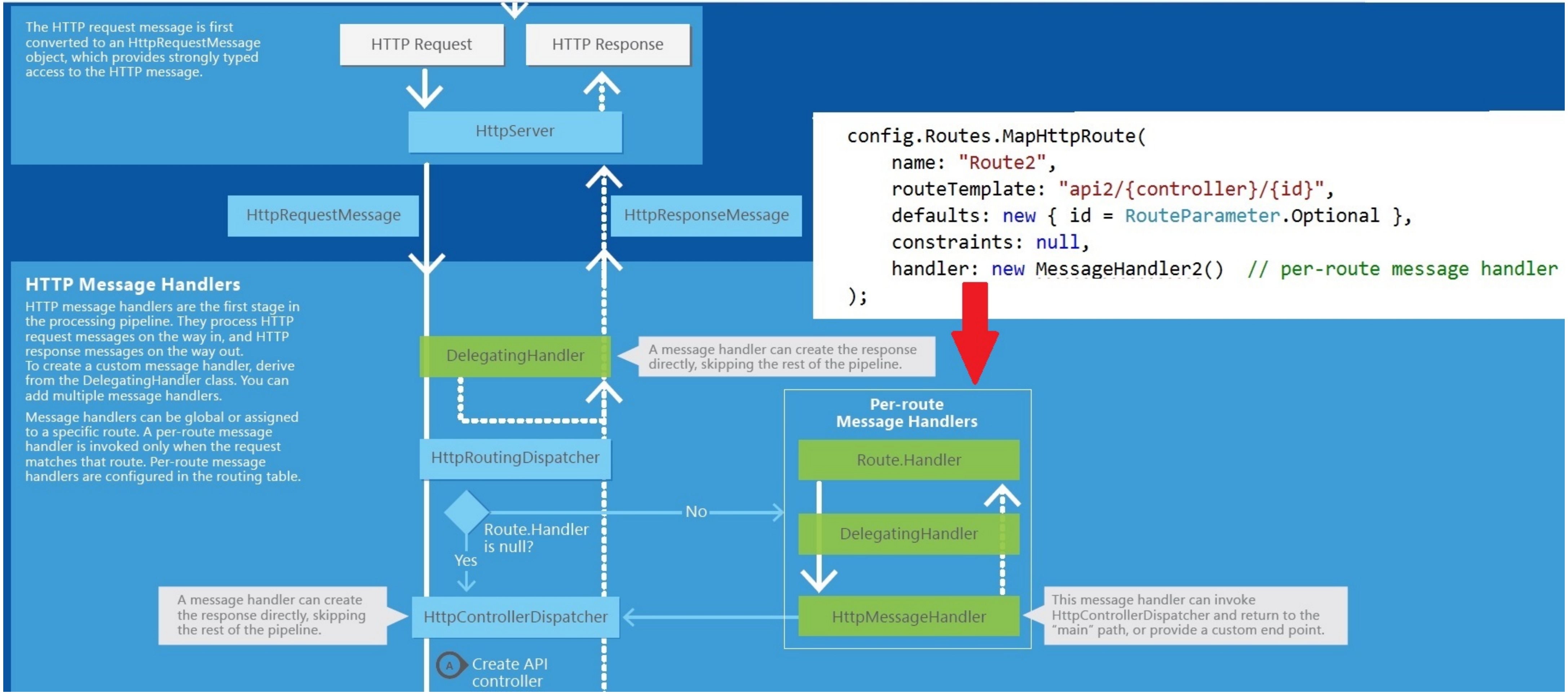
HTTP Message Handlers



The HTTP request message is first converted to an HttpRequestMessage object, which provides strongly typed access to the HTTP message.

```
config.Routes.MapHttpRoute(
    name: "Route2",
    routeTemplate: "api2/{controller}/{id}",
    defaults: new { id = RouteParameter.Optional },
    constraints: null,
    handler: new MessageHandler2()  // per-route message handler
);
```

HTTP Message Handlers

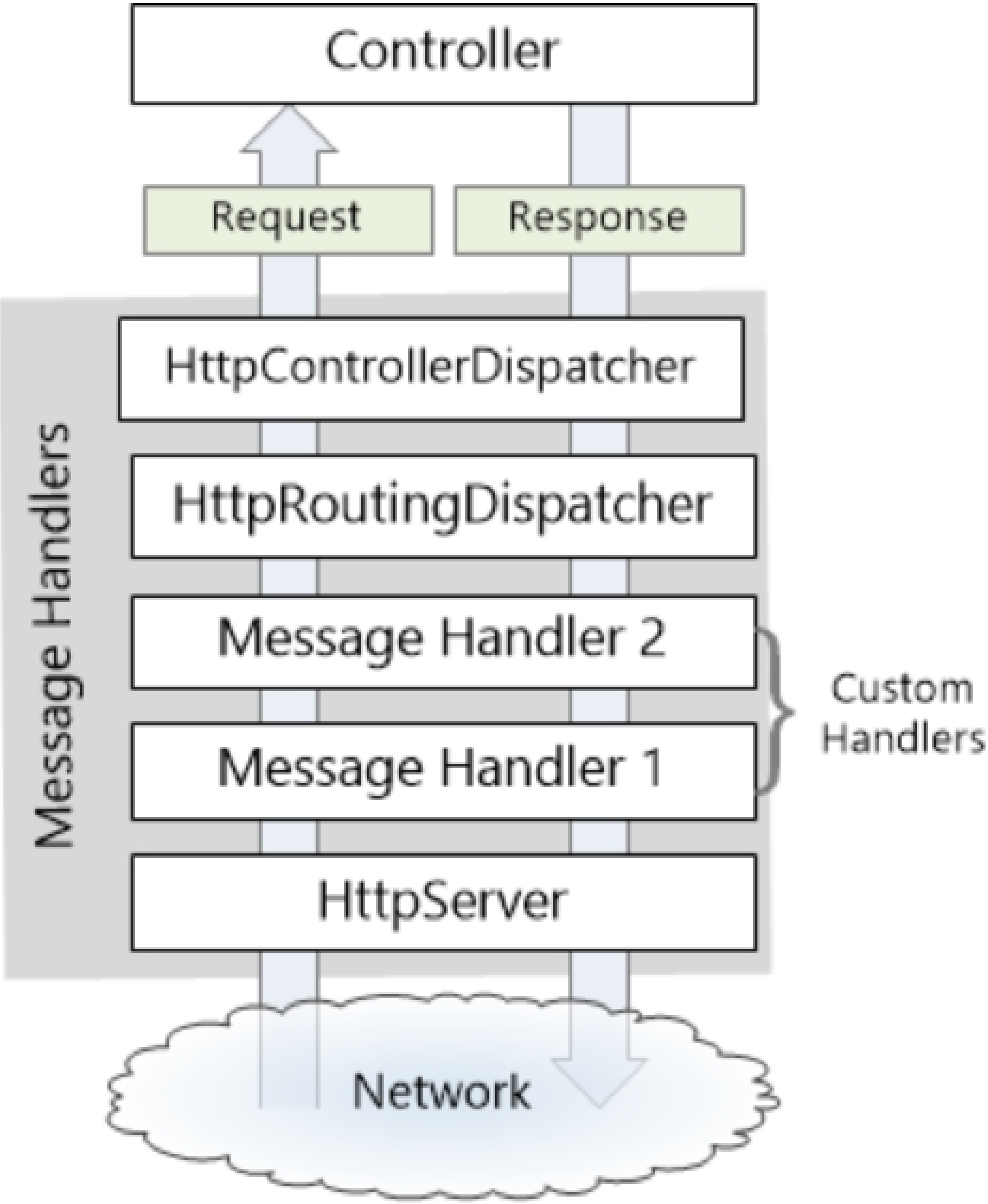
HTTP message handlers are the first stage in the processing pipeline. They process HTTP request messages on the way in, and HTTP response messages on the way out.
To create a custom message handler, derive from the DelegatingHandler class. You can add multiple message handlers.

Message handlers can be global or assigned to a specific route. A per-route message handler is invoked only when the request matches that route. Per-route message handlers are configured in the routing table.

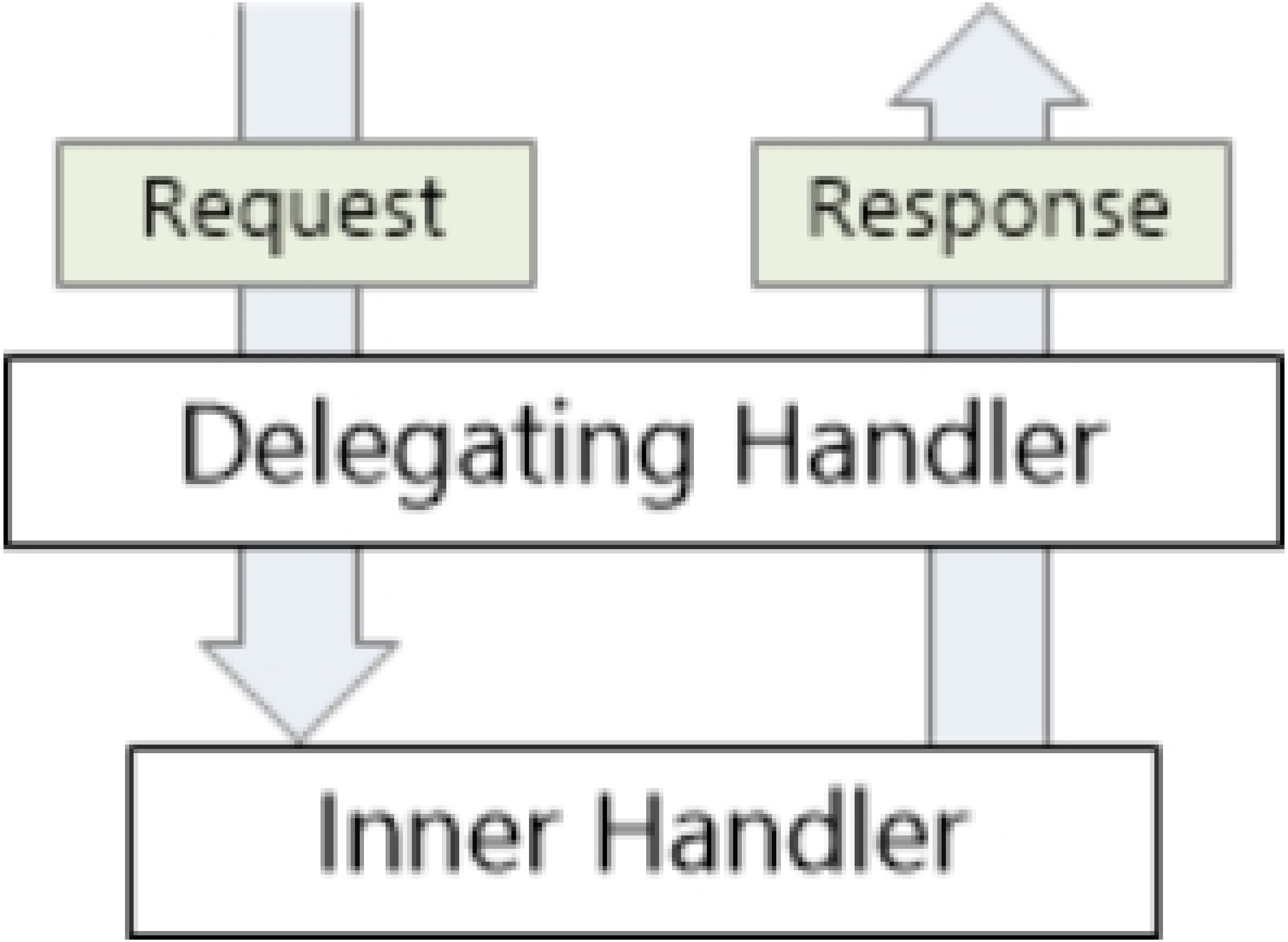
A message handler can create the response directly, skipping the rest of the pipeline.

