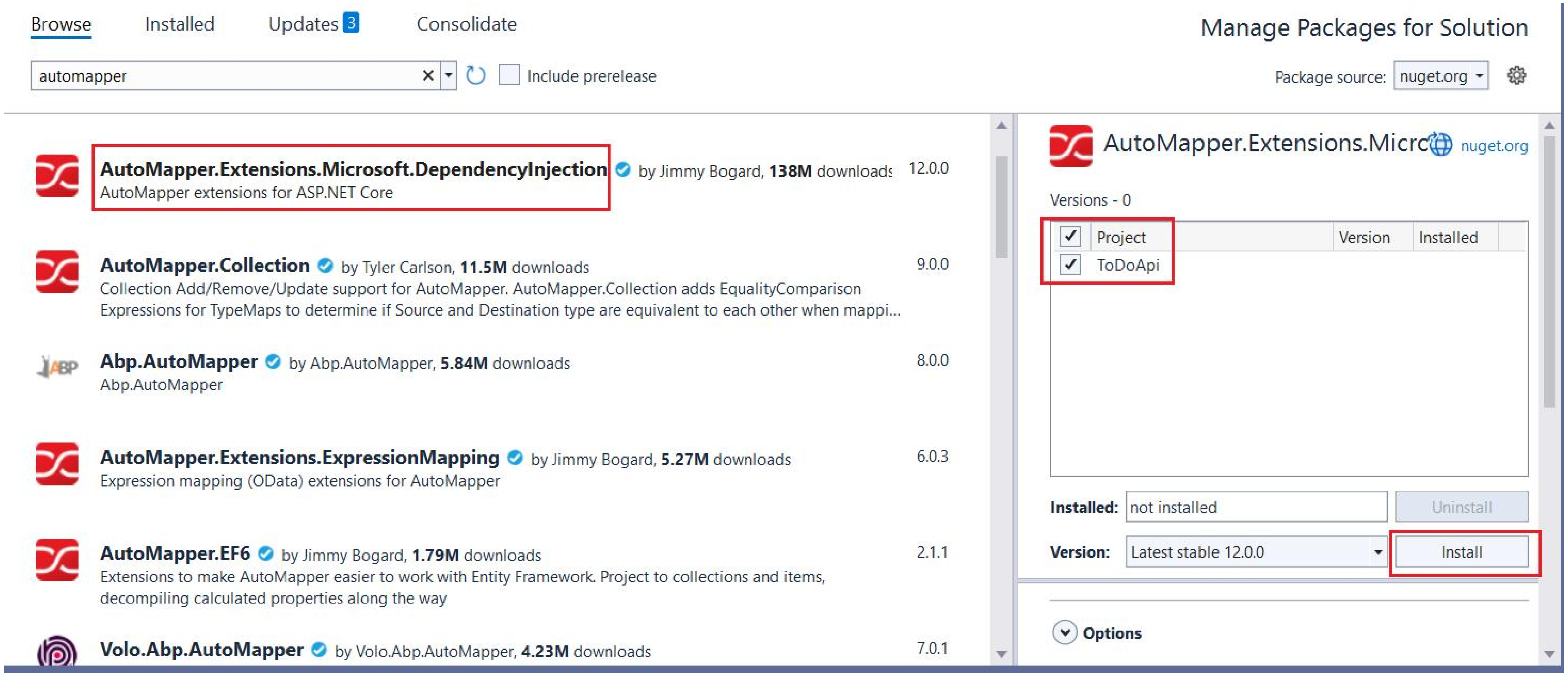
AutoMapper

- AutoMapper is a library to map one object to another.
- We can convert input object to an output object and vice versa.
- Useful in loosely coupled design having multiple layered architecture.
- Provides cleaner code and requires less time and code to map objects.
- Shorter development time.
- Offers greater maintainability by centralizing the mapping between the two different objects in a unique place.
- Unit testing is easier.

AutoMapper

It has

- Profile Classes for mappings.
- Naming Conventions, configuring the source and destination naming conventions.
- Reverse Mapping, mapping two objects both ways.
- Nested Mapping, mapping child objects.
- General Configuration, setting your preferences once in project startup.



