« Previous Next »

Action Method Return Type:

In the previous section, you learned about parameter binding with Web API action method. Here, you will learn about the return types of action methods which in turn will be embedded in the Web API response sent to the client.

The Web API action method can have following return types.

1. Void
2. Primitive type or Complex type
3. HttpResponseMessage
4. IHttpActionResult

Void:

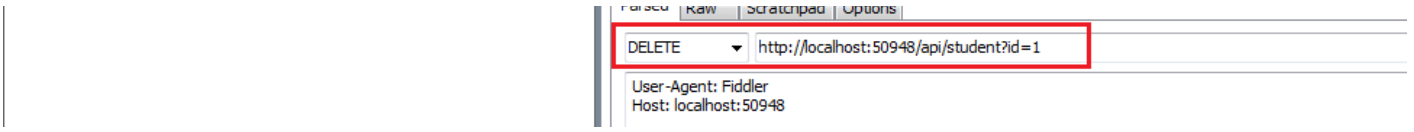
It's not necessary that all action methods must return something. It can have void return type.

For example, consider the following Delete action method that just deletes the student from the data source and returns nothing.

Example: Void Return Type

```
public class StudentController : ApiController
{
    public void Delete(int id)
    {
        DeleteStudentFromDB(id);
    }
}
```

As you can see above Delete action method returns void. It will send 204 "No Content" status code as a response when you send HTTP DELETE request as shown below.



Void Response Status

Primitive or Complex Type:

An action method can return primitive or other custom complex types as other normal methods.

Consider the following Get action methods.

Example: Primitive or Complex Return Type

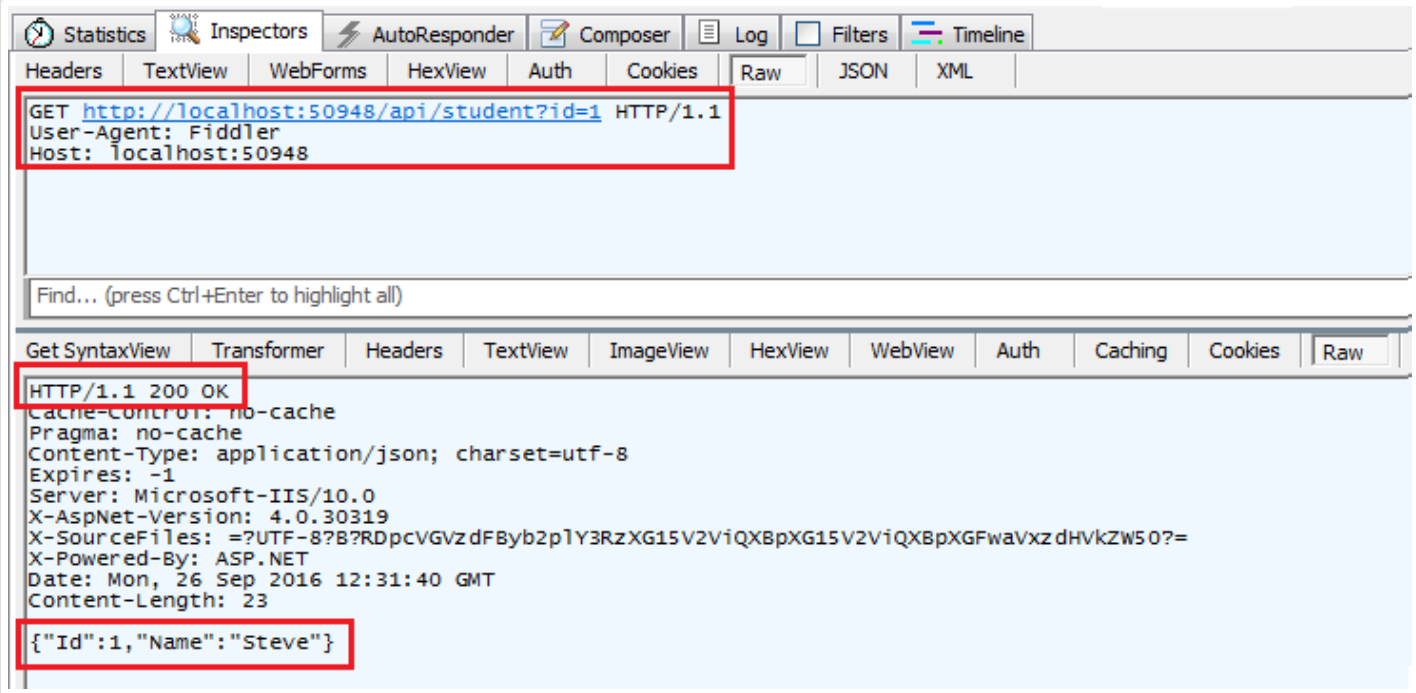
```
public class Student
{
    public int Id { get; set; }
    public string Name { get; set; }
}

public class StudentController : ApiController
{
    public int GetId(string name)
    {
        int id = GetStudentId(name);

        return id;
    }

    public Student GetStudent(int id)
    {
        var student = GetStudentFromDB(id);

        return student;
    }
}
```

