

Address2 = s.StudentAddress.Address2,

City = s.StudentAddress.City,

State = s.StudentAddress.State

}

}).ToList<StudentViewModel>();

}

if (students.Count == 0)

{

return NotFound();

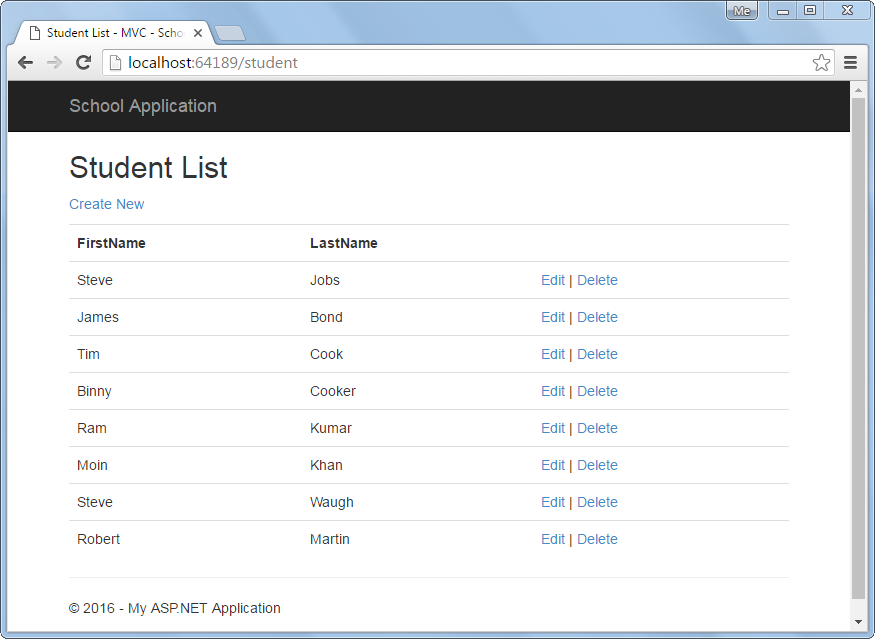
}

return Ok(students);

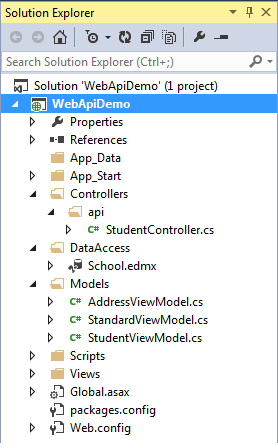
}

}

The above GetAllStudents() action method will handle HTTP GET request http://localhost:64189/api/student and will return a list of students. We will send this HTTP request in the ASP.NET MVC controller to get all the student records and display them in the MVC View. The view will look like below.

[](https://www.tutorialsteacher.com/Content/images/webapi/sample-ui.png)Student List View

The following is a Web API + MVC project structure. We will add necessary classes in this project.

[](https://www.tutorialsteacher.com/Content/images/webapi/webapi-project-structure.png)

Web API Project

We have already created the following StudentViewModel class under Models folder.

Example: Model Class

public class StudentViewModel

{

public int Id { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public AddressViewModel Address { get; set; }

public StandardViewModel Standard { get; set; }

}

Let's consume above Web API into ASP.NET MVC application step by step.

**Step 1:**

First of all, create MVC controller class called StudentController in the Controllers folder as shown below. Right click on the Controllers folder > **Add..** > select **Controller.**

Example: MVC Controller

public class StudentController : Controller

{

// GET: Student

public ActionResult Index()

{

return View();

}

}

**Step 2:**

We need to access Web API in the Index() action method using HttpClient as shown below.

Example: MVC Controller

public class StudentController : Controller

{

// GET: Student

public ActionResult Index()

{

IEnumerable<StudentViewModel> students = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:64189/api/");

//HTTP GET

var responseTask = client.GetAsync("student");

responseTask.Wait();

var result = responseTask.Result;

if (result.IsSuccessStatusCode)

{

var readTask = result.Content.ReadAsAsync<IList<StudentViewModel>>();

readTask.Wait();

students = readTask.Result;

}

else //web api sent error response

{

//log response status here.

students = Enumerable.Empty<StudentViewModel>();

ModelState.AddModelError(string.Empty, "Server error. Please contact administrator.");

}

}

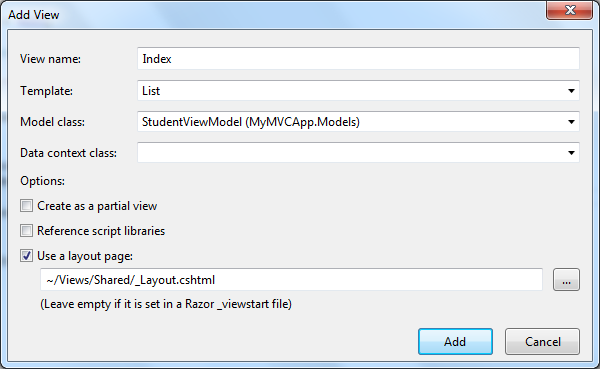
return View(students);

}

}

**Step 3:**

Now, we need to add Index view. Right click in the Index action method and select **Add View.** option. This will open Add View popup as shown below. Now, select List as template and StudentViewModel as Model class as below (we already created StudentViewModel in the previous section).