Browse   Installed   Updates 3   Consolidate

Manage Packages for Solution

automapper   Include prerelease

Package source: nuget.org

AutoMapper.Extensions.Microsoft.DependencyInjection by Jimmy Bogard, **138M** downloads   12.0.0
AutoMapper extensions for ASP.NET Core

AutoMapper.Collection by Tyler Carlson, **11.5M** downloads   9.0.0
Collection Add/Remove/Update support for AutoMapper. AutoMapper.Collection adds EqualityComparison Expressions for TypeMaps to determine if Source and Destination type are equivalent to each other when mappi...

Abp.AutoMapper by Abp.AutoMapper, **5.84M** downloads   8.0.0
Abp.AutoMapper

AutoMapper.Extensions.ExpressionMapping by Jimmy Bogard, **5.27M** downloads   6.0.3
Expression mapping (OData) extensions for AutoMapper

AutoMapper.EF6 by Jimmy Bogard, **1.79M** downloads   2.1.1
Extensions to make AutoMapper easier to work with Entity Framework. Project to collections and items, decompiling calculated properties along the way

Volo.Abp.AutoMapper by Volo.Abp.AutoMapper, **4.23M** downloads   7.0.1

AutoMapper.Extensions.Micro   nuget.org

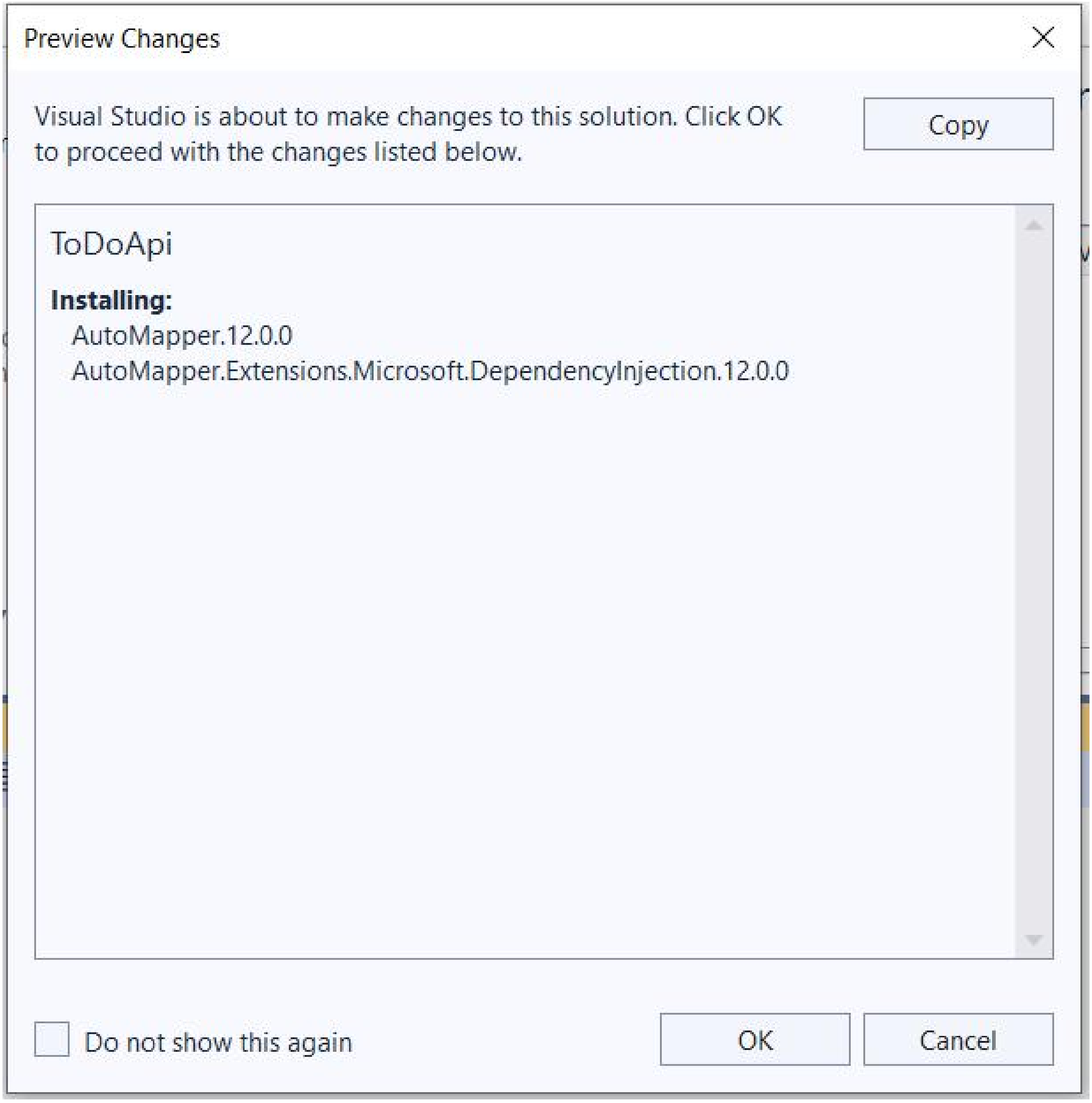
Versions - 0

✓ Project	Version	Installed
✓ ToDoApi		

Installed: not installed   Uninstall

Version: Latest stable 12.0.0   Install

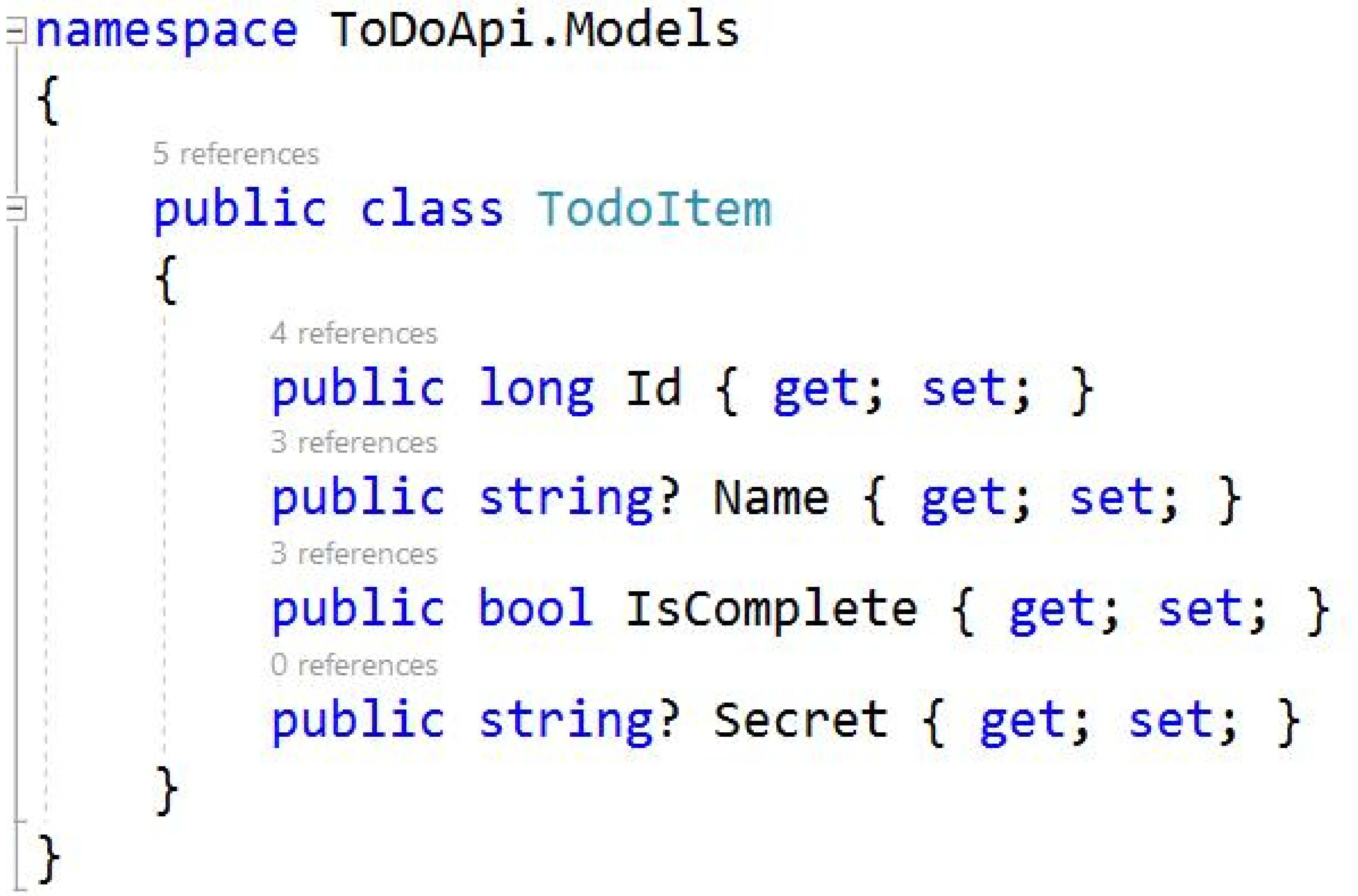
Options


Preview Changes

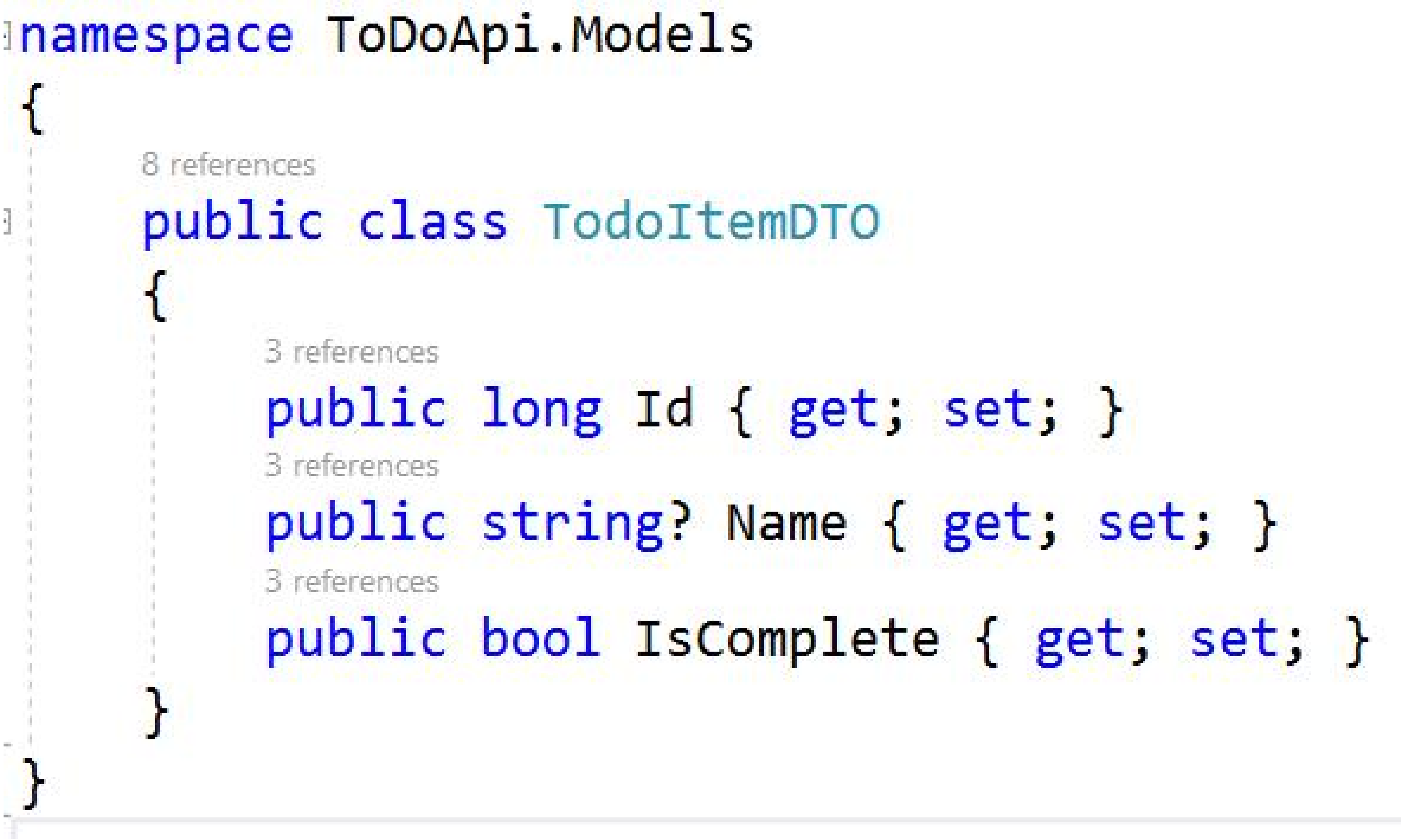
Visual Studio is about to make changes to this solution. Click OK to proceed with the changes listed below.

