# Web API

## Web Api Method

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

</head>

<body>

<button id="button">Получить дату</button>

<p id="date"></p>

<script src="~/Scripts/jquery-1.7.1.min.js"></script>

<script type="text/javascript">

$(document).ready(function () {

$("#button").click(getDate);

});

function getDate() {

var url = "/api/date"

// Создание запроса по адресу "/api/date" и вывод ответа в элемент c id="date"

$.getJSON(url, function (data) {

$("#date").text(data);

});

}

</script>

Web Api method

public class DateController : ApiController

{

// GET api/date

public string Get()

{

return DateTime.Now.ToString();

}

}

Web api config

public static class WebApiConfig

{

// ASP.NET маршрутизация необходима для связывания URI и методов контроллеров.

// Регистрация маршрута происходить с помощью его добавления в коллекцию маршрутов

// класса HttpConfiguration с помощью мтода MapHttpRoute.

public static void Register(HttpConfiguration config)

{

config.Routes.MapHttpRoute(

name: "DefaultApi", // 1) Имя маршрута

routeTemplate: "api/{controller}/{id}", // 2) Шаблон маршрута

defaults: new { id = RouteParameter.Optional } // 3) объект, содержащий дополнительные параметры и их настройки

);

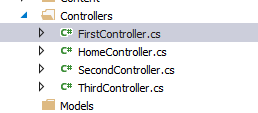
}

}

}

// Шаблон содержит в начале "api/" - это не является необходимым, нужно для совместной работы маршрутизации MVC и

## Call api method



public class FirstController : ApiController

{

// Метод сработает при получении GET запроса.

public string Get()

{

return "Сработал метод Get";

}

}

public class SecondController : ApiController

{

// Метод сработает при получении GET запроса.

[HttpGet]

public string SayHello()

{

return "Сработал метод SayHello c атрибутом HttpGet";

}

}

public class ThirdController : ApiController

{

// Метод сработает при получении GET запроса.

public string GetHelloString()

{

return "Сработал метод GetHelloString";

}

}

Web api config

public static void Register(HttpConfiguration config)

{

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

}

## Get by id||Ajax

public class FruitsController : ApiController

{

static string[] \_fruits =

{

"Orange",

"Apple",

"Plum",

"Apricot"

};

// GET api/name

public IEnumerable<string> Get()

{

return \_fruits;

}

#region Такая реализация возвращает правильный ответ: 404 - Not Found

public HttpResponseMessage Get(int id)

{

if (id < \_fruits.Length)

{

return Request.CreateResponse(HttpStatusCode.OK, \_fruits[id]);

}

else

{

return Request.CreateResponse(HttpStatusCode.NotFound, "Item not found"); //Status code 404

}

}

#endregion

}

Index.chtml

<meta name="viewport" content="width=device-width" />

<title>Index</title>

<style type="text/css">

#namesContainer {

width: 300px;

float: left;

}

#formContainer {

float: left;

margin-top: 15px;

}

#receivedElement {

clear: both;

}

</style>

</head>

<body>

@\*Обработчики элементов управленяи находястся в файле getName.js\*@

<div id="namesContainer">

<ul id="names">

</ul>

</div>

<div id="formContainer">

<div>

<input type="text" id="elementId" />

<button id="button">Получить имя</button>

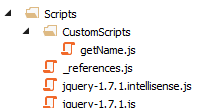
</div>

<div id="receivedElement"></div>

</div>

<script src="~/Scripts/jquery-1.7.1.min.js"></script>

<script src="~/Scripts/CustomScripts/getName.js"></script>



getName.js

(function () {

// Функция вызывается при загрузке скрипта и делает асинхронный запрос на сервер

$.ajax({

url: "/api/fruits",

success: function (names) {

var list = $("#names"); // находим элемент на странцие

for (var i = 0; i < names.length; i++) { // names - JSON объект полученый со стороны сервера.

var name = names[i];

list.append("<li>" + name + "</li>");