A message handler can create the response directly, skipping the rest of the pipeline.

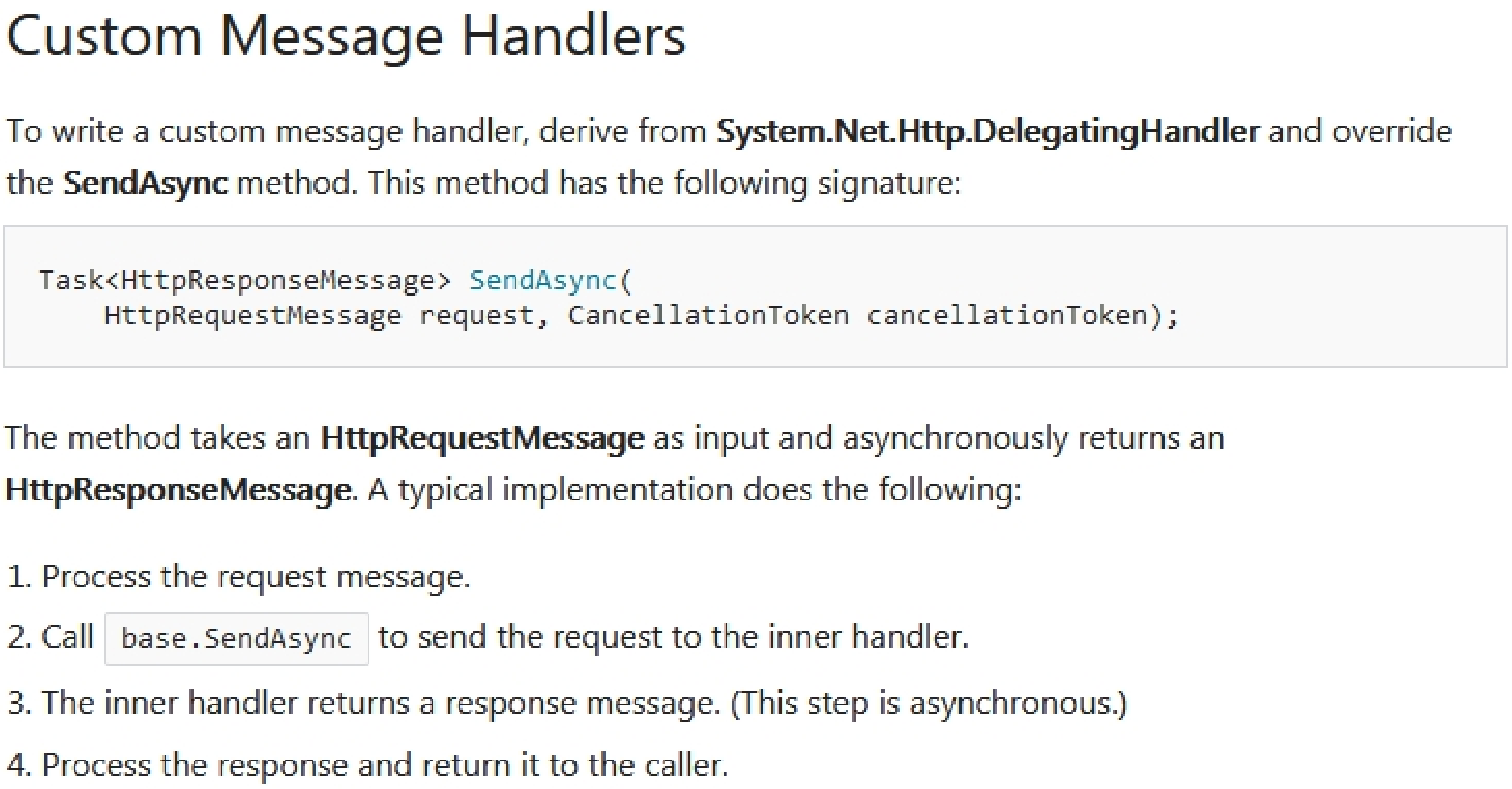
This message handler can invoke HttpControllerDispatcher and return to the “main” path, or provide a custom end point.



Server-Side Handlers



Request
Response
Delegating Handler
Inner Handler



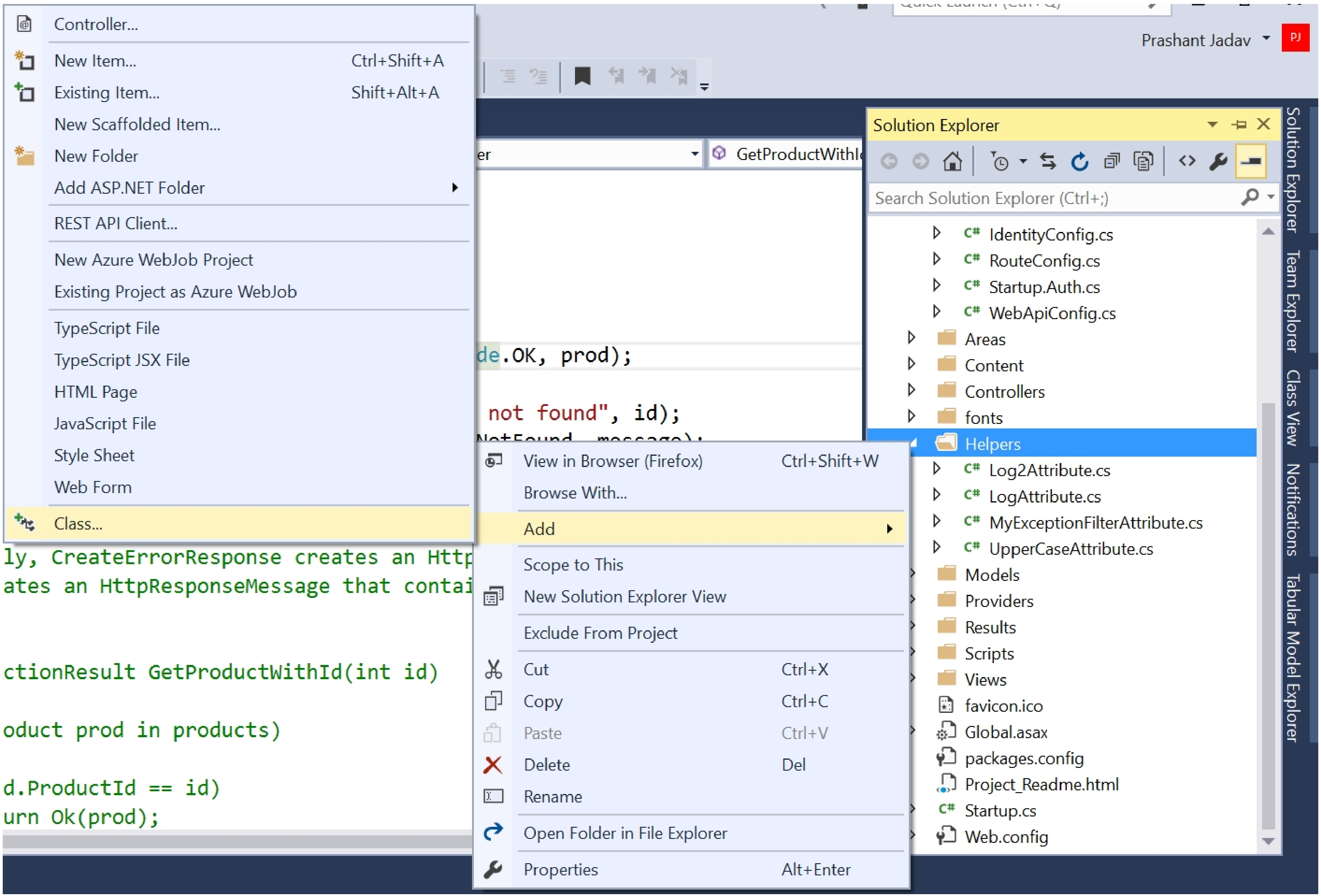
Custom Message Handlers

To write a custom message handler, derive from **System.Net.Http.DelegatingHandler** and override the **SendAsync** method. This method has the following signature:

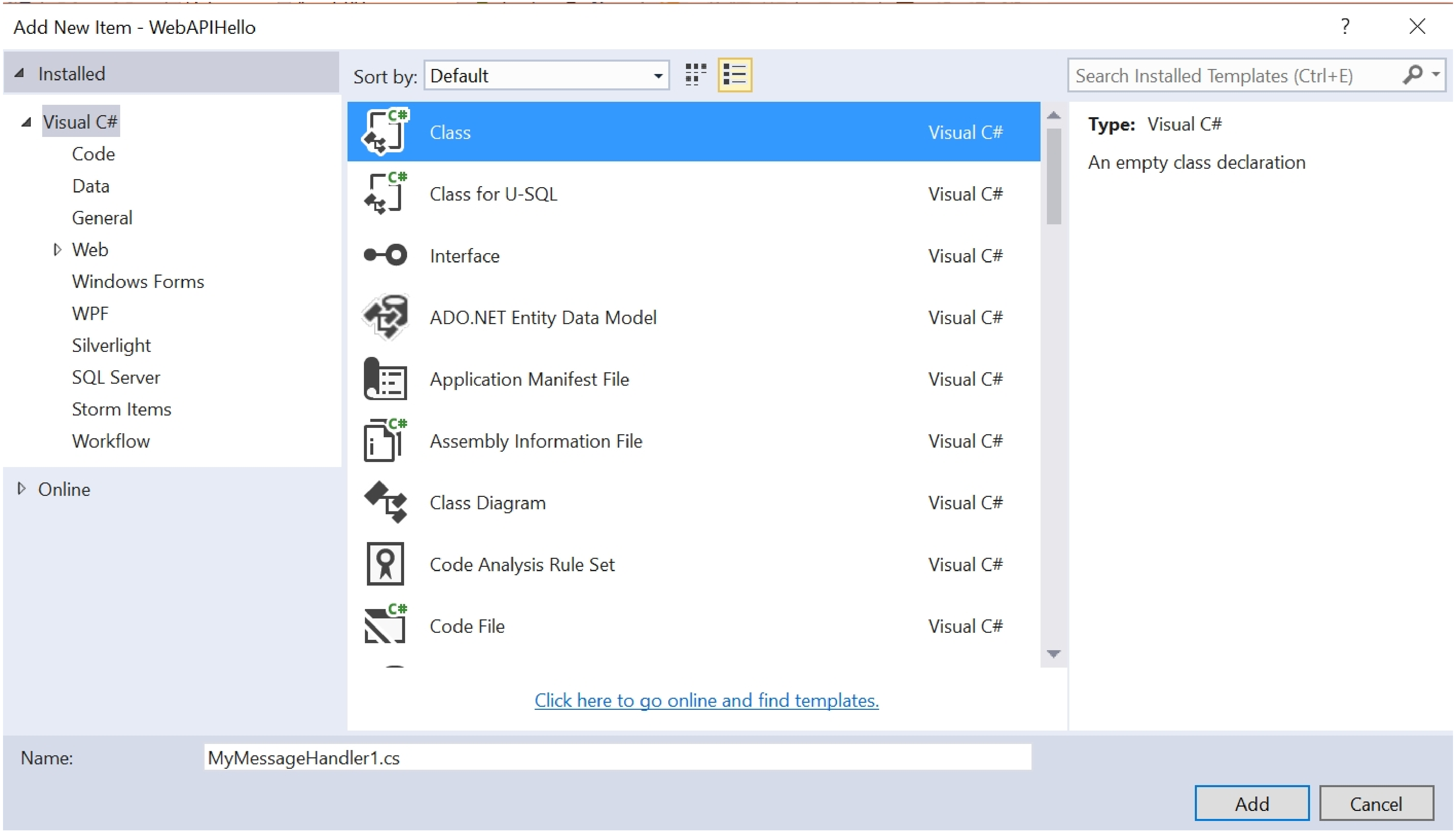
```
Task<HttpResponseMessage> SendAsync(
    HttpRequestMessage request, CancellationToken cancellationToken);
```