[](https://www.tutorialsteacher.com/Content/images/webapi/add-view-mvc.png)Add View in ASP.NET MVC

Click **Add** to add Index view in the **Views** folder. This will generate following Index.cshtml.

Index.cshtml

@model IEnumerable<WebAPIDemo.Models.StudentViewModel>

@{

ViewBag.Title = "Index";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Index</h2>

<p>

@Html.ActionLink("Create New", "Create")

</p>

<table class="table">

<tr>

<th>

@Html.DisplayNameFor(model => model.FirstName)

</th>

<th>

@Html.DisplayNameFor(model => model.LastName)

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.FirstName)

</td>

<td>

@Html.DisplayFor(modelItem => item.LastName)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id=item.Id }) |

@Html.ActionLink("Details", "Details", new { id=item.Id }) |

@Html.ActionLink("Delete", "Delete", new { id=item.Id })

</td>

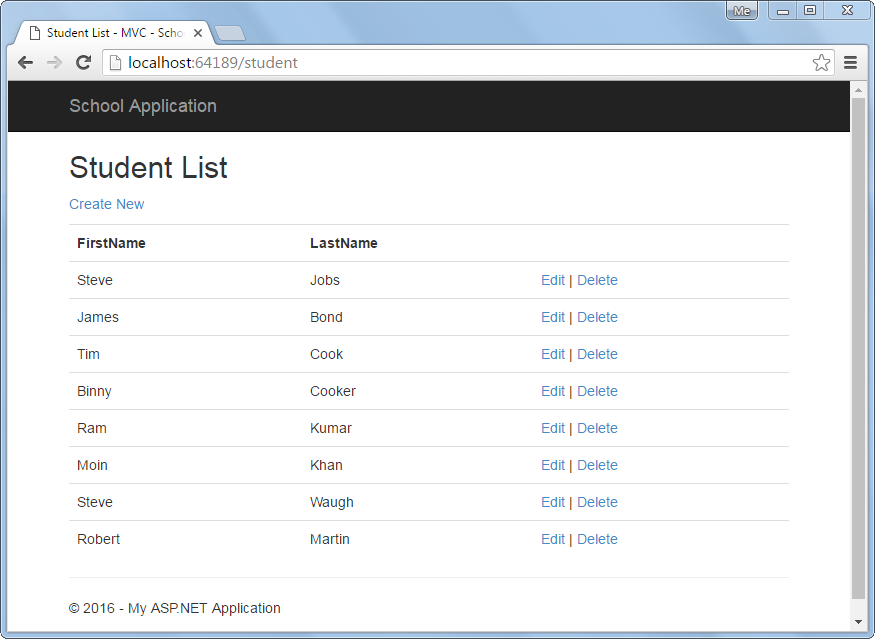
</tr>

}

</table>

Remove Details link from the View because we will not create Details page here.

Now, run the application and you will see list of students in the browser as shown below.

[](https://www.tutorialsteacher.com/Content/images/webapi/sample-ui.png)Student List View

Display Error

We have successfully displayed records in the view above but what if Web API returns error response?

To display appropriate error message in the MVC view, add the ValidationSummary() as shown below.

Index.cshtml

@model IEnumerable<WebAPIDemo.Models.StudentViewModel>

@{

ViewBag.Title = "Index";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Index</h2>

<p>

@Html.ActionLink("Create New", "Create")

</p>

<table class="table">

<tr>

<th>

@Html.DisplayNameFor(model => model.FirstName)

</th>

<th>

@Html.DisplayNameFor(model => model.LastName)

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.FirstName)

</td>

<td>

@Html.DisplayFor(modelItem => item.LastName)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id=item.Id }) |

@Html.ActionLink("Delete", "Delete", new { id=item.Id })