}

}

});

// после загрузки документа, находим на страцние кнопку и добавляем метод getName как обработчик на событие click

$(document).ready(function () {

$("#button").on("click", getName);

});

function getName() {

// uri в формате /api/names/5

var link = "/api/fruits/" + $("#elementId").val();

$.ajax({

url: link,

type: "GET",

// в случае успешной обработки запроса

success: function (data) {

$("#receivedElement").text(data); // вывод результата

},

// в случае ошибки

error: function (xhr) {

if (xhr.status == "404") {

alert("Элемен по указанному индексу не найден.");

$("#receivedElement").text(xhr.responseText);

}

if (xhr.status == "500") {

alert("Ошибка сервера");

}

}

});

};

})();

## Web Api exapmle

static List<string> \_fruits = new List<string>()

{

"Orange",

"Apple",

"Plum",

"Apricot"

};

// GET api/fruits

public IEnumerable<string> Get()

{

return \_fruits;

}

// GET api/fruits/5

public string Get(int id)

{

return \_fruits[id];

}

// POST api/fruits

// FromBody - атрибут указывает что значение для параметра name должно извлекаться из тела запроса.

public HttpResponseMessage Post([FromBody]string fruit)

{

\_fruits.Add(fruit);

HttpResponseMessage msg = Request.CreateResponse(HttpStatusCode.Created, fruit);

// var msg = Request.CreateResponse(HttpStatusCode.BadRequest);

// Location заголовок стоит создавать, если новый элемент был создан

msg.Headers.Location = new Uri(Request.RequestUri + "/" + (\_fruits.Count - 1));

return msg;

}

// Для того что бы HTTP методы могли работать при запуске в IIS Express необходимо поменять файл конфигурации, который находится по пути

// C:\\Users\<UserName>\Documents\IISExpress\config\applicationhost.config

// <add name="ExtensionlessUrl-Integrated-4.0" path="\*." verb="GET,HEAD,POST,DEBUG" type="System.Web.Handlers.TransferRequestHandler" preCondition="integratedMode,runtimeVersionv4.0" responseBufferLimit="0" />

// В атрибут verb элемента указанного выше, необъодимо добавить необходимые глаголы.

// Если на компьютере установлен WebDAV Publishing его нужно удалить через "программы и компоненты".

// http://stackoverflow.com/questions/3946283/enabling-html-put-method-on-iis-7-5

// PUT api/fruits/5

public HttpResponseMessage Put(int id, [FromBody]string value)

{

if (id > \_fruits.Count)

{

return Request.CreateResponse(HttpStatusCode.NotFound);

}

else

{

\_fruits[id] = value;

return Request.CreateResponse(HttpStatusCode.OK, \_fruits[id]);

}

}

// DELETE api/fruits/5

public HttpResponseMessage Delete(int id)

{

if (id > \_fruits.Count)

{

return Request.CreateResponse(HttpStatusCode.NotFound);

}

else

{

string deleted = \_fruits[id];

\_fruits.RemoveAt(id);

return Request.CreateResponse(HttpStatusCode.OK, deleted);

}

}

Index.chtml

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

<style type="text/css">

#namesContainer {

width: 300px;

float: left;

}

#formContainer {

float: left;

margin-top: 15px;

}

#formContainer label {

width: 200px;

float: left;

}

#formContainer div {

margin: 0px 0px 20px 0px;

}

#linkContainer {

clear: both;