The method takes an **HttpRequestMessage** as input and asynchronously returns an **HttpResponseMessage**. A typical implementation does the following:

1. Process the request message.
2. Call `base.SendAsync` to send the request to the inner handler.
3. The inner handler returns a response message. (This step is asynchronous.)
4. Process the response and return it to the caller.



```
de.OK, prod);

 not found", id);
NotFound, message);

ly, CreateErrorResponse creates an Http
ates an HttpResponseMessage that contai


ctionResult GetProductWithId(int id)

oduct prod in products)

d.ProductId == id)
urn Ok(prod);
```



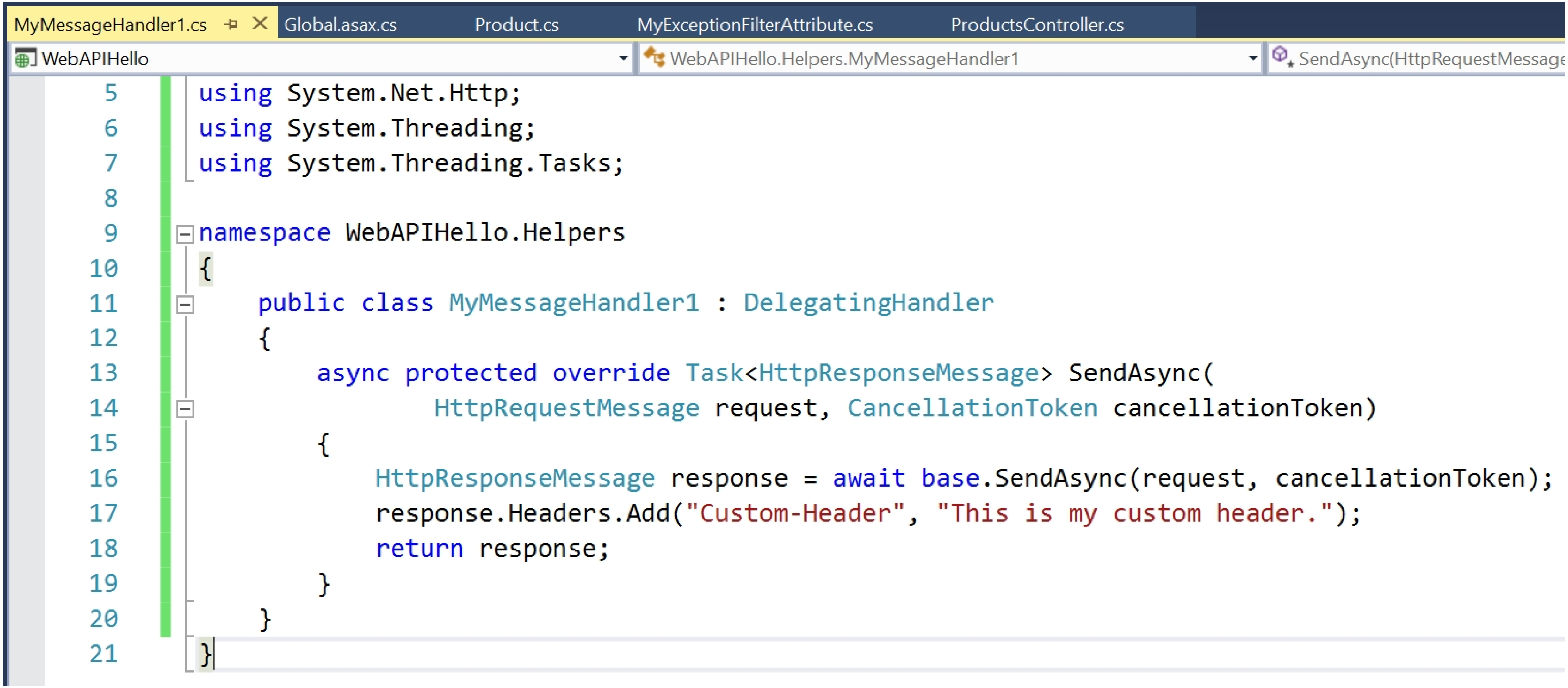
Sort by: Default

Class Visual C#

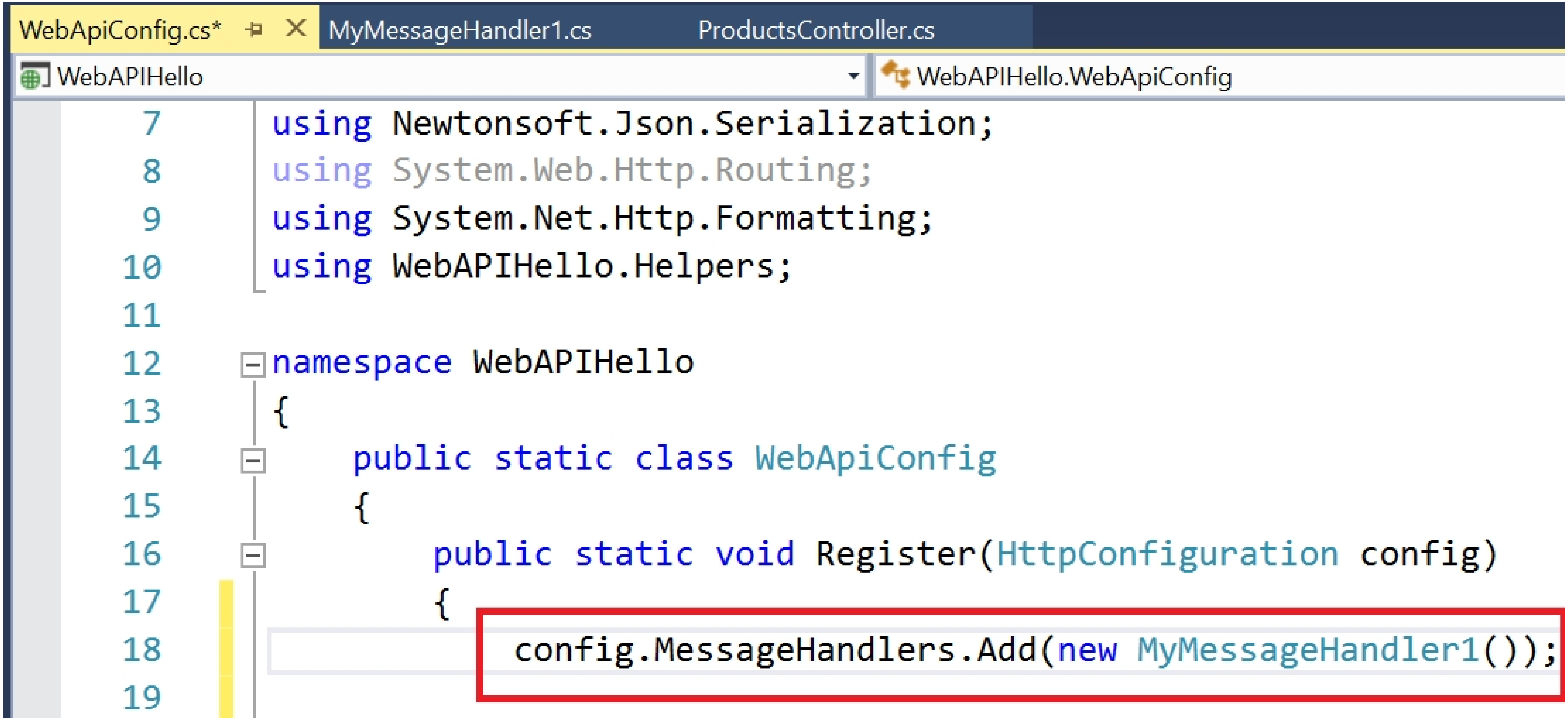
Type: Visual C#

An empty class declaration

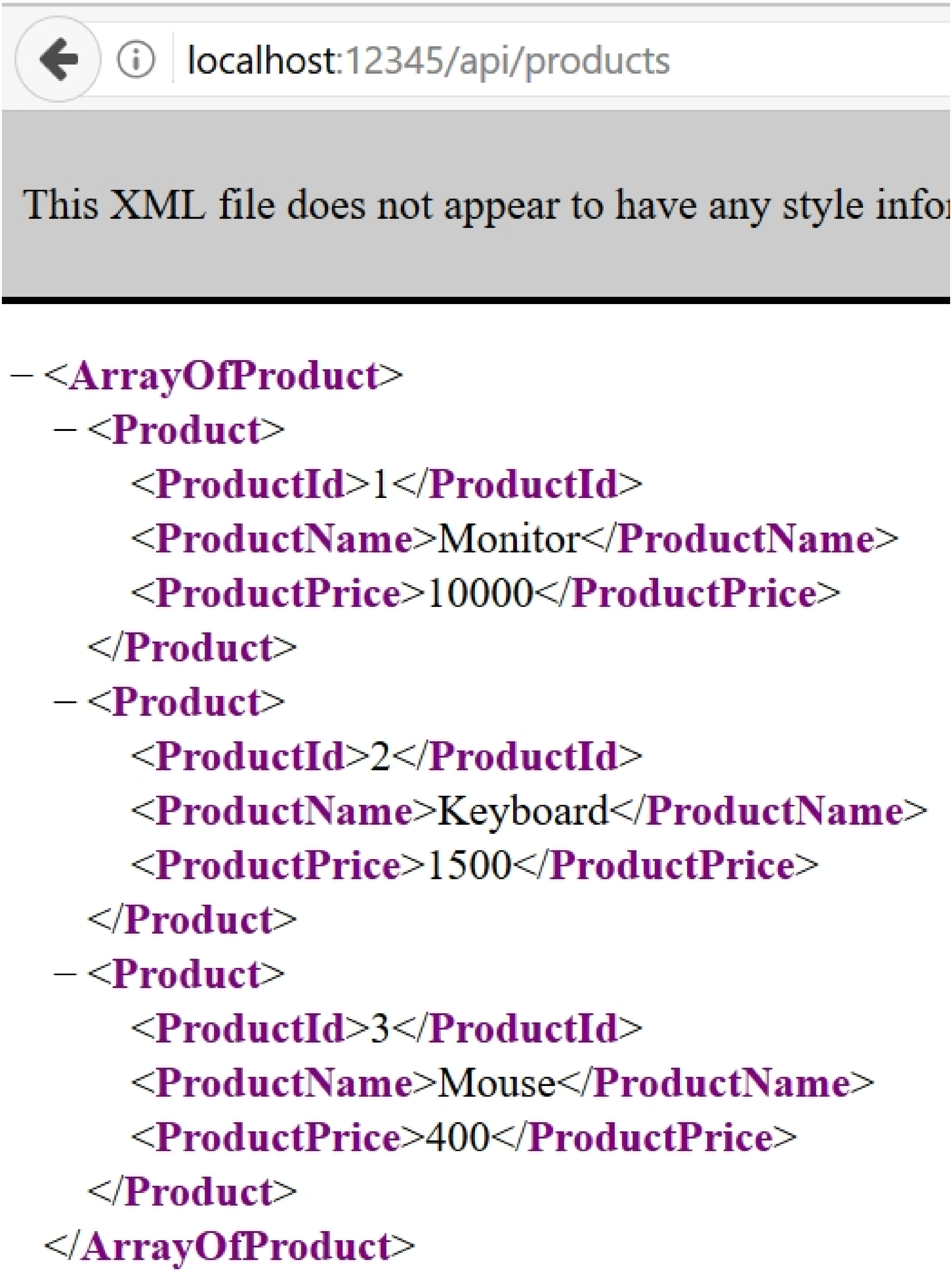
Name: MyMessageHandler1.cs


MyMessageHandler1.cs
Global.asax.cs
Product.cs
MyExceptionFilterAttribute.cs
ProductsController.cs
WebAPIHello
WebAPIHello.Helpers.MyMessageHandler1
SendAsync(HttpRequestMessage

```
using System.Net.Http;
using System.Threading;
using System.Threading.Tasks;

namespace WebAPIHello.Helpers
{
    public class MyMessageHandler1 : DelegatingHandler
    {
        async protected override Task<HttpResponseMessage> SendAsync(
                HttpRequestMessage request, CancellationToken cancellationToken)
        {
            HttpResponseMessage response = await base.SendAsync(request, cancellationToken);
            response.Headers.Add("Custom-Header", "This is my custom header.");
            return response;
        }
    }
}
```



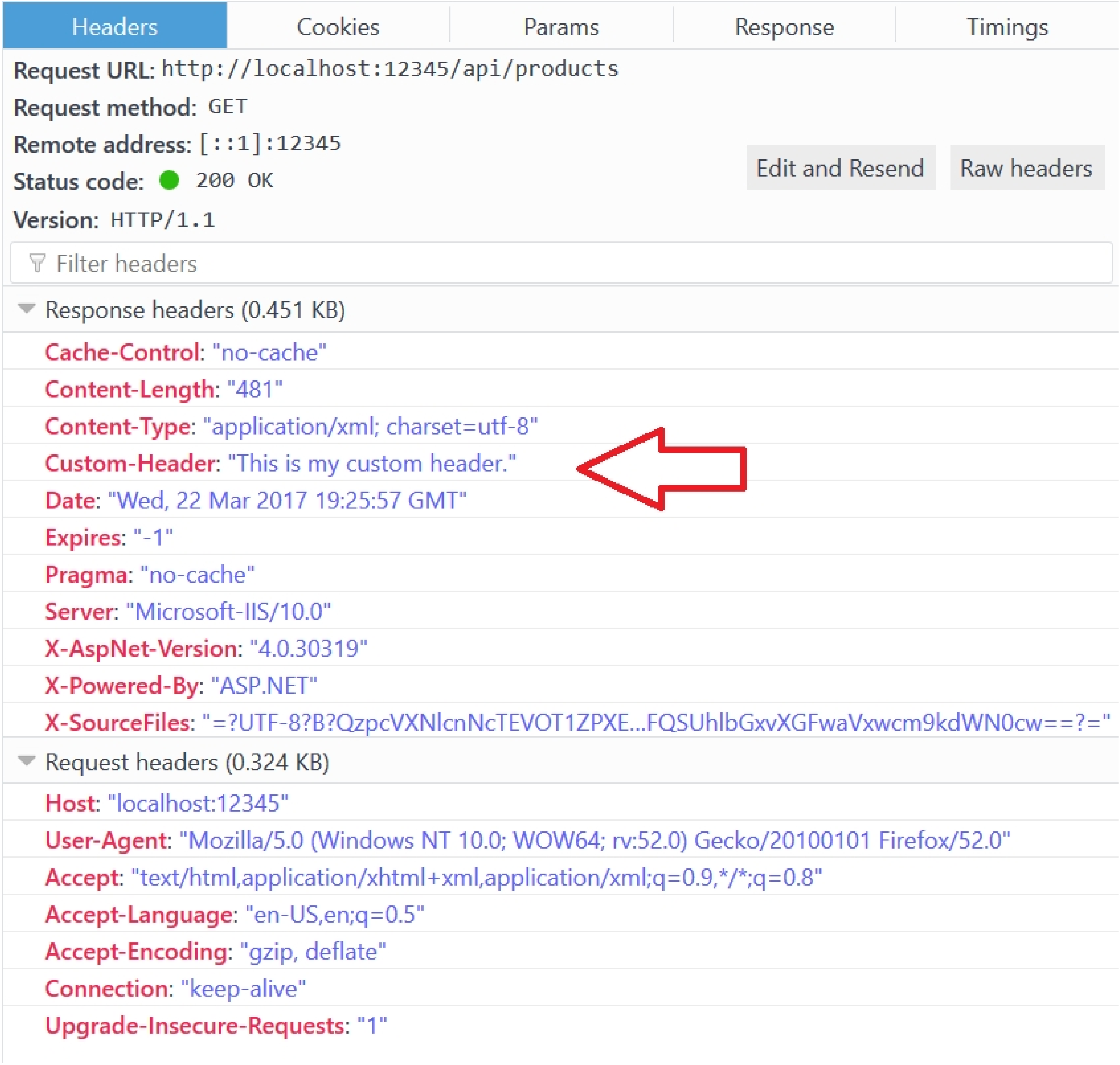
