**Web API 2**

Action Results in Web API 2

A Web API controller action can return any of the following:

1. void
2. **HttpResponseMessage**
3. **IHttpActionResult**
4. Some other type

## void

If the return type is void, Web API simply returns an empty HTTP response with status code 204 (No Content).1

Example controller:

Copy

C#

public class ValuesController : ApiController

{

public void Post()

{

}

}

HTTP response:

Copy

console

HTTP/1.1 204 No Content

Server: Microsoft-IIS/8.0

Date: Mon, 27 Jan 2014 02:13:26 GMT

## HttpResponseMessage

If the action returns an [HttpResponseMessage](https://msdn.microsoft.com/en-us/library/system.net.http.httpresponsemessage.aspx), Web API converts the return value directly into an HTTP response message, using the properties of the **HttpResponseMessage** object to populate the response.

This option gives you a lot of control over the response message. For example, the following controller action sets the Cache-Control header.

Copy

C#

public class ValuesController : ApiController

{

public HttpResponseMessage Get()

{

HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.OK, "value");

response.Content = new StringContent("hello", Encoding.Unicode);

response.Headers.CacheControl = new CacheControlHeaderValue()

{

MaxAge = TimeSpan.FromMinutes(20)

};

return response;

}

}

Response:

Copy

console

HTTP/1.1 200 OK

Cache-Control: max-age=1200

Content-Length: 10

Content-Type: text/plain; charset=utf-16

Server: Microsoft-IIS/8.0

Date: Mon, 27 Jan 2014 08:53:35 GMT

hello

If you pass a domain model to the **CreateResponse** method, Web API uses a [media formatter](https://docs.microsoft.com/en-us/aspnet/web-api/overview/formats-and-model-binding/media-formatters) to write the serialized model into the response body.

Copy

C#

public HttpResponseMessage Get()

{

// Get a list of products from a database.

IEnumerable<Product> products = GetProductsFromDB();

// Write the list to the response body.

HttpResponseMessage response = Request.CreateResponse(HttpStatusCode.OK, products);

return response;

}

## IHttpActionResult

The **IHttpActionResult** interface was introducted in Web API 2.

public IHttpActionResult Get (int id)

{

Product product = \_repository.Get (id);

if (product == null)

{

return NotFound(); // Returns a NotFoundResult

}

return Ok(product); // Returns an OkNegotiatedContentResult

}

#region Assembly System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089

// C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\.NETFramework\v4.5.2\System.dll

#endregion

namespace System.Net

{

//

// Summary:

// Contains the values of status codes defined for HTTP.

public enum HttpStatusCode

{

//

// Summary:

// Equivalent to HTTP status 100. System.Net.HttpStatusCode.Continue indicates that

// the client can continue with its request.

Continue = 100,

//

// Summary:

// Equivalent to HTTP status 101. System.Net.HttpStatusCode.SwitchingProtocols indicates

// that the protocol version or protocol is being changed.

SwitchingProtocols = 101,

//

// Summary:

// Equivalent to HTTP status 200. System.Net.HttpStatusCode.OK indicates that the

// request succeeded and that the requested information is in the response. This

// is the most common status code to receive.

OK = 200,

//

// Summary:

// Equivalent to HTTP status 201. System.Net.HttpStatusCode.Created indicates that

// the request resulted in a new resource created before the response was sent.

Created = 201,

//

// Summary:

// Equivalent to HTTP status 202. System.Net.HttpStatusCode.Accepted indicates that

// the request has been accepted for further processing.

Accepted = 202,

//

// Summary:

// Equivalent to HTTP status 203. System.Net.HttpStatusCode.NonAuthoritativeInformation

// indicates that the returned metainformation is from a cached copy instead of

// the origin server and therefore may be incorrect.

NonAuthoritativeInformation = 203,

//

// Summary:

// Equivalent to HTTP status 204. System.Net.HttpStatusCode.NoContent indicates

// that the request has been successfully processed and that the response is intentionally

// blank.

NoContent = 204,

//

// Summary:

// Equivalent to HTTP status 205. System.Net.HttpStatusCode.ResetContent indicates

// that the client should reset (not reload) the current resource.

ResetContent = 205,

//

// Summary:

// Equivalent to HTTP status 206. System.Net.HttpStatusCode.PartialContent indicates

// that the response is a partial response as requested by a GET request that includes

// a byte range.

PartialContent = 206,

//

// Summary:

// Equivalent to HTTP status 300. System.Net.HttpStatusCode.MultipleChoices indicates

// that the requested information has multiple representations. The default action

// is to treat this status as a redirect and follow the contents of the Location

// header associated with this response.

