Web API Document for Task 2

# **Backend**

Add a model to store the properties for your application. Afterwards, add a class in the Models folder and name it Product and then which you add the following code:

namespace ProductsApp.Models

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public string Category { get; set; }

public decimal Price { get; set; }

}

}

Add an empty Web API 2 Controller and name it ProductsV1Controller. Add the following code:

using ProductStore.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Web.Http;

using System.Web.Http.Cors;

namespace ProductStore.Controllers

{

//[EnableCors(origins: "http://localhost:49345,http://csc123client.azurewebsites.net", headers: "\*", methods: "\*")]

public class ProductsController : ApiController

{

// static readonly IProductRepository repository = new ProductRepository();

Product[] products = new Product[]

{

new Product { Id = 1, Name = "Tomato Soup", Category = "Groceries", Price = 1 },

new Product { Id = 2, Name = "Yo-yo", Category = "Toys", Price = 3.75M },

new Product { Id = 3, Name = "Hammer", Category = "Hardware", Price = 16.99M }

};

//code for version 1

[HttpGet]

[Route("api/v1/products/version")]

//http://localhost:9000/api/v1/products/version

public string[] GetVersion()

{

return new string[]

{"hello",

"version 2",

"2"

};

}

[HttpGet]

[Route("api/v1/products/message/")]

//http://localhost:9000/api/v1/products/message?name1=ji&name2=jii1&name3=ji3

public HttpResponseMessage GetMultipleNames(String name1, string name2, string name3)

{

var response = new HttpResponseMessage();

response.Content = new StringContent("<html><body>Hello World " +

" name1 =" + name1 +

" name2= " + name2 +

" name3=" + name3 +

" is provided in path parameter</body></html>");

response.Content.Headers.ContentType = new MediaTypeHeaderValue("text/html");

return response;

}

[HttpGet]

[Route("api/v1/products")]

//http://localhost:9000/api/v1/products

public IEnumerable<Product> GetAllProducts()

{

return products;

}

[HttpGet]

[Route("api/v1/products/{id:int:min(2)}")]

//http://localhost:9000/api/v1/products/3

public IHttpActionResult GetProduct(int id)

{

var product = products.FirstOrDefault((p) => p.Id == id);

if (product == null)

{

return NotFound();

}

return Ok(product);

}

}

}

## **FRONTEND**

AJAX to call the web API:

$(document).ready(function () {

// Send an AJAX request

$.getJSON(uri)

.done(function (data) {

// On success, 'data' contains a list of products.

$.each(data, function (key, item) {

// Add a list item for the product.

$('<li>', { text: formatItem(item) }).appendTo($('#products'));

});

});

});

function formatItem(item) {

return item.Name + ': $' + item.Price;

}

function find() {

var id = $('#prodId').val();

$.getJSON(uri + '/' + id)

.done(function (data) {

$('#product').text(formatItem(data));

})

.fail(function (jqXHR, textStatus, err) {

$('#product').text('Error: ' + err);

});

}

</script>

</body>

</html>

Getting a List of Products

$(document).ready(function () {

// Send an AJAX request

$.getJSON(apiUrl)

.done(function (data) {

// On success, 'data' contains a

list of products.

$.each(data, function (key, item) {

// Add a list item for the product.

$('<li>', { text: formatItem(item) }).appendTo($('#products'));

});

});

});

Getting a Product By ID

function find() {

var id = $('#prodId').val();

$.getJSON(apiUrl + '/' + id)

.done(function (data) {

$('#product').text(formatItem(data));

})

.fail(function (jqXHR,

textStatus, err) {

$('#product').text('Error: ' + err);

});

}

## Enabling CRUD Operations

|  |  |  |
| --- | --- | --- |
| HTTP Method | Relative URI | Action |
| GET | /api/v2/products | Get a list of all products |
| GET | /api/v2/products | Get a product by ID |
| GET | /api/v2/products | Get a product by category |
| POST | /api/v2/products | Create a new product |
| PUT | /api/v2/products | Update a product |
| DELETE | /api/v2/products | Delete a product |

## **BACKEND**

}

Add a repository store a collection of the products. Then, create a repository named IProductRepository in Models folder and then you add the following code:

namespace ProductStore.Models

{

public interface IProductRepository

{

IEnumerable<Product> GetAll();

Product Get(int id);

Product Add(Product item);

void Remove(int id);

bool Update(Product item);

}

}

Create another class in Models folder called ProductRepository and then you add the following code:

namespace ProductStore.Models

{

public class ProductRepository : IProductRepository

{

private List<Product> products = new List<Product>();

private int \_nextId = 1;

public ProductRepository()

{

Add(new Product { Name = "Tomato soup", Category = "Groceries", Price = 1.39M });

Add(new Product { Name = "Yo-yo", Category = "Toys", Price = 3.75M });

Add(new Product { Name = "Hammer", Category = "Hardware", Price = 16.99M });

}

public IEnumerable<Product> GetAll()

{

return products;

}

public Product Get(int id)

{

return products.Find(p => p.Id == id);

}

public Product Add(Product item)

{

if (item == null)

{

throw new ArgumentNullException("item");

}

item.Id = \_nextId++;

products.Add(item);

return item;

}

public void Remove(int id)

{

products.RemoveAll(p => p.Id == id);

}

public bool Update(Product item)

{

if (item == null)

{

throw new ArgumentNullException("item");

}

int index = products.FindIndex(p => p.Id == item.Id);

if (index == -1)

{

return false;

}

products.RemoveAt(index);

products.Add(item);

return true;

}

}

}

Add a controller called ProductsController in Controllers folder named ProductsV2Controller and then add the following code:

public class ProductsController : ApiController

{

static readonly IProductRepository repository = new ProductRepository();

}

ublic class ProductsController : ApiController

{

public IEnumerable<Product> GetAllProducts()

{

return repository.GetAll();

}

// ....

}

public Product GetProduct(int id)

{

Product item = repository.Get(id);

if (item == null)

{

throw new HttpResponseException(HttpStatusCode.NotFound);

}

return item;

}

Creating a resource:

// Not the final implementation!

public Product PostProduct(Product item)

{

item = repository.Add(item);

return item;

}

Updating a resource

public void PutProduct(int id, Product product)

{

product.Id = id;

if (!repository.Update(product))

{

throw new HttpResponseException(HttpStatusCode.NotFound);

Deleting a resource:

public void DeleteProduct(int id)

{

Product item = repository.Get(id);

if (item == null)

{

throw new HttpResponseException(HttpStatusCode.NotFound);

}

repository.Remove(id);

}