ToDoApi

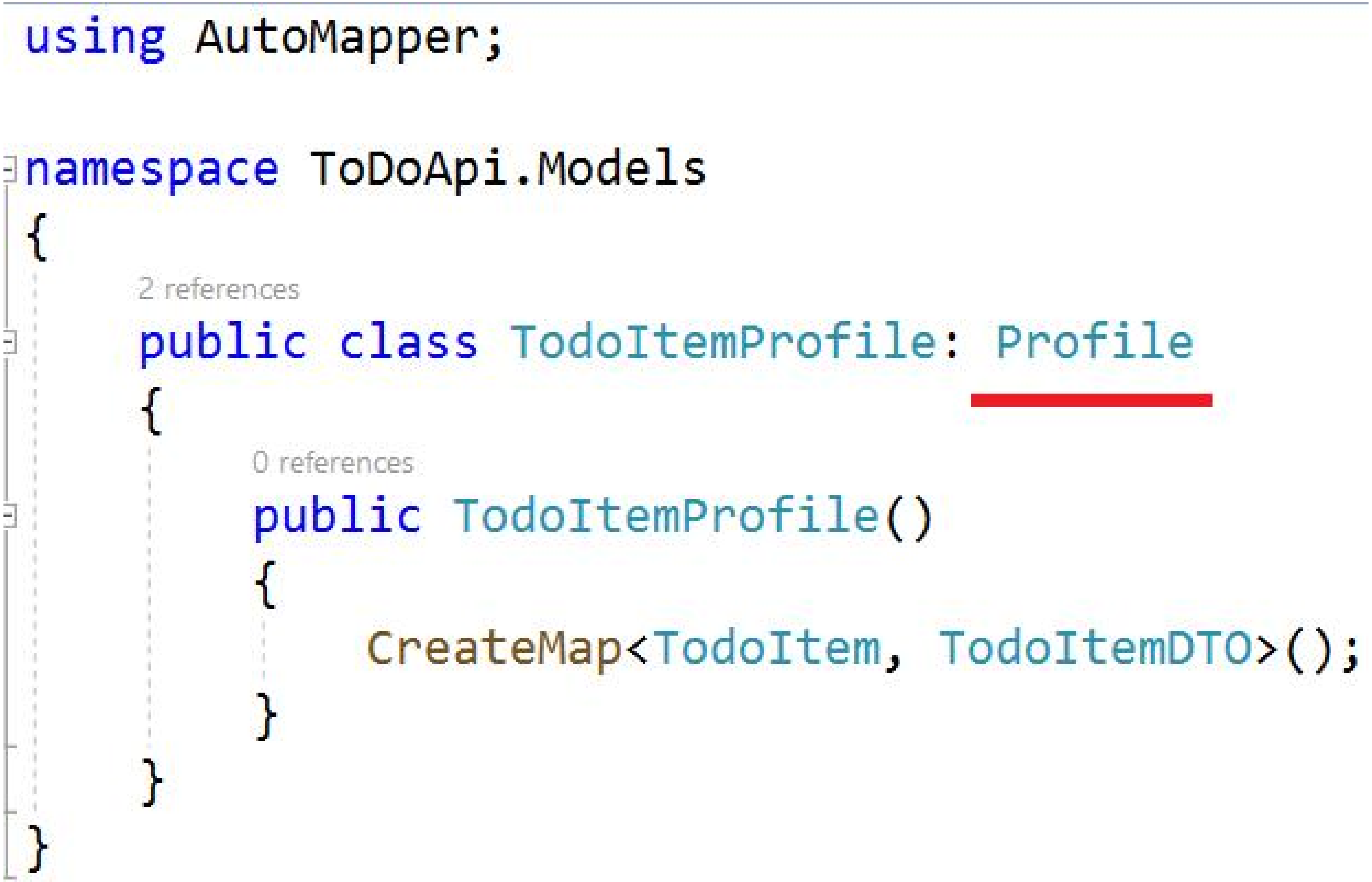
Installing:
AutoMapper.12.0.0
AutoMapper.Extensions.Microsoft.DependencyInjection.12.0.0