MultipleChoices = 300,

//

// Summary:

// Equivalent to HTTP status 300. System.Net.HttpStatusCode.Ambiguous indicates

// that the requested information has multiple representations. The default action

// is to treat this status as a redirect and follow the contents of the Location

// header associated with this response.

Ambiguous = 300,

//

// Summary:

// Equivalent to HTTP status 301. System.Net.HttpStatusCode.MovedPermanently indicates

// that the requested information has been moved to the URI specified in the Location

// header. The default action when this status is received is to follow the Location

// header associated with the response.

MovedPermanently = 301,

//

// Summary:

// Equivalent to HTTP status 301. System.Net.HttpStatusCode.Moved indicates that

// the requested information has been moved to the URI specified in the Location

// header. The default action when this status is received is to follow the Location

// header associated with the response. When the original request method was POST,

// the redirected request will use the GET method.

Moved = 301,

//

// Summary:

// Equivalent to HTTP status 302. System.Net.HttpStatusCode.Found indicates that

// the requested information is located at the URI specified in the Location header.

// The default action when this status is received is to follow the Location header

// associated with the response. When the original request method was POST, the

// redirected request will use the GET method.

Found = 302,

//

// Summary:

// Equivalent to HTTP status 302. System.Net.HttpStatusCode.Redirect indicates that

// the requested information is located at the URI specified in the Location header.

// The default action when this status is received is to follow the Location header

// associated with the response. When the original request method was POST, the

// redirected request will use the GET method.

Redirect = 302,

//

// Summary:

// Equivalent to HTTP status 303. System.Net.HttpStatusCode.SeeOther automatically

// redirects the client to the URI specified in the Location header as the result

// of a POST. The request to the resource specified by the Location header will

// be made with a GET.

SeeOther = 303,

//

// Summary:

// Equivalent to HTTP status 303. System.Net.HttpStatusCode.RedirectMethod automatically

// redirects the client to the URI specified in the Location header as the result

// of a POST. The request to the resource specified by the Location header will

// be made with a GET.

RedirectMethod = 303,

//

// Summary:

// Equivalent to HTTP status 304. System.Net.HttpStatusCode.NotModified indicates

// that the client's cached copy is up to date. The contents of the resource are

// not transferred.

NotModified = 304,

//

// Summary:

// Equivalent to HTTP status 305. System.Net.HttpStatusCode.UseProxy indicates that

// the request should use the proxy server at the URI specified in the Location

// header.

UseProxy = 305,

//

// Summary:

// Equivalent to HTTP status 306. System.Net.HttpStatusCode.Unused is a proposed

// extension to the HTTP/1.1 specification that is not fully specified.

Unused = 306,

//

// Summary:

// Equivalent to HTTP status 307. System.Net.HttpStatusCode.TemporaryRedirect indicates

// that the request information is located at the URI specified in the Location

// header. The default action when this status is received is to follow the Location

// header associated with the response. When the original request method was POST,

// the redirected request will also use the POST method.

TemporaryRedirect = 307,

//

// Summary:

// Equivalent to HTTP status 307. System.Net.HttpStatusCode.RedirectKeepVerb indicates

// that the request information is located at the URI specified in the Location

// header. The default action when this status is received is to follow the Location

// header associated with the response. When the original request method was POST,

// the redirected request will also use the POST method.

RedirectKeepVerb = 307,

//

// Summary:

// Equivalent to HTTP status 400. System.Net.HttpStatusCode.BadRequest indicates

// that the request could not be understood by the server. System.Net.HttpStatusCode.BadRequest

// is sent when no other error is applicable, or if the exact error is unknown or

// does not have its own error code.

BadRequest = 400,

//

// Summary:

// Equivalent to HTTP status 401. System.Net.HttpStatusCode.Unauthorized indicates

// that the requested resource requires authentication. The WWW-Authenticate header

// contains the details of how to perform the authentication.

Unauthorized = 401,

//

// Summary:

// Equivalent to HTTP status 402. System.Net.HttpStatusCode.PaymentRequired is reserved

// for future use.

PaymentRequired = 402,

//

// Summary:

// Equivalent to HTTP status 403. System.Net.HttpStatusCode.Forbidden indicates

// that the server refuses to fulfill the request.

Forbidden = 403,

//

// Summary:

// Equivalent to HTTP status 404. System.Net.HttpStatusCode.NotFound indicates that

// the requested resource does not exist on the server.

NotFound = 404,

//

// Summary:

// Equivalent to HTTP status 405. System.Net.HttpStatusCode.MethodNotAllowed indicates

// that the request method (POST or GET) is not allowed on the requested resource.