```
using Newtonsoft.Json.Serialization;
using System.Web.Http.Routing;
using System.Net.Http.Formatting;
using WebAPIHello.Helpers;

namespace WebAPIHello
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.MessageHandlers.Add(new MyMessageHandler1());

```



localhost:12345/api/products

This XML file does not appear to have any style infor

```
– <ArrayOfProduct>
  – <Product>
      <ProductId>1</ProductId>
      <ProductName>Monitor</ProductName>
      <ProductPrice>10000</ProductPrice>
    </Product>
  – <Product>
      <ProductId>2</ProductId>
      <ProductName>Keyboard</ProductName>
      <ProductPrice>1500</ProductPrice>
    </Product>
  – <Product>
      <ProductId>3</ProductId>
      <ProductName>Mouse</ProductName>
      <ProductPrice>400</ProductPrice>
    </Product>
  </ArrayOfProduct>
```



Request URL: http://localhost:12345/api/products

Request method: GET

Remote address: [::1]:12345

Status code: ● 200 OK

Version: HTTP/1.1

Response headers (0.451 KB)

Cache-Control: "no-cache"

Content-Length: "481"

Content-Type: "application/xml; charset=utf-8"

Custom-Header: "This is my custom header."

Date: "Wed, 22 Mar 2017 19:25:57 GMT"

Expires: "-1"

Pragma: "no-cache"

Server: "Microsoft-IIS/10.0"

X-AspNet-Version: "4.0.30319"

X-Powered-By: "ASP.NET"

X-SourceFiles: "=?UTF-8?B?QzpcVXNlcnNcTEVOT1ZPXE...FQSUhlbGxvXGFwaVxwcm9kdWN0cw==?="

Request headers (0.324 KB)

Host: "localhost:12345"

User-Agent: "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0"

Accept: "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"

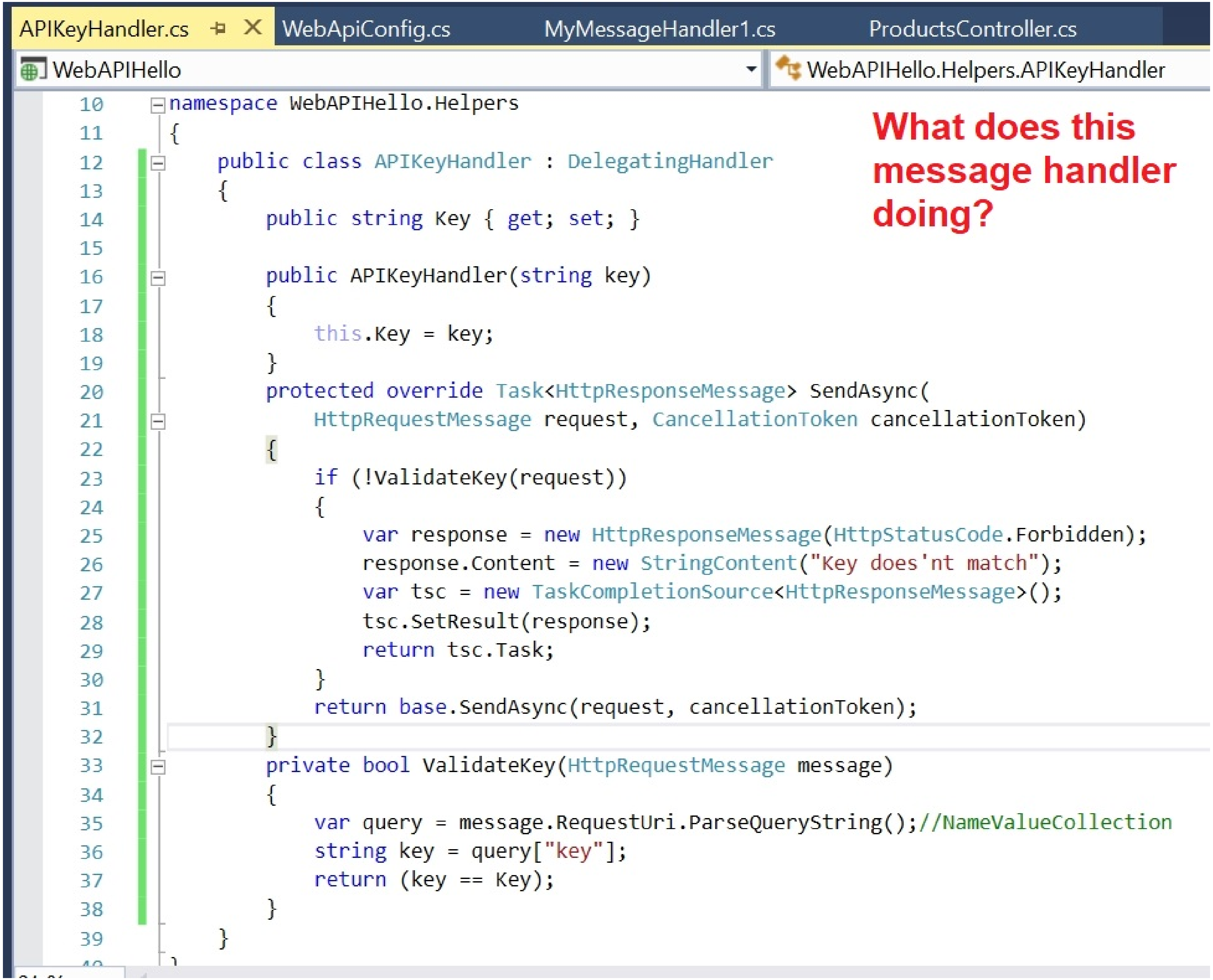
Accept-Language: "en-US,en;q=0.5"