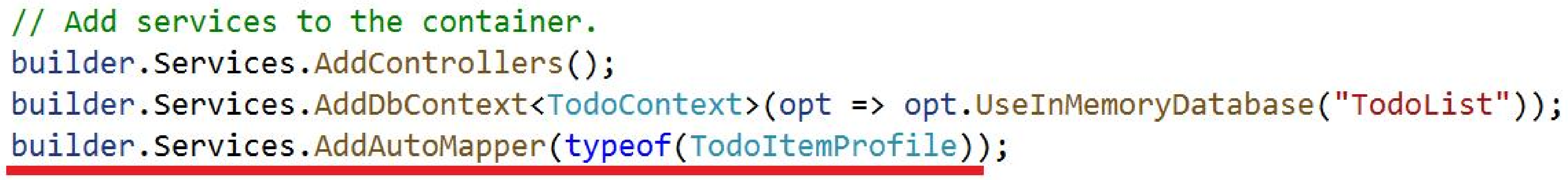
```
namespace ToDoApi.Models
{
        5 references
        public class TodoItem
        {
                4 references
                public long Id { get; set; }
                3 references
                public string? Name { get; set; }
                3 references
                public bool IsComplete { get; set; }
                0 references
                public string? Secret { get; set; }
        }
}
```



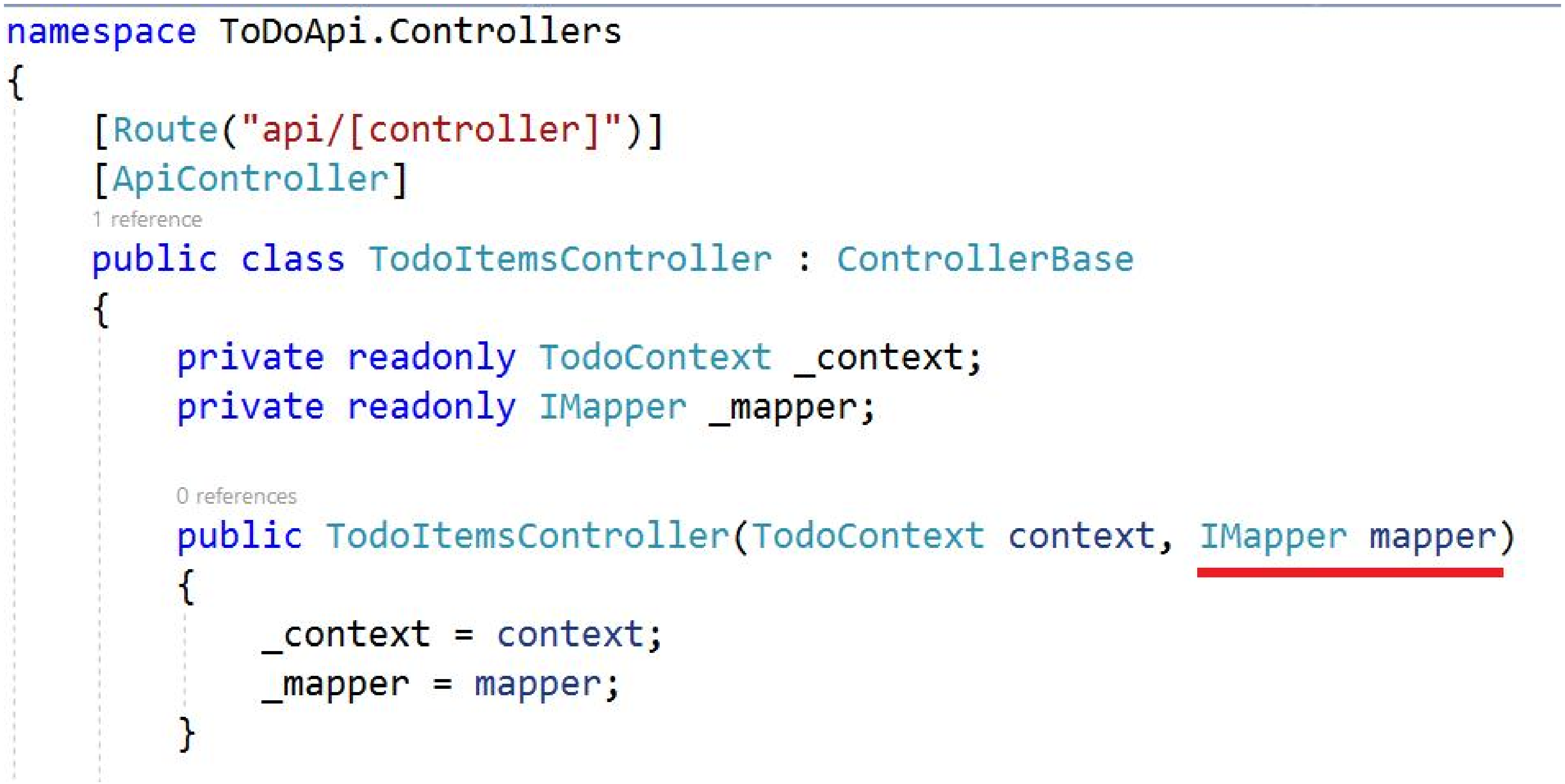
```
namespace ToDoApi.Models
{
    8 references
    public class TodoItemDTO
    {
        3 references
        public long Id { get; set; }
        3 references
        public string? Name { get; set; }
        3 references
        public bool IsComplete { get; set; }
    }
}
```