</td>

</tr>

}

<tr>

<td>

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

</td>

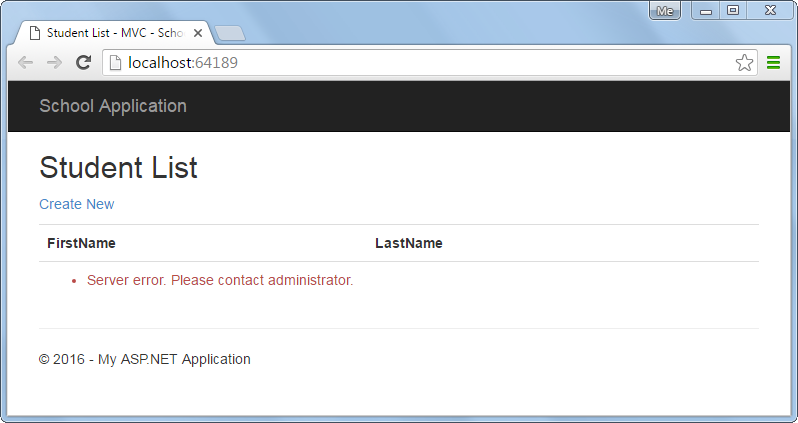
</tr>

</table>

In the above view, we have added @Html.ValidationSummary(true, "", new { @class = "text-danger" }) in the last row of the table. This is to display error message if Web API returns error response with the status other than 200 OK.

Please notice that we have added model error in the Index() action method in StudentController class created in the step 2 if Web API responds with the status code other than 200 OK.

So now, if Web API returns any kind of error then Student List view will display the message below.

[](https://www.tutorialsteacher.com/Content/images/webapi/display-error.png)Display Error Message

Consume Web API Post method in ASP.NET MVC

In the previous section, we learned how to consume Web API Get method and display records in the ASP.NET View. Here, we will see how to consume Post method of Web API to create a new record in the data source.

We already created Web API with Post method in the [Implement Post Method](https://www.tutorialsteacher.com/webapi/implement-post-method-in-web-api) section shown below.

Example: Sample Web API with Post Method

public class StudentController : ApiController

{

public StudentController()

{

}

//Get action methods of the previous section

public IHttpActionResult PostNewStudent(StudentViewModel student)

{

if (!ModelState.IsValid)

return BadRequest("Not a valid model");

using (var ctx = new SchoolDBEntities())

{

ctx.Students.Add(new Student()

{

StudentID = student.Id,

FirstName = student.FirstName,

LastName = student.LastName

});

ctx.SaveChanges();

}

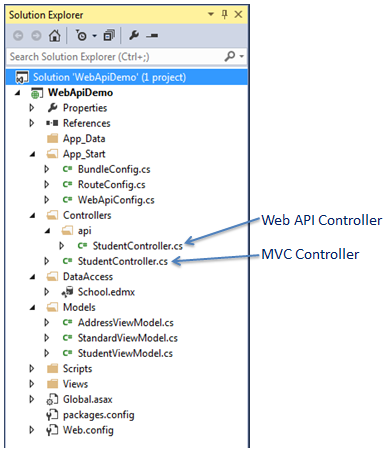
return Ok();

}

}

In the above Web API, PostNewStudent method will handle HTTP POST request http://localhost:64189/api/student. It will insert new record in the database using Entity Framework and will return 200 OK response status.

The following is a Web API + MVC project structure created in the previous sections. We will add necessary classes in this project.

[](https://www.tutorialsteacher.com/Content/images/webapi/project-structure.png)Web API Project

We have already created the following StudentViewModel class under Models folder.

Example: Model Class

public class StudentViewModel

{

public int Id { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public AddressViewModel Address { get; set; }

public StandardViewModel Standard { get; set; }

}

Now, let's create MVC view to create a new record by consuming the above Web API Post method.

**Step 1:**

First, we need to add action method "create" which will render "Create New Student" view where user can enter data and submit it. We have already created StudentController class in the previous section to display student list view. Here, add "create" action method to render "Create New Student" view shown below.

Example: MVC Controller

public class StudentController : Controller

{

public ActionResult Index()

{

//consume Web API Get method here..

return View();

}

public ActionResult create()

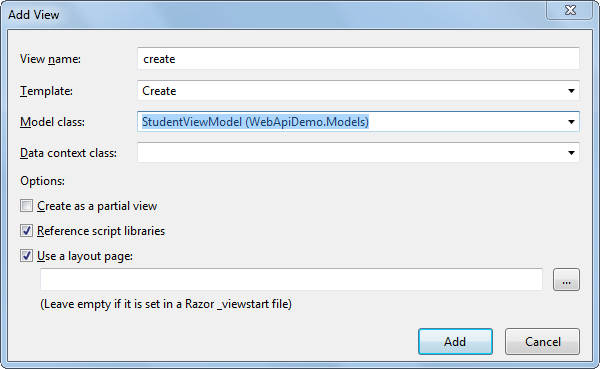
{

return View();

}

}

Now, right click in the above action method and select **Add View..** This will open following Add View popup.