Accept-Encoding: "gzip, deflate"

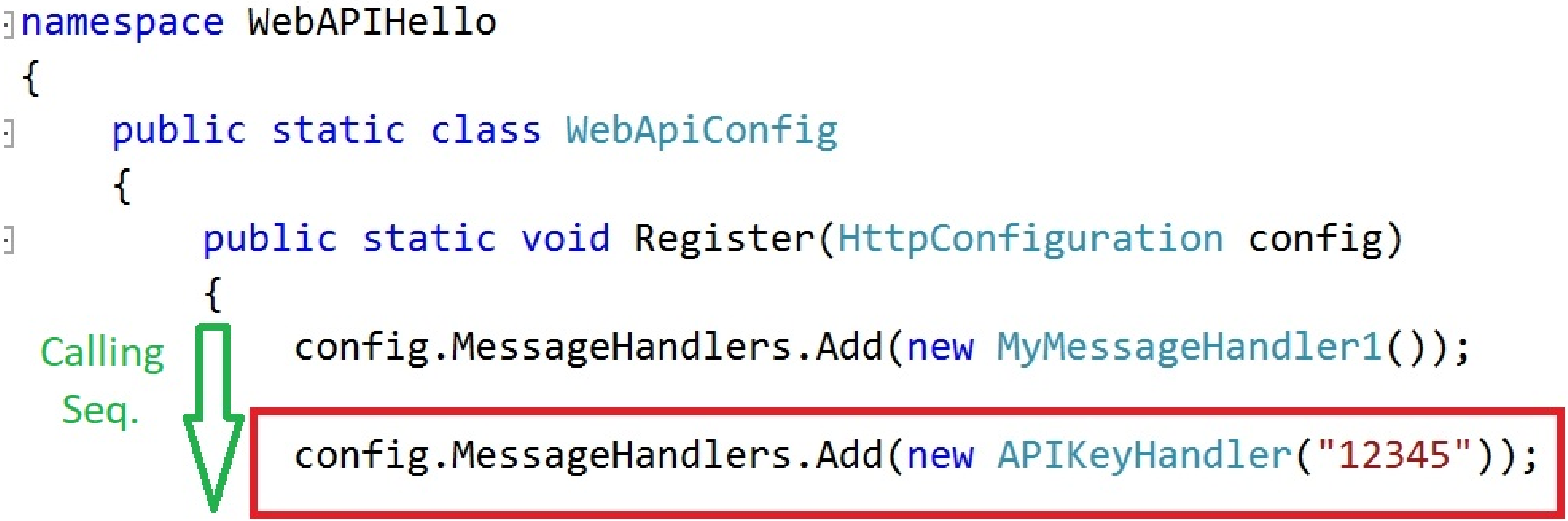
Connection: "keep-alive"

Upgrade-Insecure-Requests: "1"

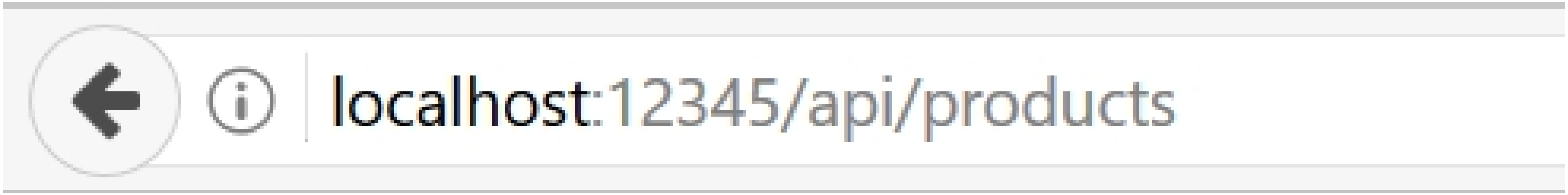
What does this message handler doing?



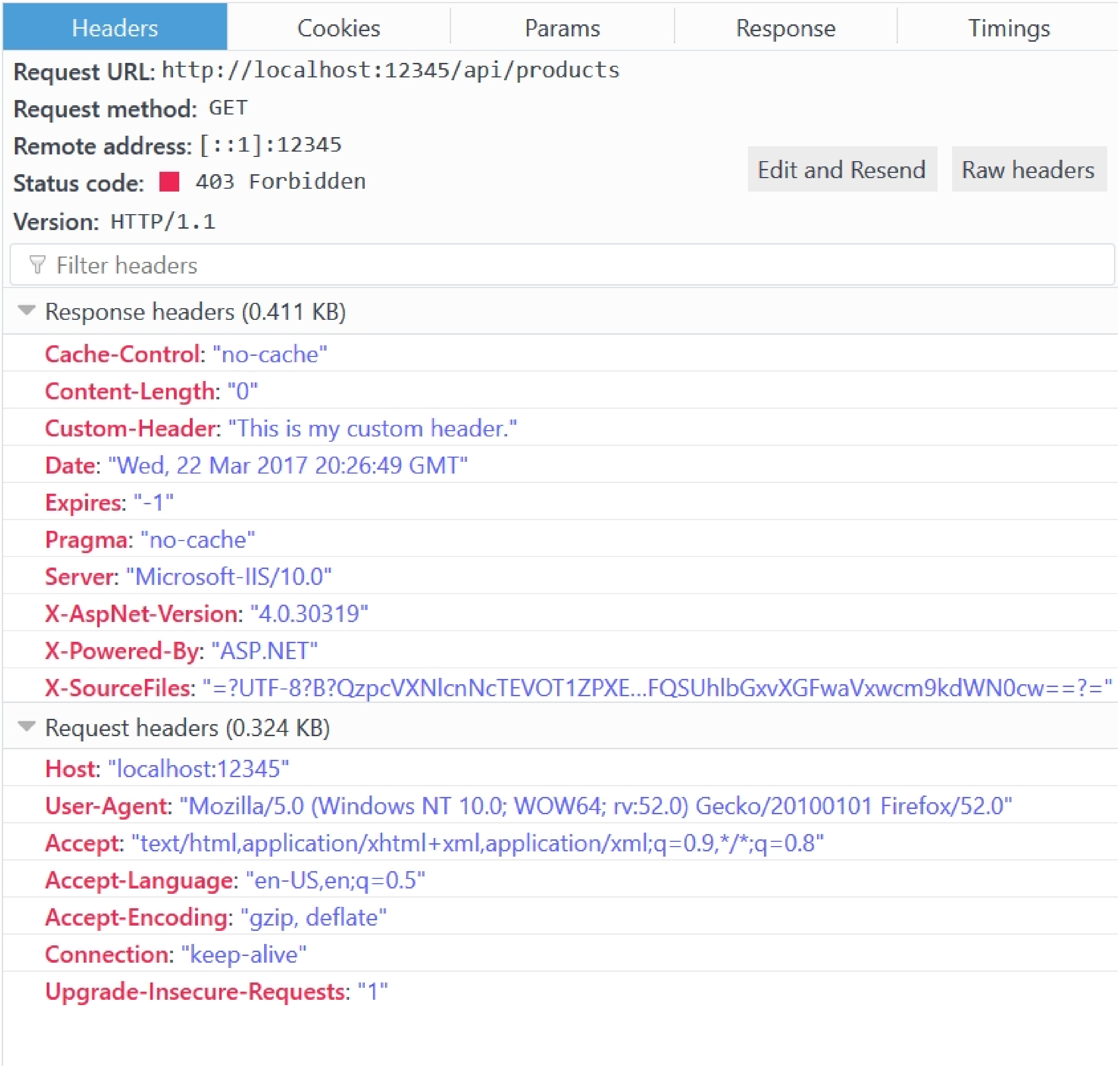
```csharp
namespace WebAPIHello.Helpers
{
    public class APIKeyHandler : DelegatingHandler
    {
        public string Key { get; set; }

        public APIKeyHandler(string key)
        {
            this.Key = key;
        }
        protected override Task<HttpResponseMessage> SendAsync(
            HttpRequestMessage request, CancellationToken cancellationToken)
        {
            if (!ValidateKey(request))
            {
                var response = new HttpResponseMessage(HttpStatusCode.Forbidden);
                response.Content = new StringContent("Key does'nt match");
                var tsc = new TaskCompletionSource<HttpResponseMessage>();
                tsc.SetResult(response);
                return tsc.Task;
            }
            return base.SendAsync(request, cancellationToken);
        }
        private bool ValidateKey(HttpRequestMessage message)
        {
            var query = message.RequestUri.ParseQueryString();//NameValueCollection
            string key = query["key"];
            return (key == Key);
        }
    }
}
```



```
namespace WebAPIHello
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.MessageHandlers.Add(new MyMessageHandler1());

            config.MessageHandlers.Add(new APIKeyHandler("12345"));
```



localhost:12345/api/products



Request URL: http://localhost:12345/api/products

Request method: GET

Remote address: [::1]:12345

Status code: 🟥 403 Forbidden

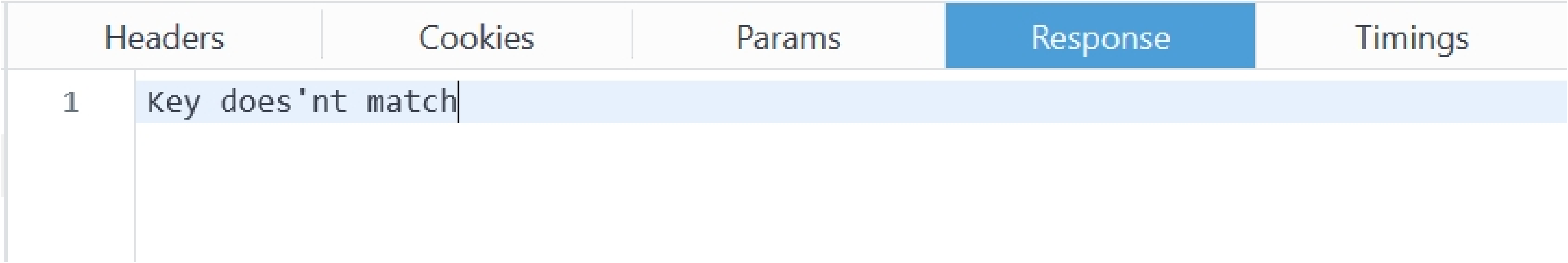
Version: HTTP/1.1

Response headers (0.411 KB)