MethodNotAllowed = 405,

//

// Summary:

// Equivalent to HTTP status 406. System.Net.HttpStatusCode.NotAcceptable indicates

// that the client has indicated with Accept headers that it will not accept any

// of the available representations of the resource.

NotAcceptable = 406,

//

// Summary:

// Equivalent to HTTP status 407. System.Net.HttpStatusCode.ProxyAuthenticationRequired

// indicates that the requested proxy requires authentication. The Proxy-authenticate

// header contains the details of how to perform the authentication.

ProxyAuthenticationRequired = 407,

//

// Summary:

// Equivalent to HTTP status 408. System.Net.HttpStatusCode.RequestTimeout indicates

// that the client did not send a request within the time the server was expecting

// the request.

RequestTimeout = 408,

//

// Summary:

// Equivalent to HTTP status 409. System.Net.HttpStatusCode.Conflict indicates that

// the request could not be carried out because of a conflict on the server.

Conflict = 409,

//

// Summary:

// Equivalent to HTTP status 410. System.Net.HttpStatusCode.Gone indicates that

// the requested resource is no longer available.

Gone = 410,

//

// Summary:

// Equivalent to HTTP status 411. System.Net.HttpStatusCode.LengthRequired indicates

// that the required Content-length header is missing.

LengthRequired = 411,

//

// Summary:

// Equivalent to HTTP status 412. System.Net.HttpStatusCode.PreconditionFailed indicates

// that a condition set for this request failed, and the request cannot be carried

// out. Conditions are set with conditional request headers like If-Match, If-None-Match,

// or If-Unmodified-Since.

PreconditionFailed = 412,

//

// Summary:

// Equivalent to HTTP status 413. System.Net.HttpStatusCode.RequestEntityTooLarge

// indicates that the request is too large for the server to process.

RequestEntityTooLarge = 413,

//

// Summary:

// Equivalent to HTTP status 414. System.Net.HttpStatusCode.RequestUriTooLong indicates

// that the URI is too long.

RequestUriTooLong = 414,

//

// Summary:

// Equivalent to HTTP status 415. System.Net.HttpStatusCode.UnsupportedMediaType

// indicates that the request is an unsupported type.

UnsupportedMediaType = 415,

//

// Summary:

// Equivalent to HTTP status 416. System.Net.HttpStatusCode.RequestedRangeNotSatisfiable

// indicates that the range of data requested from the resource cannot be returned,

// either because the beginning of the range is before the beginning of the resource,

// or the end of the range is after the end of the resource.

RequestedRangeNotSatisfiable = 416,

//

// Summary:

// Equivalent to HTTP status 417. System.Net.HttpStatusCode.ExpectationFailed indicates

// that an expectation given in an Expect header could not be met by the server.

ExpectationFailed = 417,

//

// Summary:

// Equivalent to HTTP status 426. System.Net.HttpStatusCode.UpgradeRequired indicates

// that the client should switch to a different protocol such as TLS/1.0.

UpgradeRequired = 426,

//

// Summary:

// Equivalent to HTTP status 500. System.Net.HttpStatusCode.InternalServerError

// indicates that a generic error has occurred on the server.

InternalServerError = 500,

//

// Summary:

// Equivalent to HTTP status 501. System.Net.HttpStatusCode.NotImplemented indicates

// that the server does not support the requested function.

NotImplemented = 501,

//

// Summary:

// Equivalent to HTTP status 502. System.Net.HttpStatusCode.BadGateway indicates

// that an intermediate proxy server received a bad response from another proxy

// or the origin server.

BadGateway = 502,

//

// Summary:

// Equivalent to HTTP status 503. System.Net.HttpStatusCode.ServiceUnavailable indicates

// that the server is temporarily unavailable, usually due to high load or maintenance.

ServiceUnavailable = 503,

//

// Summary:

// Equivalent to HTTP status 504. System.Net.HttpStatusCode.GatewayTimeout indicates

// that an intermediate proxy server timed out while waiting for a response from

// another proxy or the origin server.

GatewayTimeout = 504,

//

// Summary:

// Equivalent to HTTP status 505. System.Net.HttpStatusCode.HttpVersionNotSupported

// indicates that the requested HTTP version is not supported by the server.

HttpVersionNotSupported = 505

}

}

public class Book

{

public int BookId { get; set; }

[Required]

public string Title { get; set; }

public decimal Price { get; set; }

public string Genre { get; set; }

public DateTime PublishDate { get; set; }

public string Description { get; set; }

public int AuthorId { get; set; }

[ForeignKey("AuthorId")]

public Author Author { get; set; }

}

public class Author

{

public int AuthorId { get; set; }

[Required]

public string Name { get; set; }

}