As you can see above, GetId action method returns an integer and GetStudent action method returns a Student type.

An HTTP GET request `http://localhost:xxxx/api/student?name=john` will return following response in Fiddler.

Primitive Return Type in Response

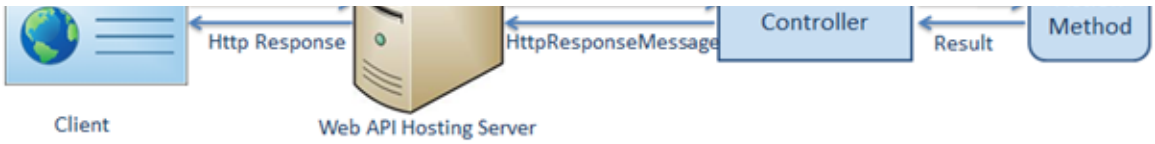
An HTTP GET request `http://localhost:xxxx/api/student?id=1` will return following response in Fiddler.



Complex Return Type in Response

HttpResponseMessage:

Web API controller always returns an object of HttpResponseMessage to the hosting infrastructure. The following figure illustrates the overall Web API request/response pipeline.



Web API Request Pipeline

Visit Web API HTTP Message Life Cycle Poster for more details.

As you can see in the above figure, the Web API controller returns HttpResponseMessage object. You can also create and return an object of HttpResponseMessage directly from an action method.

The advantage of sending HttpResponseMessage from an action method is that you can configure a response your way. You can set the status code, content or error message (if any) as per your requirement.

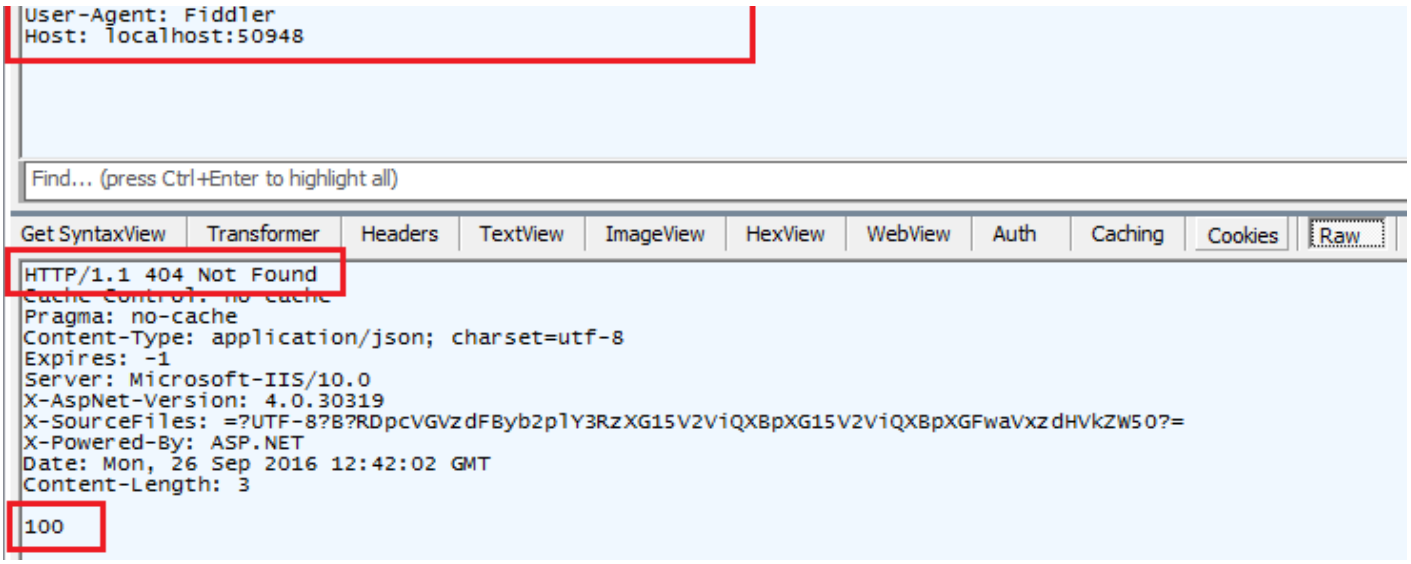
Consider the following example.