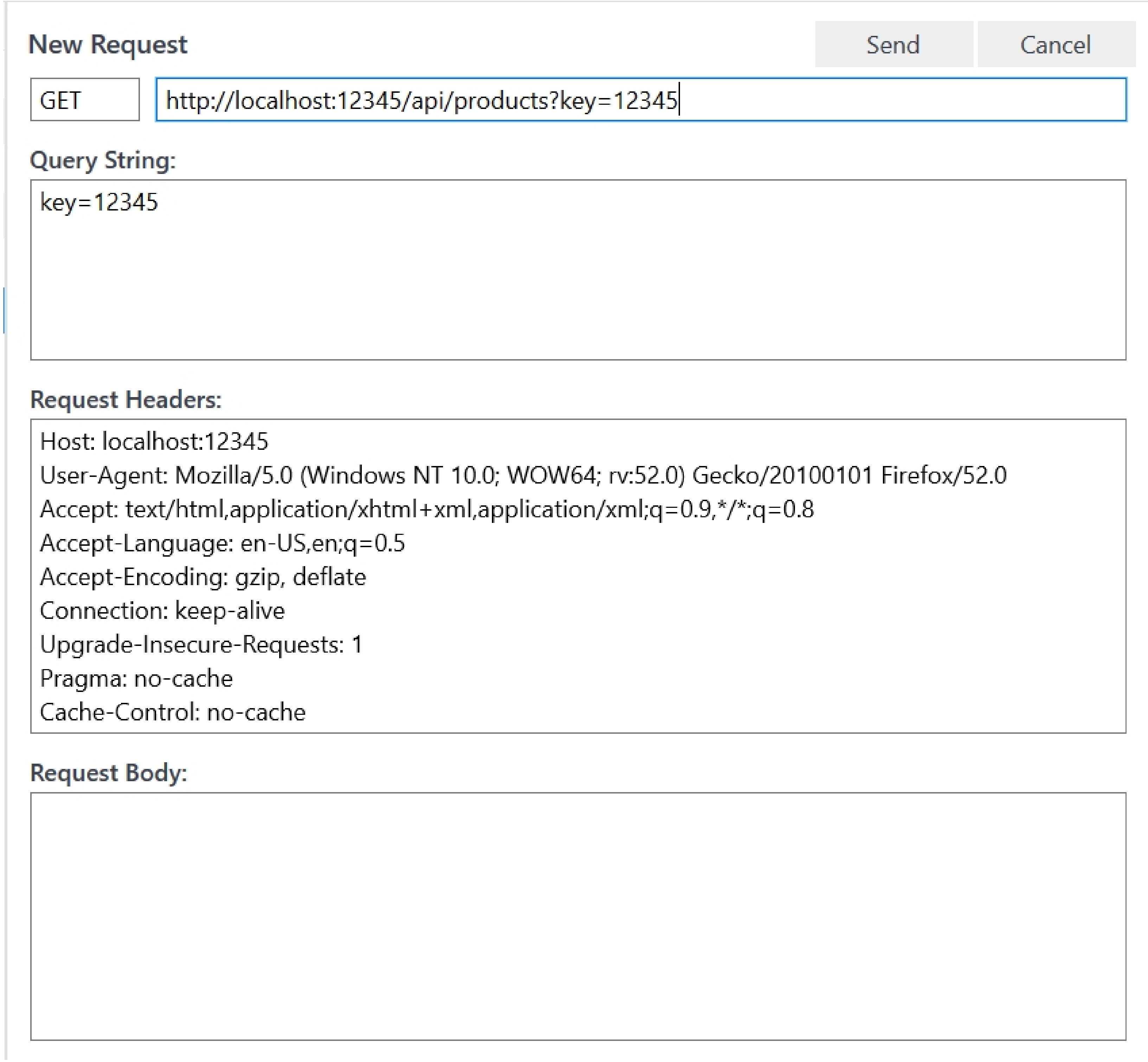
```
Cache-Control: "no-cache"
Content-Length: "0"
Custom-Header: "This is my custom header."
Date: "Wed, 22 Mar 2017 20:26:49 GMT"
Expires: "-1"
Pragma: "no-cache"
Server: "Microsoft-IIS/10.0"
X-AspNet-Version: "4.0.30319"
X-Powered-By: "ASP.NET"
X-SourceFiles: "=?UTF-8?B?QzpcVXNlcnNcTEVOT1ZPXE...FQSUhlbGxvXGFwaVxwcm9kdWN0cw==?="
```

Request headers (0.324 KB)

```
Host: "localhost:12345"
User-Agent: "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0"
Accept: "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
Accept-Language: "en-US,en;q=0.5"
Accept-Encoding: "gzip, deflate"
Connection: "keep-alive"
Upgrade-Insecure-Requests: "1"
```



```
Key does'nt match
```



GET

http://localhost:12345/api/products?key=12345

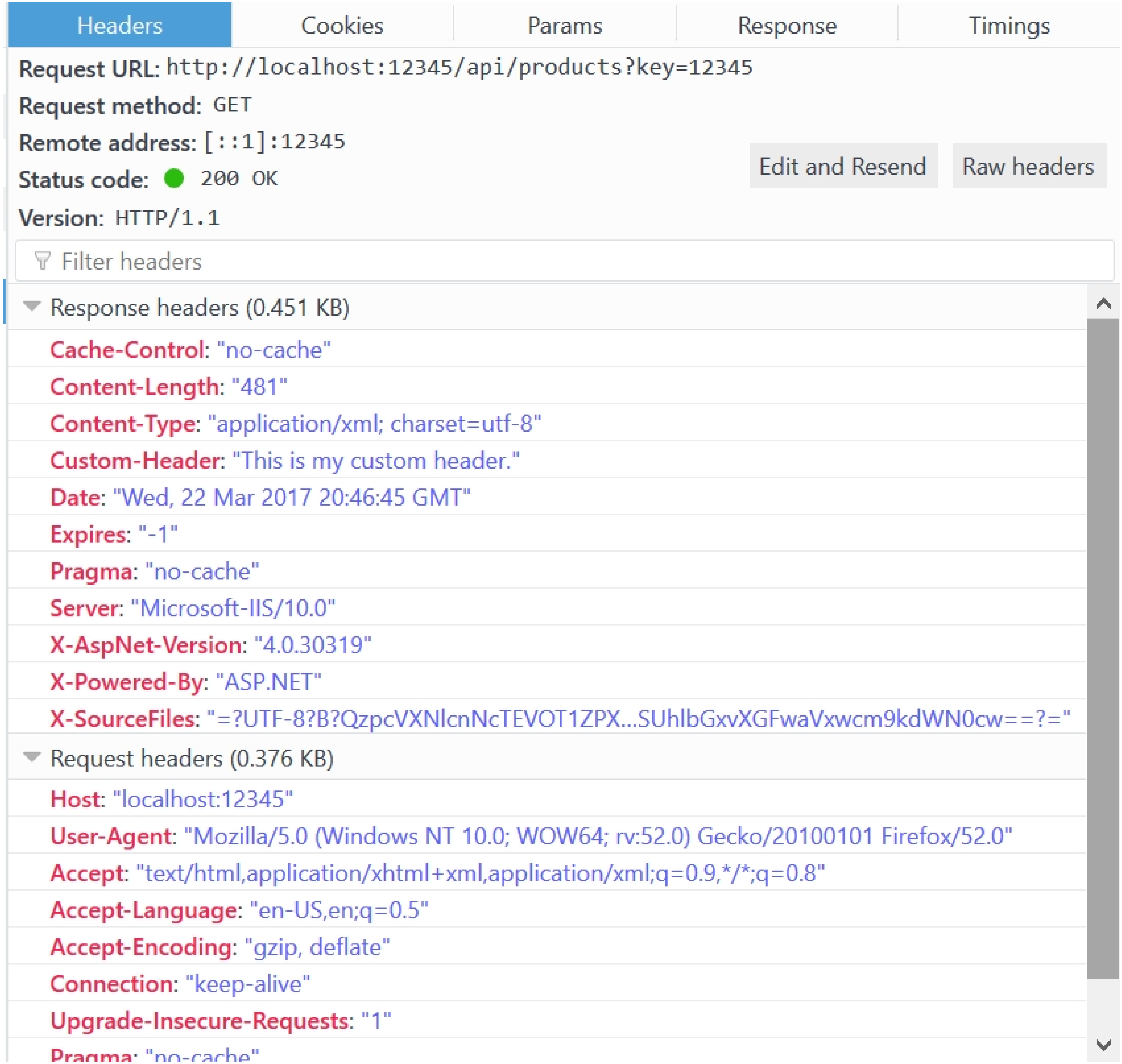
Query String:

key=12345

Request Headers:

```
Host: localhost:12345
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache
```

Request Body:



Request URL: http://localhost:12345/api/products?key=12345

Request method: GET

Remote address: [::1]:12345

Status code: ● 200 OK

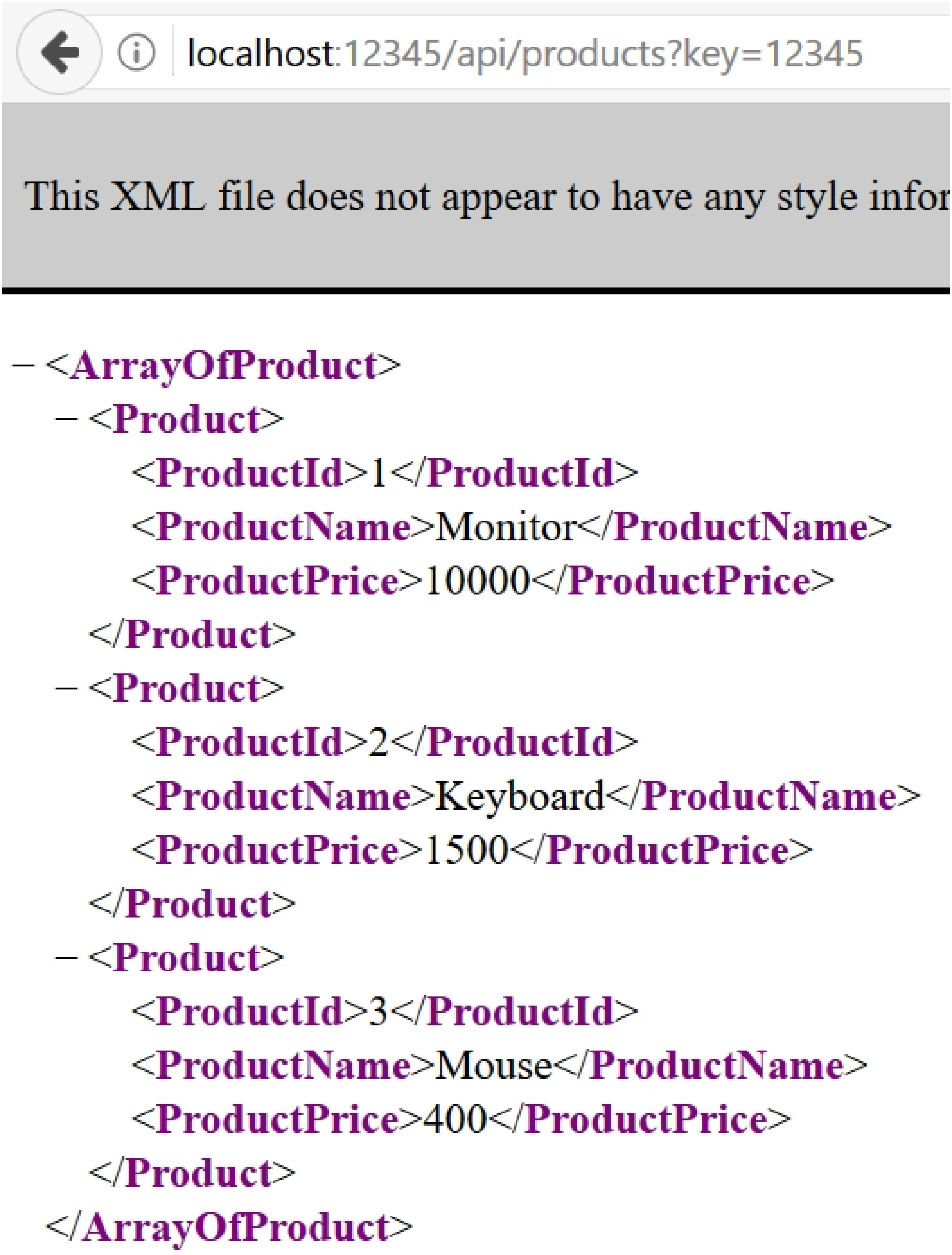
Version: HTTP/1.1

Response headers (0.451 KB)