}

</style>

</head>

<body>

@\*Все элементы полученные со стороны сервера.\*@

<div id="namesContainer">

<ol id="names">

</ol>

</div>

@\*Форма для отправки запросов на сервер\*@

<div id="formContainer">

<div>

<label>Имя нового элемента</label>

<input type="text" id="newName" />

<button id="addNewBtn">Добавить имя</button>

</div>

<div>

<label>Номер элемента</label><input type="text" id="numberOfElement" /><br />

<label>Новое имя</label><input type="text" id="changedName" />

<button id="changeBtn">Изменить</button>

</div>

<div>

<label>Номер элемента для удаления</label>

<input type="text" id="numberToDelete" />

<button id="deleteBtn">Удалить</button>

</div>

@\*Ссылка на последний добавленный элемент\*@

<div id="linkContainer">

<div>Ссылка на последний элемент (значение заголовка 'Location') :</div>

<div id="location"></div>

</div>

</div>

<script src="~/Scripts/jquery-1.7.1.min.js"></script>

<script src="~/Scripts/CustomScripts/namesUpdate.js"></script>

</body>

</html>

namesUpdate.js

(function () {

var baseUri = "/api/fruits";

function loadList() {

$.ajax({

url: baseUri,

success: function (fruits) {

var list = $("#names");

list.empty();

for (var i = 0; i < fruits.length; i++) {

var name = fruits[i];

list.append("<li>" + name + "</li>");

}

}

});

}

loadList();

$(document).ready(function () {

$("#addNewBtn").on("click", postFruit);

$("#changeBtn").on("click", putFruit);

$("#deleteBtn").on("click", deleteFruit);

});

// добавление новго элемента

function postFruit() {

var nameToPost = $("#newName").val();

$.ajax({

url: baseUri,

type: "POST",

data: JSON.stringify(nameToPost), // сериализуем данные в JSON объект перед отправкой на сервер.

dataType: "json",

contentType: "application/json",

//позволяет выполнять различные функции, в зависимости от полученного статус-кода

statusCode: {

201: function () {

alert("Created. Имя успешно добавлено в коллекцию.");

},

400: function () {

alert("Bad Request. Операция не выполнена.");

}

},

success: function (data, textStatus, xhr) {

// data - информация, переданная обратно в теле ответа

// textStatus - статус операции

// xhr - обьект XMLHttpRequest

var locationHeader = xhr.getResponseHeader("Location");

$("#location").html("<a href='" + locationHeader + "'>последний элемент</a>");

$("#names").append("<li>" + data + "</li>");

},

error: errorHandler

});

}

// изменение элемента

function putFruit() {

var id = $("#numberOfElement").val(),

newValue = $("#changedName").val();

$.ajax({

url: baseUri + "/" + id,

type: "PUT",

data: JSON.stringify(newValue),

dataType: "json",

contentType: "application/json",

success: function (data, status, xhr) {

alert('Элемент ' + id + ' изменен');

loadList();

},

error: errorHandler

});

}

// удаление элемента

function deleteFruit() {

var id = $("#numberToDelete").val();

$.ajax({

url: baseUri + "/" + id,

type: "DELETE",

success: function () {

loadList();

},

error: errorHandler

});

}

function errorHandler(xhr, textStatus, error) {

if (xhr.status == "404") {

alert('Элемент не найден.')

}

else if (xhr.status == "400") {

alert('Запрос сформирован не правильно.')

}

else if (xhr.status == "500") {

alert('Ошибка сервера.')

}

}

})();

## Odata

public class ProductsController : ApiController

{

// GET api/values

// Для того, чтоб работал OData протокол необходимо через NuGet добавить в проект библиотеку OData

[HttpGet]

[Queryable]

public IQueryable<Product> AllProducts()

{

return Products.GetAllProducts().AsQueryable<Product>();

}

}

Model

public static class Products

{

static List<Product> products = new List<Product>()

{

new Product() { Id = 1, Name = "Car", Price = 15000 },

new Product() { Id = 2, Name = "Book", Price = 40 },

new Product() { Id = 3, Name = "Laptop", Price = 1400 },

new Product() { Id = 4, Name = "Pen", Price = 1 },

new Product() { Id = 5, Name = "Phone", Price = 4000 },

new Product() { Id = 6, Name = "House", Price = 120000 }

};

public static IEnumerable<Product> GetAllProducts()

{

return products;

}

}

}

getProduct.js

$(document).ready(function () {

$("#button").on("click", getProducts);