[](https://www.tutorialsteacher.com/Content/images/webapi/add-create-view.png)Add View in ASP.NET MVC

Now, select Create Template, StudentViewModel class as a model and click on Add button as shown above. This will generate createcshtml in the Views > Student folder as below.

create.cshtml

@model WebApiDemo.Models.StudentViewModel

@{

ViewBag.Title = "Create New Student - MVC";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Create New Student</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.FirstName, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.FirstName, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.FirstName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LastName, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.LastName, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.LastName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Create" class="btn btn-default" />

</div>

</div>

</div>

}

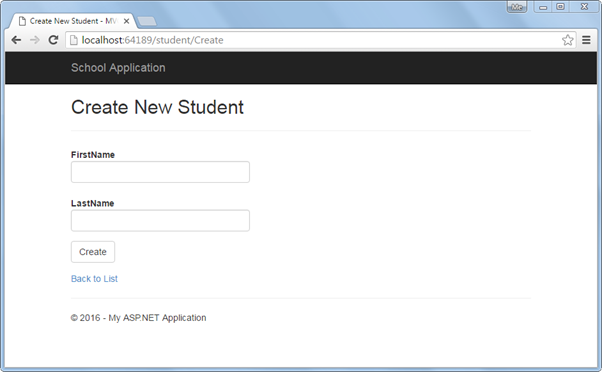
<div>

@Html.ActionLink("Back to List", "Index")

</div>

In the above view, Html.BeginForm() generates HTML form tag <form> action="/Student/Create" method="post" </form> which will send post request when user clicks on the create button.

Now, run the project and navigate to http://localhost:64189/student/create. It will display the simple data entry view as shown below.

[](https://www.tutorialsteacher.com/Content/images/webapi/create-new-student-ui.png)Create New Student View

As soon as the user enters student data and clicks on the **Create** button in the above view, it will send Post request to the Student MVC controller. To handle this post request add HttpPost action method "create" as shown below.

Example: Post Method in MVC Controller

public class StudentController : Controller

{

public ActionResult Index()

{

//consume Web API Get method here..

return View();

}

public ActionResult create()

{

return View();

}

[HttpPost]

public ActionResult create(StudentViewModel student)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:64189/api/student");

//HTTP POST

var postTask = client.PostAsJsonAsync<StudentViewModel>("student", student);

postTask.Wait();

var result = postTask.Result;

if (result.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

}

ModelState.AddModelError(string.Empty, "Server Error. Please contact administrator.");

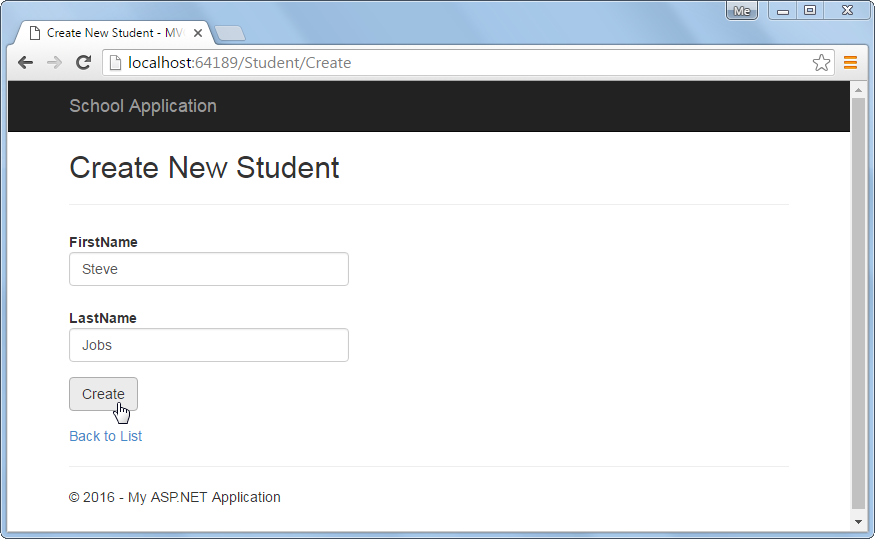
return View(student);

}

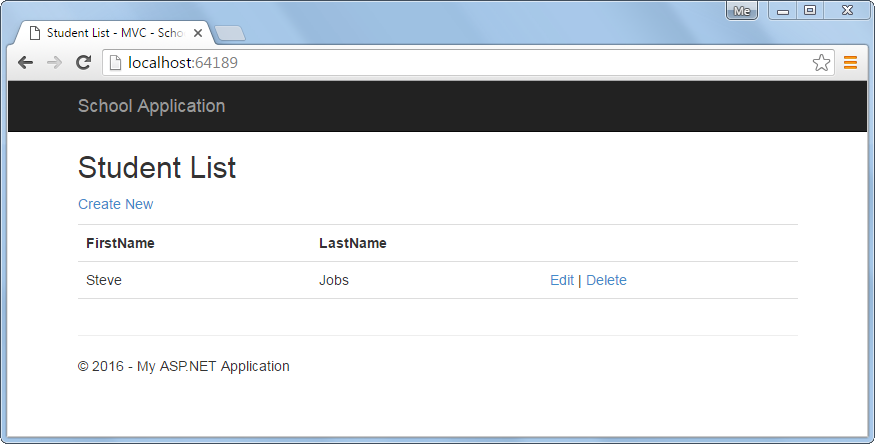
}

As you can see in the above HttpPost action method create(), it uses HttpClient to send HTTP POST request to Web API with StudentViewModel object. If response returns success status then it will redirect to the list view. Visit [HttpClient](http://localhost:56670/webapi/consuming-web-api-in-dotnet-using-httpclient) section to learn more about it.