Cache-Control: "no-cache"
Content-Length: "481"
Content-Type: "application/xml; charset=utf-8"
Custom-Header: "This is my custom header."
Date: "Wed, 22 Mar 2017 20:46:45 GMT"
Expires: "-1"
Pragma: "no-cache"
Server: "Microsoft-IIS/10.0"
X-AspNet-Version: "4.0.30319"
X-Powered-By: "ASP.NET"
X-SourceFiles: "=?UTF-8?B?QzpcVXNlcnNcTEVOT1ZPX...SUhlbGxvXGFwaVxwcm9kdWN0cw==?="

Request headers (0.376 KB)

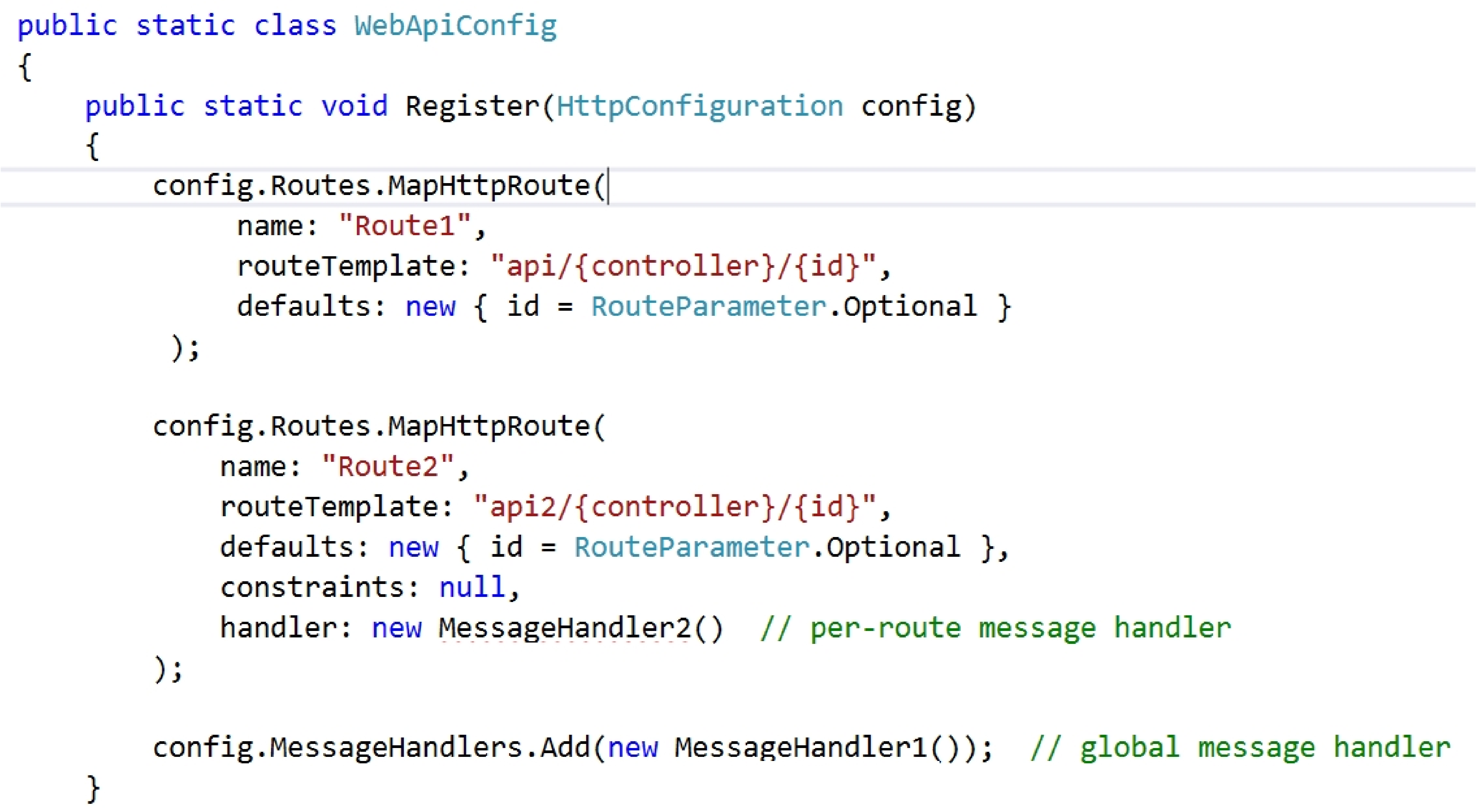
Host: "localhost:12345"
User-Agent: "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0"
Accept: "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
Accept-Language: "en-US,en;q=0.5"
Accept-Encoding: "gzip, deflate"
Connection: "keep-alive"
Upgrade-Insecure-Requests: "1"
Pragma: "no-cache"



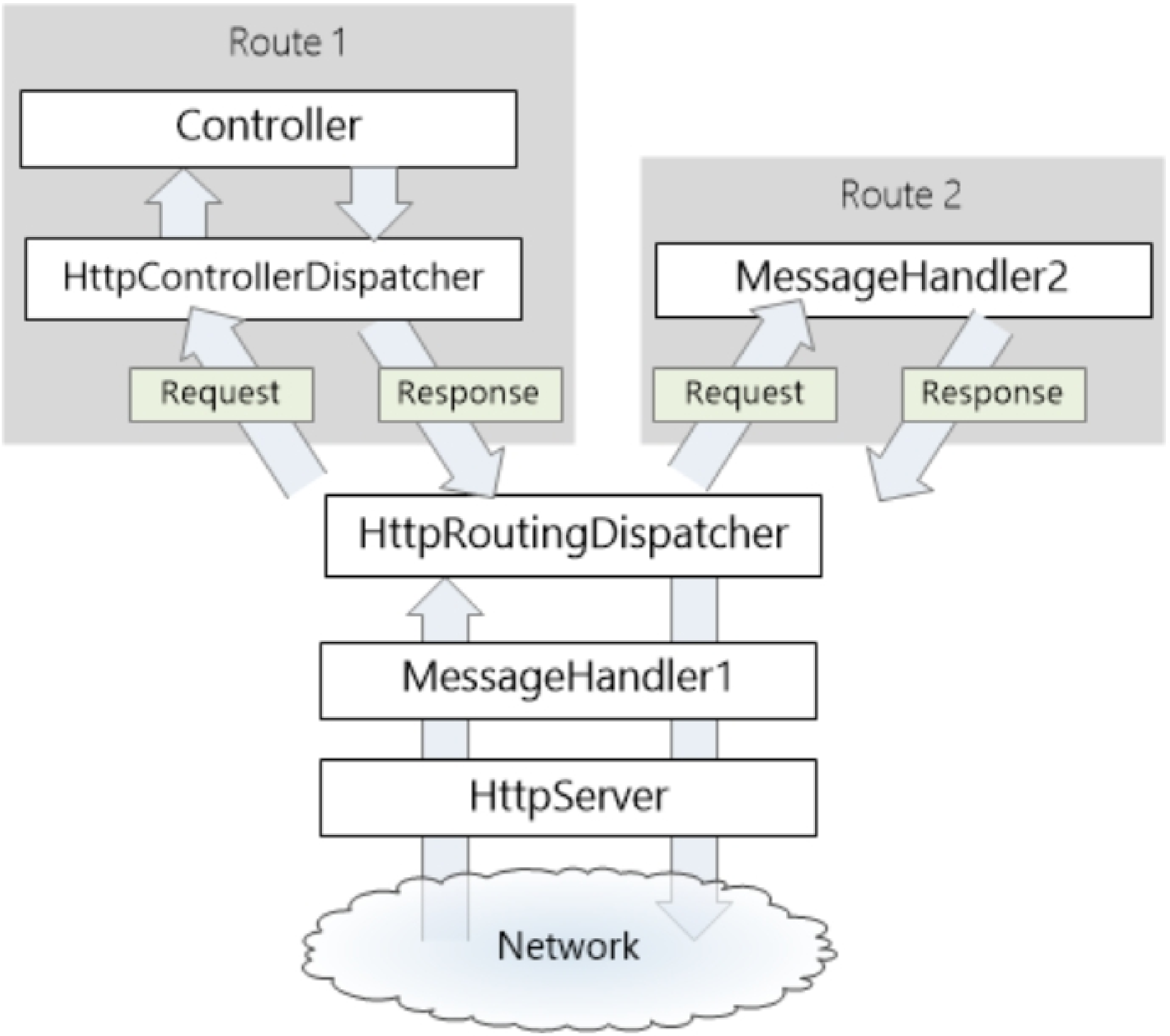
localhost:12345/api/products?key=12345

This XML file does not appear to have any style infor

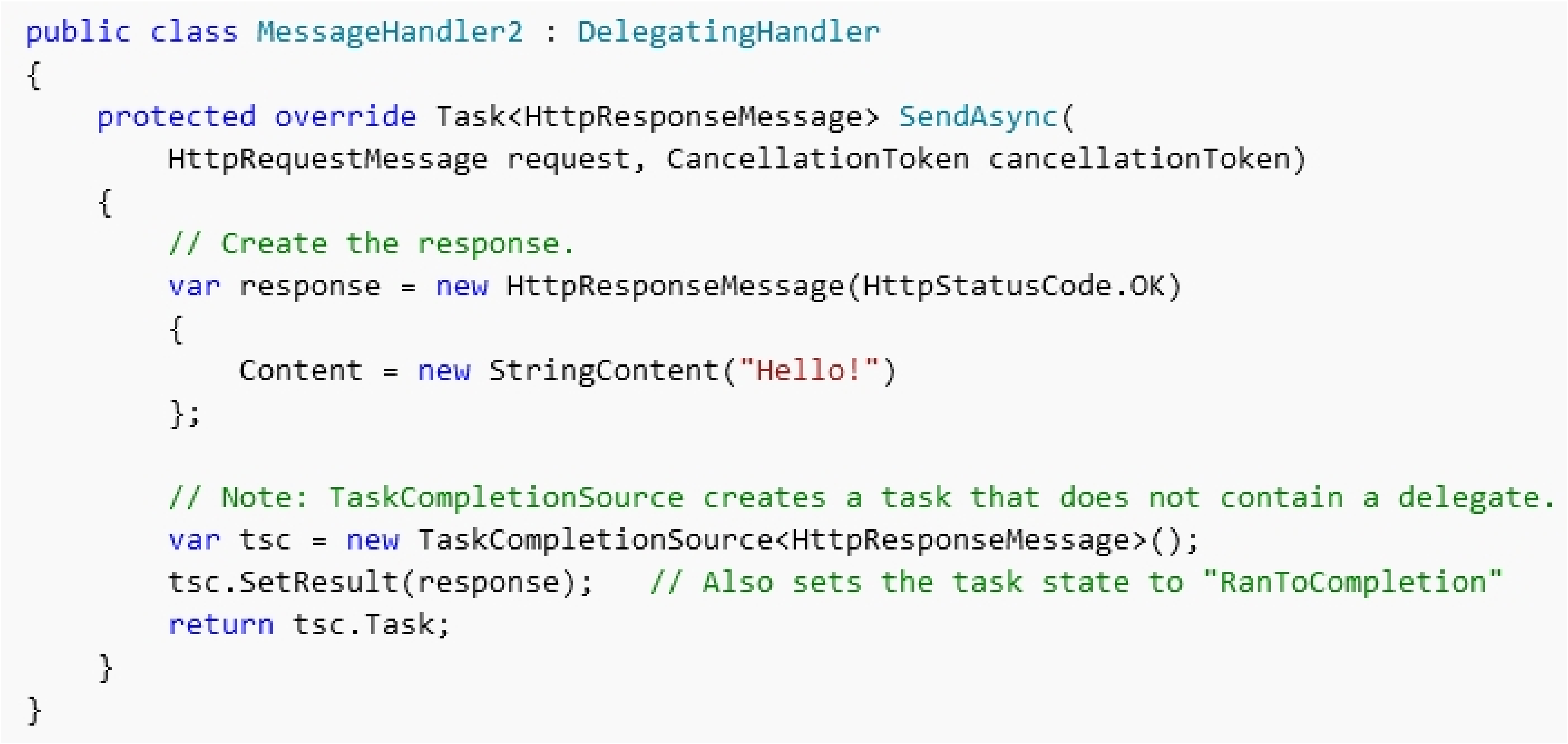
```
– <ArrayOfProduct>
  – <Product>
      <ProductId>1</ProductId>
      <ProductName>Monitor</ProductName>
      <ProductPrice>10000</ProductPrice>
    </Product>
  – <Product>
      <ProductId>2</ProductId>
      <ProductName>Keyboard</ProductName>
      <ProductPrice>1500</ProductPrice>
    </Product>
  – <Product>
      <ProductId>3</ProductId>
      <ProductName>Mouse</ProductName>
      <ProductPrice>400</ProductPrice>
    </Product>
  </ArrayOfProduct>
```