});

function getProducts() {

var query = $("#query").val();

$.ajax({

url: "/api/products" + query,

success: function (names) {

var content = '<table>';

$.each(names, function (key, value) {

content += '<tr><th style="width:20px;">'

+ value.Id +

'</th><td style="width:100px;">'

+ value.Name +

'</td><td style="width:50px;">'

+ value.Price +

"</td></tr>";

});

content += "</table>";

$("#products").html(content);

}

});

Index.chtml

@{

Layout = null;

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Index</title>

<style type="text/css">

td {

border: 1px solid black;

}

th {

border: 1px solid black;

text-align: center;

}

#query {

width: 250px;

}

</style>

</head>

<body>

<div style="float: left;">

<div style="height: 200px;">

<div id="formContainer" style="margin-top: 15px;">

<input type="text" id="query" />

<button id="button">Получить продукты</button>

</div>

<div id="products"></div>

</div>

<div style="background-color: #808080; height: 2px;"></div>

<div>

<span>Примеры запросов:</span>

<table>

<tr>

<th>Operator</th>

<th>Description</th>

</tr>

<tr>

<th colspan="2">Orderby System Query Option</th>

</tr>

<tr>

<td>Сортировка</td>

<td>?$orderby=Name</td>

</tr>

<tr>

<td>Сортировка по убыванию</td>

<td>?$orderby=Name desc</td>

</tr>

<tr>

<td>Сортировка по возростанию</td>

<td>?$orderby=Name asc</td>

</tr>

<tr>

<th colspan="2">Top System Query Option</th>

</tr>

<tr>

<td>Первые 5 элементов</td>

<td>?$top=5</td>

</tr>

<tr>

<td>Первые 5 элементов, с сортировкой</td>

<td>?$top=5&$orderby=Price desc</td>

</tr>

<tr>

<th colspan="2">Skip System Query Option</th>

</tr>

<tr>

<td>Продукты, начиная с 3-го</td>

<td>?$skip=2</td>

</tr>

<tr>

<td>Комплекстный запрос</td>

<td>?$skip=2&$top=2&$orderby=Price</td>

</tr>

</table>

</div>

</div>

<div style="float: left; margin-left: 50px;">

<span>Примеры запросов filter:</span>

<table>

<tr>

<th>Operator</th>

<th>Description</th>

<th>Example</th>

</tr>

<tr>

<th colspan="3">Logical Operators</th>

</tr>

<tr>

<td>eq</td>

<td>Equal</td>

<td>?$filter=Name eq 'House'</td>

</tr>

<tr>

<td>ne</td>

<td>Not equal</td>

<td>?$filter=Name ne 'House'</td>

</tr>

<tr>

<td>gt</td>

<td>Greater than</td>

<td>?$filter=Price gt 1400</td>

</tr>

<tr>

<td>ge</td>

<td>Greater than or equal</td>

<td>?$filter=Price ge 1400</td>

</tr>

<tr>

<td>lt</td>

<td>

Less than

<td>?$filter=Price lt 4000</td>

</tr>

<tr>

<td>le</td>

<td>Less than or equal</td>

<td>?$filter=Price le 4000</td>

</tr>

<tr>

<td>and</td>

<td>Logical and</td>

<td>?$filter=Price le 10000 and Price gt 1000</td>

</tr>

<tr>

<td>or</td>

<td>Logical or</td>

<td>?$filter=Price le 100 or Price gt 5000</td>

</tr>

<tr>

<td>not</td>

<td>Logical negation</td>

<td>?$filter=not endswith(Name,'e')</td>