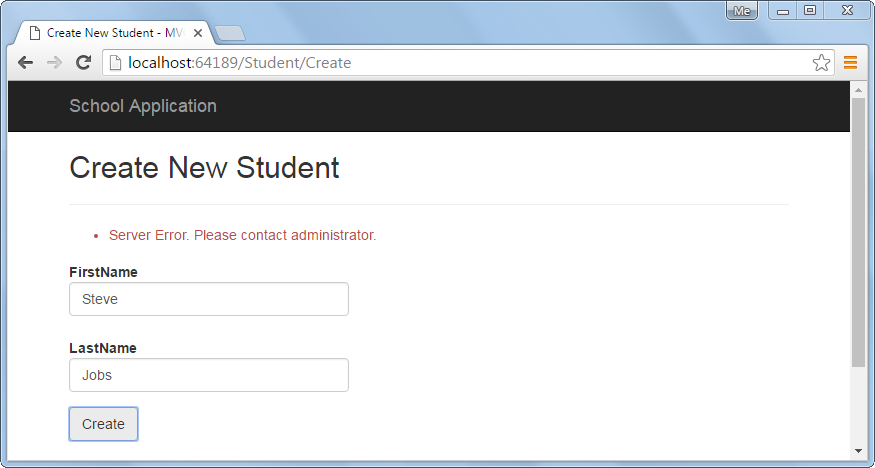
Now, run the project and navigate to http://localhost:64189/student/create, enter student information as shown below.

[](https://www.tutorialsteacher.com/Content/images/webapi/post-record.png)Create a New Student

Now, on the click of create button above, it will insert a new record in the DB and redirect to the list view as shown below.

[](https://www.tutorialsteacher.com/Content/images/webapi/list-view.png)Redirect to Student List View

Also, the above create view will display an error message if Web API sends error response as shown below.

[](https://www.tutorialsteacher.com/Content/images/webapi/error-message-in-post-view.png)Display Error Message

So in this way we can consume Post method of Web API to execute HTTP POST request to create a new record.

# Consume Web API Put method in ASP.NET MVC

In the previous two sections, we learned how to consume Web API Get and Post methods in the ASP.NET View. Here, we will see how to consume Put method of Web API to update an existing record.

We already created Web API with Put method that handles HTTP PUT request in the [Implement Put Method](https://www.tutorialsteacher.com/webapi/implement-put-method-in-web-api) section as below.

Example: Sample Web API with Put method

public class StudentController : ApiController

{

public StudentController()

{

}

public IHttpActionResult Put(StudentViewModel student)

{

if (!ModelState.IsValid)

return BadRequest("Not a valid data");

using (var ctx = new SchoolDBEntities())

{

var existingStudent = ctx.Students.Where(s => s.StudentID == student.Id).FirstOrDefault<Student>();

if (existingStudent != null)

{

existingStudent.FirstName = student.FirstName;

existingStudent.LastName = student.LastName;

ctx.SaveChanges();

}

else

{

return NotFound();

}

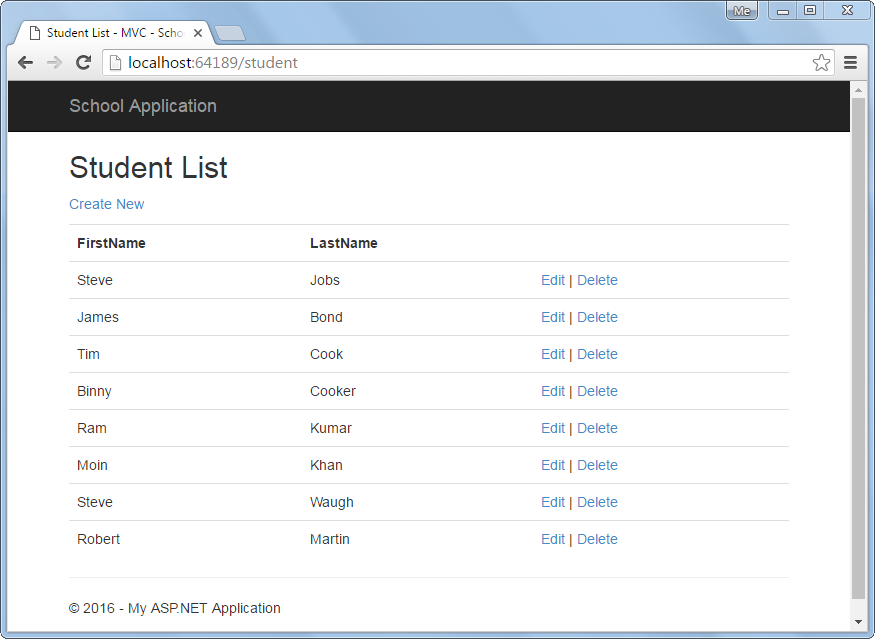
}

return Ok();