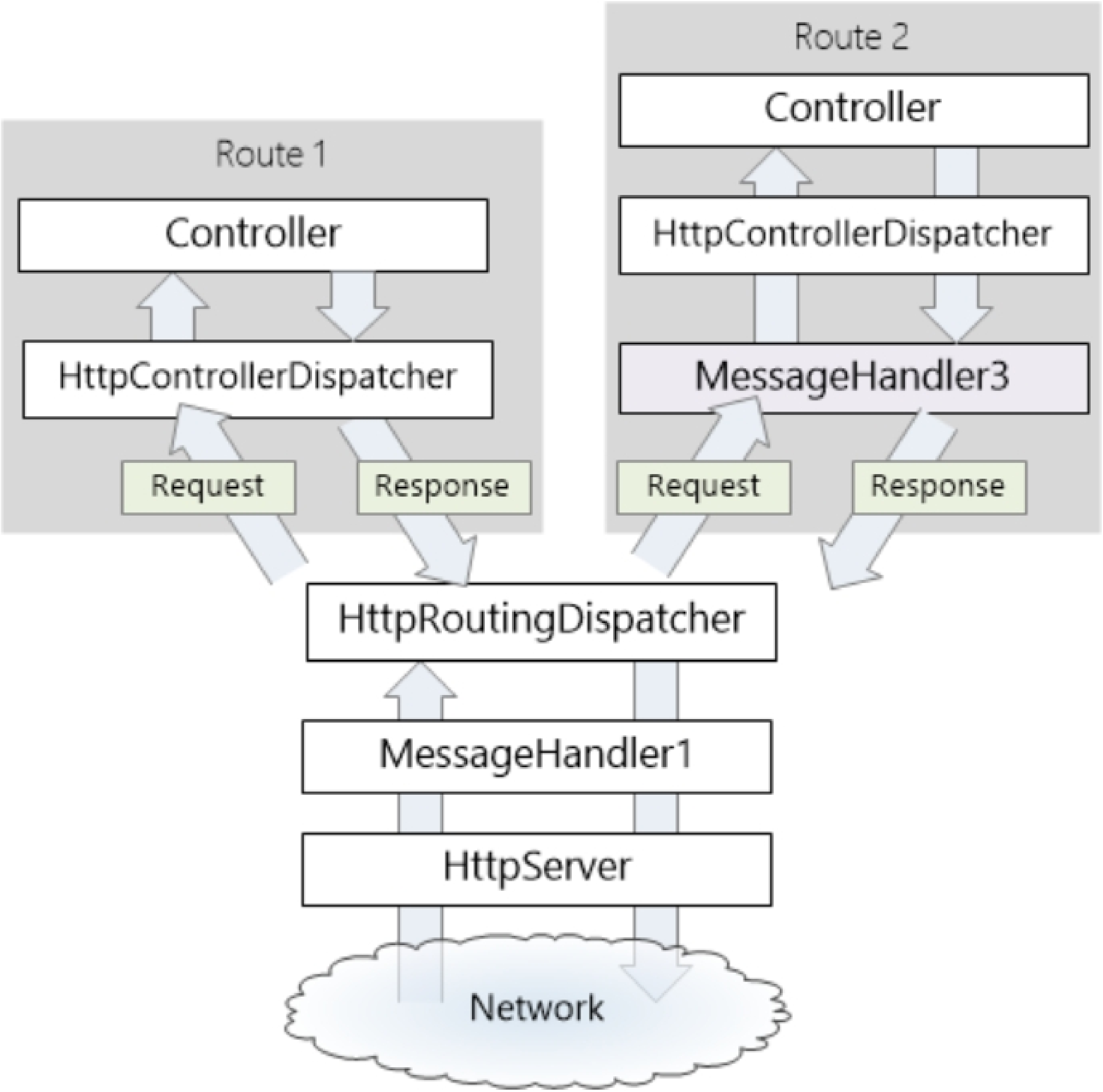
```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        config.Routes.MapHttpRoute(
             name: "Route1",
             routeTemplate: "api/{controller}/{id}",
             defaults: new { id = RouteParameter.Optional }
         );

        config.Routes.MapHttpRoute(
            name: "Route2",
            routeTemplate: "api2/{controller}/{id}",
            defaults: new { id = RouteParameter.Optional },
            constraints: null,
            handler: new MessageHandler2()  // per-route message handler
        );

        config.MessageHandlers.Add(new MessageHandler1());  // global message handler
    }
```

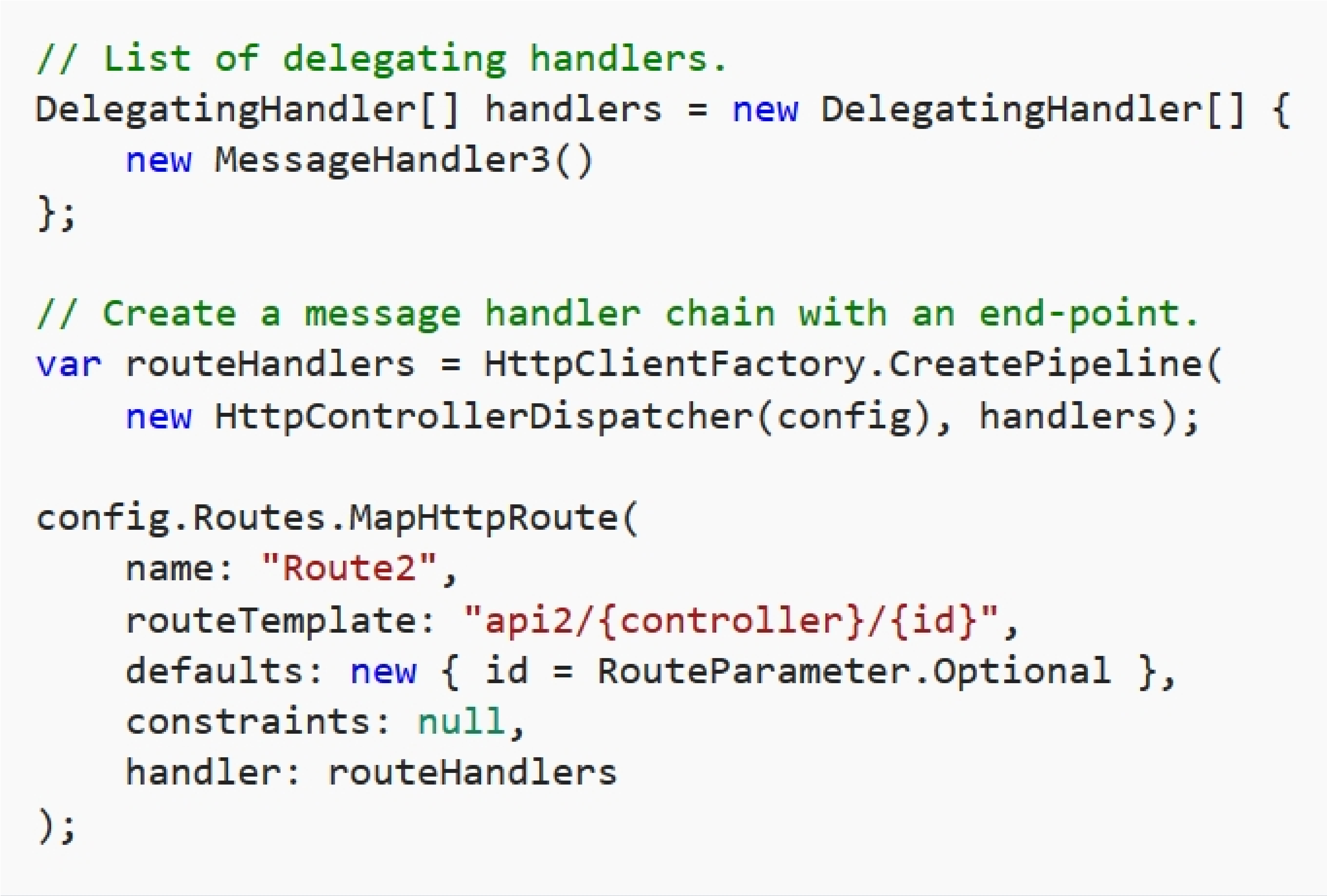


Notice that `MessageHandler2` replaces the default **HttpControllerDispatcher**. In this example, `MessageHandler2` creates the response, and requests that match "Route2" never go to a controller. This lets you replace the entire Web API controller mechanism with your own custom endpoint.



```
public class MessageHandler2 : DelegatingHandler
{
    protected override Task<HttpResponseMessage> SendAsync(
        HttpRequestMessage request, CancellationToken cancellationToken)
    {
        // Create the response.
        var response = new HttpResponseMessage(HttpStatusCode.OK)
        {
            Content = new StringContent("Hello!")
        };

        // Note: TaskCompletionSource creates a task that does not contain a delegate.
        var tsc = new TaskCompletionSource<HttpResponseMessage>();
        tsc.SetResult(response);   // Also sets the task state to "RanToCompletion"
        return tsc.Task;
    }
}
```


Route 1
Controller
HttpControllerDispatcher
Request
Response
Route 2
Controller
HttpControllerDispatcher
MessageHandler3
Request
Response
HttpRoutingDispatcher
MessageHandler1
HttpServer
Network



```
// List of delegating handlers.
DelegatingHandler[] handlers = new DelegatingHandler[] {
    new MessageHandler3()
};

// Create a message handler chain with an end-point.
var routeHandlers = HttpClientFactory.CreatePipeline(
    new HttpControllerDispatcher(config), handlers);

config.Routes.MapHttpRoute(
    name: "Route2",
    routeTemplate: "api2/{controller}/{id}",
    defaults: new { id = RouteParameter.Optional },
    constraints: null,
    handler: routeHandlers
);
```