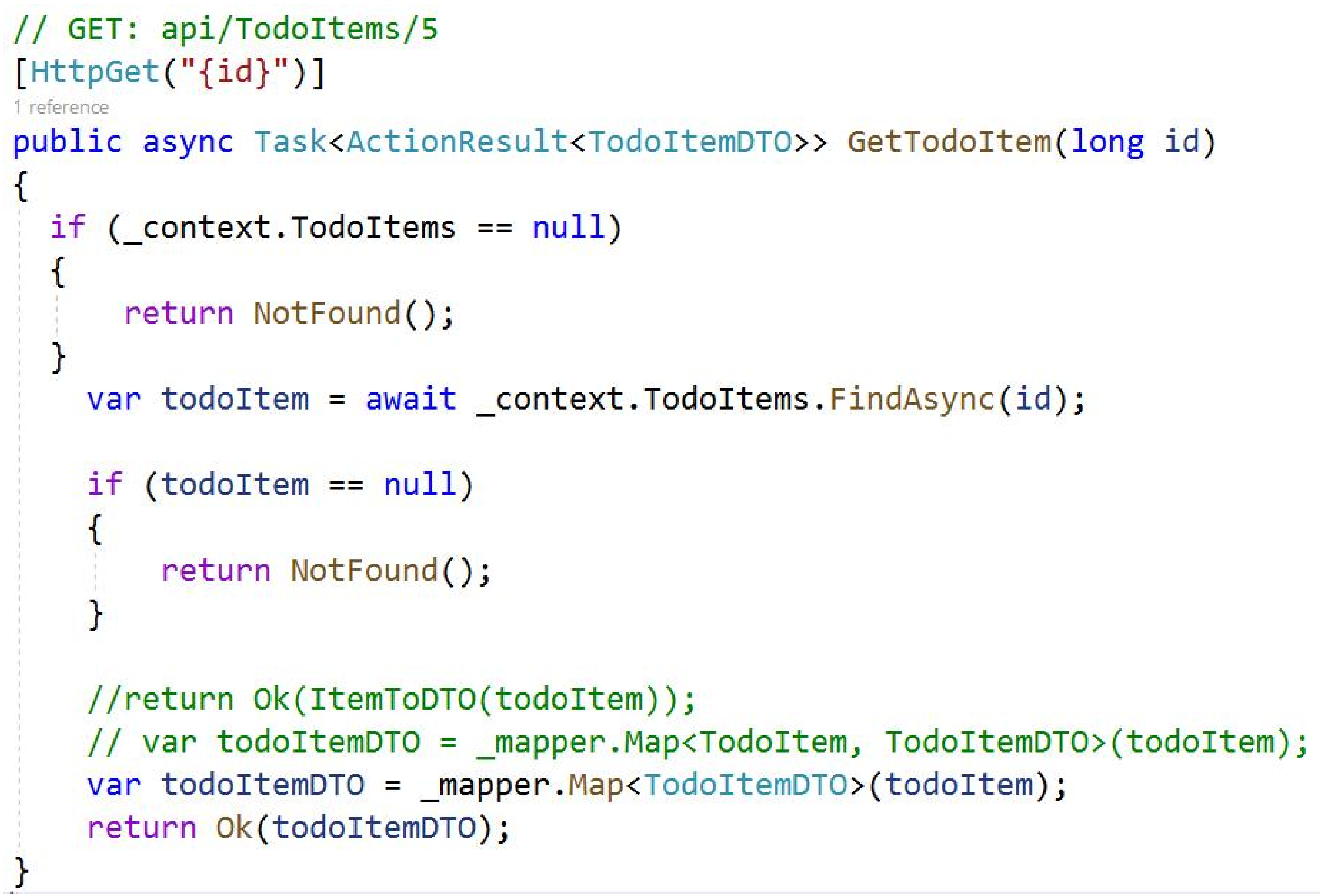
```
using AutoMapper;

namespace ToDoApi.Models
{
    2 references
    public class TodoItemProfile: Profile
    {
        0 references
        public TodoItemProfile()
        {
            CreateMap<TodoItem, TodoItemDTO>();
        }
    }
}
```