Example: Return HttpResponseMessage

```
public HttpResponseMessage Get(int id)
{
    Student stud = GetStudentFromDB(id);

    if (stud == null) {
        return Request.CreateResponse(HttpStatusCode.NotFound, id);
    }

    return Request.CreateResponse(HttpStatusCode.OK, stud);
}
```

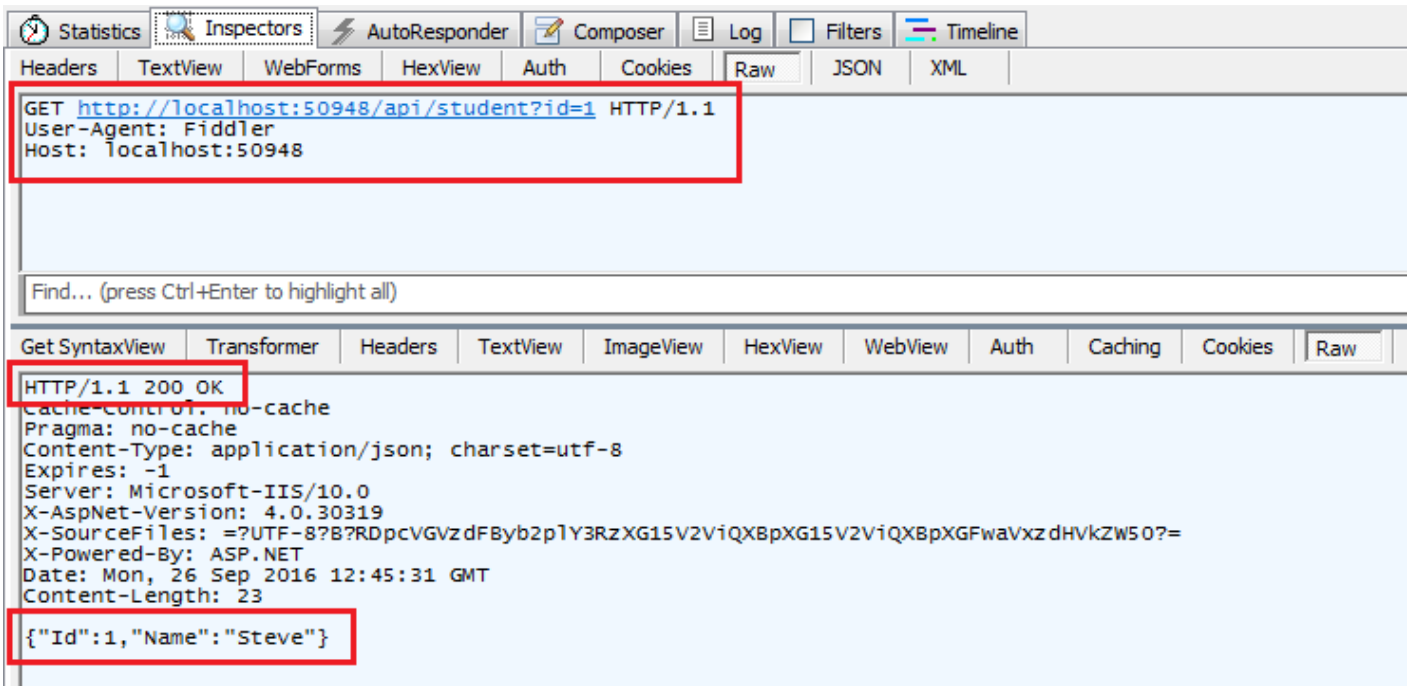
In the above action method, if there is no student with specified id in the DB then it will return HTTP 404 Not Found status code, otherwise it will return 200 OK status with student data.