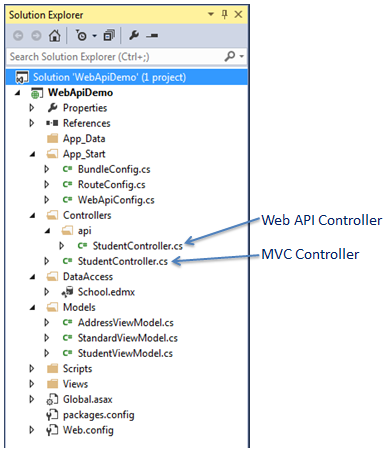
}

}

We created Student List view in the previous section as below. In the below view there is an edit link for each record to edit that particular record. We will handle edit functionality in this section.

[](https://www.tutorialsteacher.com/Content/images/webapi/sample-ui.png)Student List View

The following is a Web API + MVC project structure created in the previous sections. We will add necessary classes in this project.

[](https://www.tutorialsteacher.com/Content/images/webapi/project-structure.png)Web API Project

We have already created the following StudentViewModel class under Models folder.

Example: Model Class

public class StudentViewModel

{

public int Id { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public AddressViewModel Address { get; set; }

public StandardViewModel Standard { get; set; }

}

So let's consume Web API Put method by implementing edit functionality.

**Step 1:**

In the above Student List view, when user clicks on the Edit link it will send HTTP GET request http://localhost:64189/student/edit/{id} to the MVC controller. So, we need to add HttpGet action method "Edit" in the StudentController to render an edit view as shown below.

Example: Implement Edit Action Method

public class StudentController : Controller

{

public ActionResult Index()

{

//consume Web API Get method here..

return View();

}

public ActionResult Edit(int id)

{

StudentViewModel student = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:64189/api/");

//HTTP GET

var responseTask = client.GetAsync("student?id=" + id.ToString());

responseTask.Wait();

var result = responseTask.Result;

if (result.IsSuccessStatusCode)

{

var readTask = result.Content.ReadAsAsync<StudentViewModel>();

readTask.Wait();

student = readTask.Result;

}

}

return View(student);

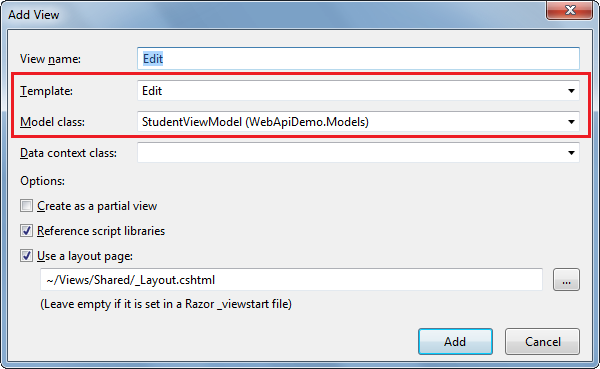
}

}

As you can see above, Edit() action method includes id parameter. This id parameter will be bound to the query string id parameter. We use this id to get a student record from the database using HttpClient and pass the student record in the edit view. Visit [HttpClient](https://www.tutorialsteacher.com/webapi/consuming-web-api-in-dotnet-using-httpclient) section to know more about it.

**Step 2:**

Create edit view by right clicking in the above Edit action method and select **Add View..** This will open Add View popup as shown below.

[](https://www.tutorialsteacher.com/Content/images/webapi/edit-view-mvc.png)Add View in ASP.NET MVC

In the Add View popup, select Edit template and StudentViewModel as a model class as shown above. Click Add button to generate Edit.cshtml view in the Views > Student folder as shown below.

