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);
   }
}
```

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;
    }
}
```

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.

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.

You can create your own class that implements IHttpActionResult or use various methods of ApiController class that returns an object that implement the IHttpActionResult.

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.
Unauthorized()	Creates an UnauthorizedResult with status code 401.

If you need to make the controller async then you have to wrap the result in a task and use await, in the same way that you make any other method asynchronous. Remember that if you are doing async, you should use it all the way down and not synchronously block on an async method. We have async controllers for this purpose:

```
public async Task<IHttpActionResult> GetAsync(int id)

{

    var product = await dbContext.Products.GetAsync(id);



    if (product == null)

       return NotFound();



   return Ok(product);
```

```
}
```