For example, an http GET request `http://localhost:xxxx/api/student?id=100` will get following response considering student with id=100 does not exists in the DB.



Web API Response in Fiddler

The same way, an HTTP GET request `http://localhost:60464/api/student?id=1` will get following response considering student with id=1 exists in the database .



Web API Response in Fiddler

IHttpActionResult:

The *IHttpActionResult* was introduced in Web API 2 (.NET 4.5). An action method in Web API 2 can return an implementation of IHttpActionResult class which is more or less similar to ActionResult class in ASP.NET MVC.

Example: Return IHttpActionResult Type using Ok() and NotFound() Methods

```
public IHttpActionResult Get(int id)
{
    Student stud = GetStudentFromDB(id);

    if (stud == null)
    {
        return NotFound();
    }

    return Ok(stud);
}
```

In the above example, if student with specified id does not exists in the database then it will return response with the status code 404 otherwise it sends student data with status code 200 as a response. As you can see, we don't have to write much code because NotFound() and Ok() method does it all for us.

The following table lists all the methods of ApiController class that returns an object of a class that implements IHttpActionResult interface.

ApiController Method	Description
BadRequest()	Creates a BadRequestResult object with status code 400.
Conflict()	Creates a ConflictResult object with status code 409.
Content()	Creates a NegotiatedContentResult with the specified status code and data.
Created()	Creates a CreatedNegotiatedContentResult with status code 201 Created.
CreatedAtRoute()	Creates a CreatedAtRouteNegotiatedContentResult with status code 201 created.
InternalServerError()	Creates an InternalServerErrorResult with status code 500 Internal server error.
NotFound()	Creates a NotFoundResult with status code404.
Ok()	Creates an OkResult with status code 200.
Redirect()	Creates a RedirectResult with status code 302.
RedirectToRoute()	Creates a RedirectToRouteResult with status code 302.
ResponseMessage()	Creates a ResponseMessageResult with the specified HttpResponseMessage.
StatusCode()	Creates a StatusCodeResult with the specified http status code.

Visit MSDN to know all the members of ApiController.

Create Custom Result Type:

You can create your own custom class as a result type that implements IHttpActionResult interface.

The following example demonstrates implementing IHttpActionResult class.

Example: Create Custom Result Type

```
public class TextResult : IHttpActionResult
{
    string _value;
    HttpRequestMessage _request;

    public TextResult(string value, HttpRequestMessage request)
    {
        _value = value;
        _request = request;
    }

    public Task<HttpResponseMessage> ExecuteAsync(CancellationToken cancellationToken)
    {
        var response = new HttpResponseMessage()
        {
            Content = new StringContent(_value),
            RequestMessage = _request
        };
        return Task.FromResult(response);
    }
}
```

Now, you can return TextResult object from the action method as shown below.

Example: Return Custom Result Type

```
public IHttpActionResult GetName(int id)
{
    string name = GetStudentName(id);
```

```
        return NotFound();
    }

    return new TextResult(name, Request);
}
```


« Previous

Next »

TUTORIALSTEACHER.COM

TutorialsTeacher.com is optimized for learning web technologies step by step. Examples might be simplified to improve reading and basic understanding. While using this site, you agree to have read and accepted our terms of use and privacy policy.

feedback@tutorialsteacher.com

TUTORIALS

SHARE

E-MAIL LIST

Subscribe to TutorialsTeacher email list and get latest updates, tips & tricks on C#, .Net, JavaScript, jQuery, AngularJS, Node.js to your inbox.

Email address GO

We respect your privacy.