Edit.cshtml

@model WebApiDemo.Models.StudentViewModel

@{

ViewBag.Title = "Edit Student - MVC";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Edit Student</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

@Html.HiddenFor(model => model.Id)

<div class="form-group">

@Html.LabelFor(model => model.FirstName, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.FirstName, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.FirstName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LastName, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.LastName, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.LastName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Save" class="btn btn-default" />

</div>

</div>

</div>

}

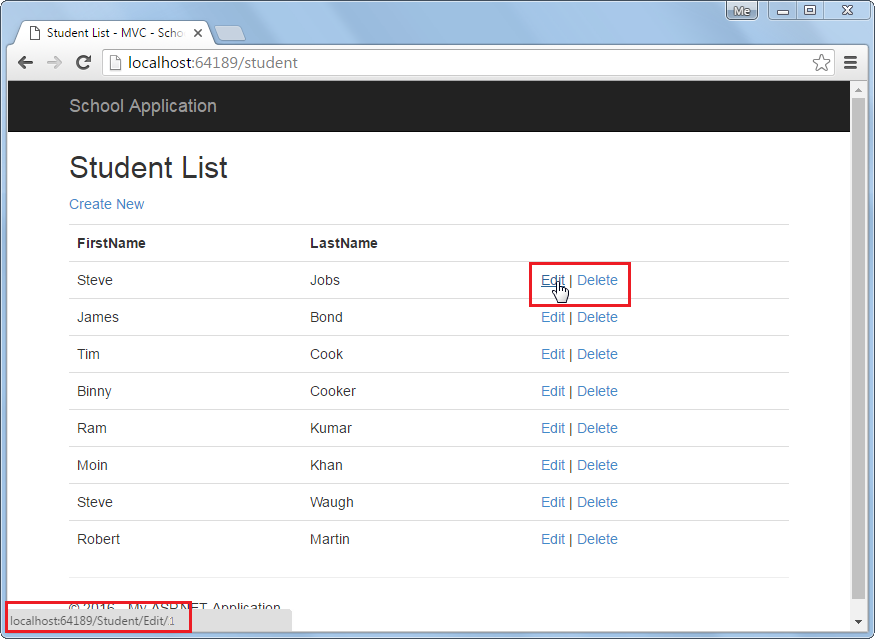
<div>

@Html.ActionLink("Back to List", "Index")

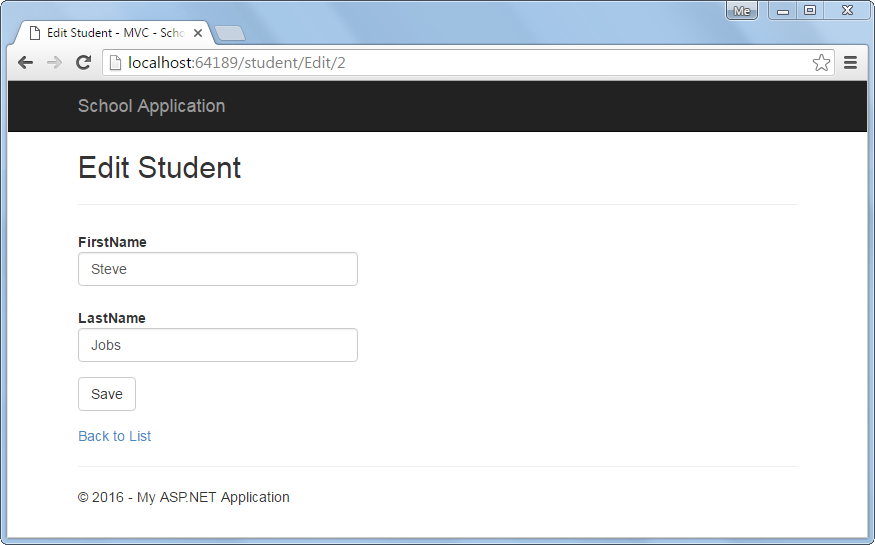
</div>

In the above view, Html.BeginForm() generates HTML form tag <form> action="/Student/edit" method="post" </form> which will send post request when user clicks on the save button.

Now, it will display following Student List view when you run the project by pressing Ctrl + F5.

[](https://www.tutorialsteacher.com/Content/images/webapi/edit-click.png)Student List View

It will display following edit view when you click on the Edit link in the above view.

[](https://www.tutorialsteacher.com/Content/images/webapi/edit-view-mvc2.png)Edit View

Now, implement HttpPost Edit action method which will be executed when user clicks on the **Save** button above.

**Step 3:**

Add HttpPost action method in StudentController of MVC which will send HTTP PUT request to Web API to update current record.

Example: Implement HttpPost Action Method

public class StudentController : Controller

{

public ActionResult Edit(int id)

{

StudentViewModel student = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:64189/api/");

//HTTP GET

var responseTask = client.GetAsync("student?id=" + id.ToString());

responseTask.Wait();

var result = responseTask.Result;

if (result.IsSuccessStatusCode)

{

var readTask = result.Content.ReadAsAsync<StudentViewModel>();

readTask.Wait();

student = readTask.Result;

}

}

return View(student);

}

[HttpPost]

public ActionResult Edit(StudentViewModel student)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:64189/api/student");

//HTTP POST

var putTask = client.PutAsJsonAsync<StudentViewModel>("student", student);

putTask.Wait();

var result = putTask.Result;

if (result.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

}

return View(student);

}

}

As you can see above, HttpPost Edit action method uses HttpClient to send HTTP PUT request to the Web API with updated student record. Visit [HttpClient](https://www.tutorialsteacher.com/webapi/consuming-web-api-in-dotnet-using-httpclient) section to learn more about it.