</tr>

<tr>

<th colspan="3">Arithmetic Operators</th>

</tr>

<tr>

<td>add </td>

<td>Addition</td>

<td>?$filter=Price add 5 gt 45</td>

</tr>

<tr>

<td>sub</td>

<td>Subtraction</td>

<td>?$filter=Price sub 5 lt 45</td>

</tr>

<tr>

<td>mul</td>

<td>Multiplication</td>

<td>?$filter=Price mul 1.5 gt 1000</td>

</tr>

<tr>

<td>div</td>

<td>Division</td>

<td>?$filter=Price div 5 gt 4</td>

</tr>

<tr>

<td>mod</td>

<td>Modulo</td>

<td>?$filter=Price mod 2 ne 0</td>

</tr>

<tr>

<th colspan="3">Grouping Operators</th>

</tr>

<tr>

<td>( )</td>

<td>Precedence grouping</td>

<td>?$filter=((Price sub 5) div 10) gt 10</td>

</tr>

</table>

</div>

<script src="~/Scripts/jquery-1.7.1.min.js"></script>

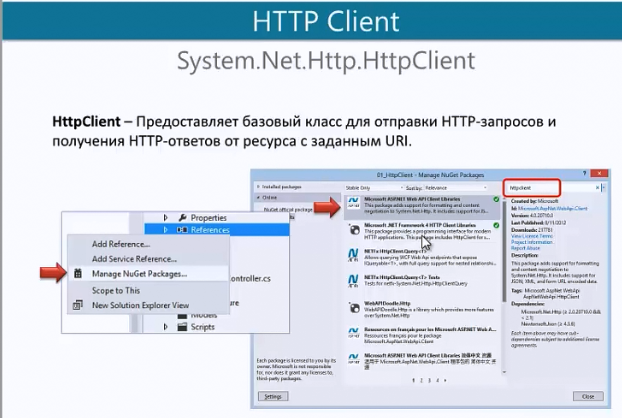
<script src="~/Scripts/CustomScripts/getProducts.js"></script>

<table style="width: 20px;"></table>

</body>

</html>

## Http Client



Model

public class TasksDataSource

{

private static List<Task> \_tasks = null;

public static List<Task> All

{

get

{

if (\_tasks == null)

{

\_tasks = new List<Task>();

\_tasks.Add(new Task() { ID = 1, Name = "Купить молоко", IsCompleted = false });

\_tasks.Add(new Task() { ID = 2, Name = "Спасти мир", IsCompleted = false });

\_tasks.Add(new Task() { ID = 3, Name = "Изучить MVC Web API", IsCompleted = false });

}

return \_tasks;

}

}

Controller

namespace HttpClientSample.Controllers

{

public class TasksController : ApiController

{

// Получение всех задач

public IEnumerable<Task> Get()

{

return TasksDataSource.All;

}

// Создание новой задачи

public HttpResponseMessage Post([FromBody]Task task)

{

try

{

int id = TasksDataSource.All.Max(x => x.ID);

task.ID = id + 1;

TasksDataSource.All.Add(task);

return Request.CreateResponse(HttpStatusCode.Created, task);

}

catch (Exception)

{

return Request.CreateResponse(HttpStatusCode.InternalServerError);

}

}

// Обновление задачи по id

public HttpResponseMessage Put(int id, [FromBody]Task task)

{

Task oldTask = TasksDataSource.All.Where(x => x.ID == id).FirstOrDefault();

if (oldTask == null)

{

return Request.CreateResponse(HttpStatusCode.NotFound);

}

try

{

oldTask.Name = task.Name;

oldTask.IsCompleted = task.IsCompleted;

return Request.CreateResponse(HttpStatusCode.OK, task);

}

catch (Exception)

{

return Request.CreateResponse(HttpStatusCode.InternalServerError);

}

}

// Удаление задачи по id

public HttpResponseMessage Delete(int id)

{

Task toDelete = TasksDataSource.All.Where(x => x.ID == id).FirstOrDefault();

if (toDelete == null)

{

return Request.CreateResponse(HttpStatusCode.NotFound);

}

try

{

TasksDataSource.All.Remove(toDelete);

return Request.CreateResponse(HttpStatusCode.OK);

}

catch (Exception)

{

return Request.CreateResponse(HttpStatusCode.InternalServerError);

}

}

}

}