So in this way we can consume Put method of Web API to execute HTTP PUT request to edit an existing record.

# Consume Web API Delete Method in ASP.NET MVC

In the previous sections, we consumed Get, Post and Put methods of the Web API. Here, we will consume Delete method of Web API in ASP.NET MVC to delete a record.

We have already created Web API with Delete method that handles HTTP DELETE request in the [Implement Delete Method](https://www.tutorialsteacher.com/webapi/implement-delete-method-in-web-api) section as below.

Sample Web API with Delete Method

public class StudentController : ApiController

{

public StudentController()

{

}

public IHttpActionResult Delete(int id)

{

if (id <= 0)

return BadRequest("Not a valid student id");

using (var ctx = new SchoolDBEntities())

{

var student = ctx.Students

.Where(s => s.StudentID == id)

.FirstOrDefault();

ctx.Entry(student).State = System.Data.Entity.EntityState.Deleted;

ctx.SaveChanges();

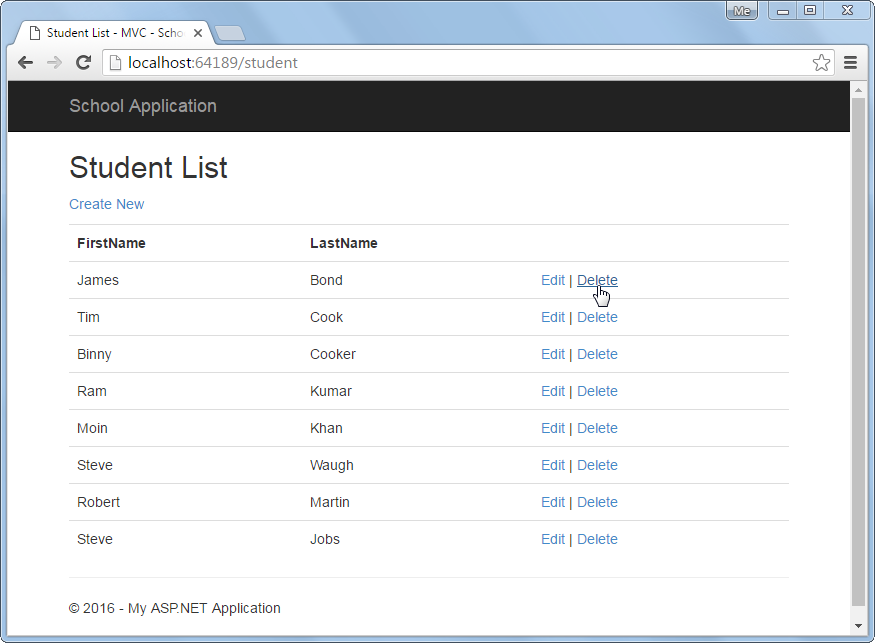
}

return Ok();

}

}

The following is a Student list view created in the [Consuming get method in MVC](https://www.tutorialsteacher.com/webapi/consume-web-api-get-method-in-aspnet-mvc) section. Here, we will implement delete functionality when user clicks on the Delete link in the following UI.

[](https://www.tutorialsteacher.com/Content/images/webapi/delete-view-mvc.png)Student List View

When user clicks on the Delete link in the above UI, it sends HTTP Get request http://localhost:64189/student/delete/{id} to the Student controller with the current id parameter. So let's implement delete functionality by consuming Web API Delete method.

**Step 1:**

Create HttpGet action method Delete with id parameter in the MVC StudentController as shown below.

Example: Implement HttpGet Delete method

public class StudentController : Controller

{

// GET: Student

public ActionResult Index()

{

IList<StudentViewModel> students = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:64189/api/student");

//HTTP GET

var responseTask = client.GetAsync("student");

responseTask.Wait();

var result = responseTask.Result;

if (result.IsSuccessStatusCode)

{

var readTask = result.Content.ReadAsAsync<IList<StudentViewModel>>();

readTask.Wait();

students = readTask.Result;

}

}

return View(students);

}

public ActionResult Delete(int id)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri("http://localhost:64189/api/");

//HTTP DELETE

var deleteTask = client.DeleteAsync("student/" + id.ToString());

deleteTask.Wait();

var result = deleteTask.Result;

if (result.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

}

return RedirectToAction("Index");

}

}

As you can see, Delete() action method above uses HttpClient to send HTTP DELETE request with the current id parameter. The Web API controller shown in the first code example, will handle this DELETE request and delete the record from the data source. Visit [HttpClient](http://localhost:56670/webapi/consuming-web-api-in-dotnet-using-httpclient) section to learn more about it.

So, in this way you can consume Delete method of Web API in ASP.NET MVC.