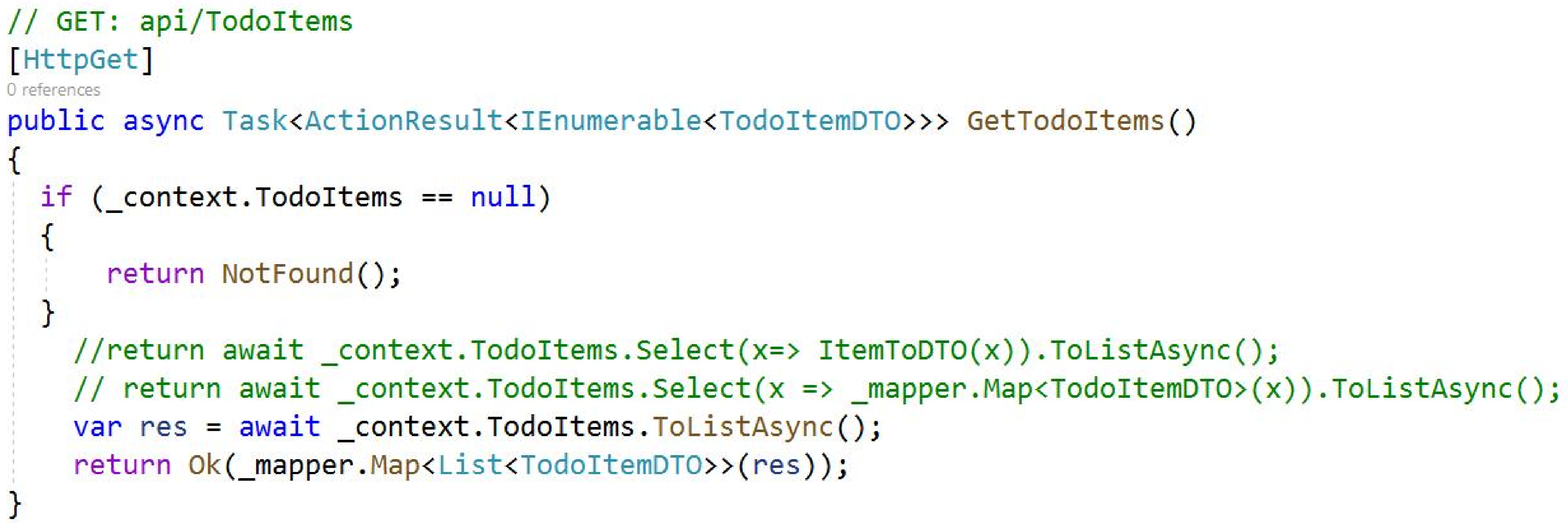
```
// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddDbContext<TodoContext>(opt => opt.UseInMemoryDatabase("TodoList"));
builder.Services.AddAutoMapper(typeof(TodoItemProfile));
```



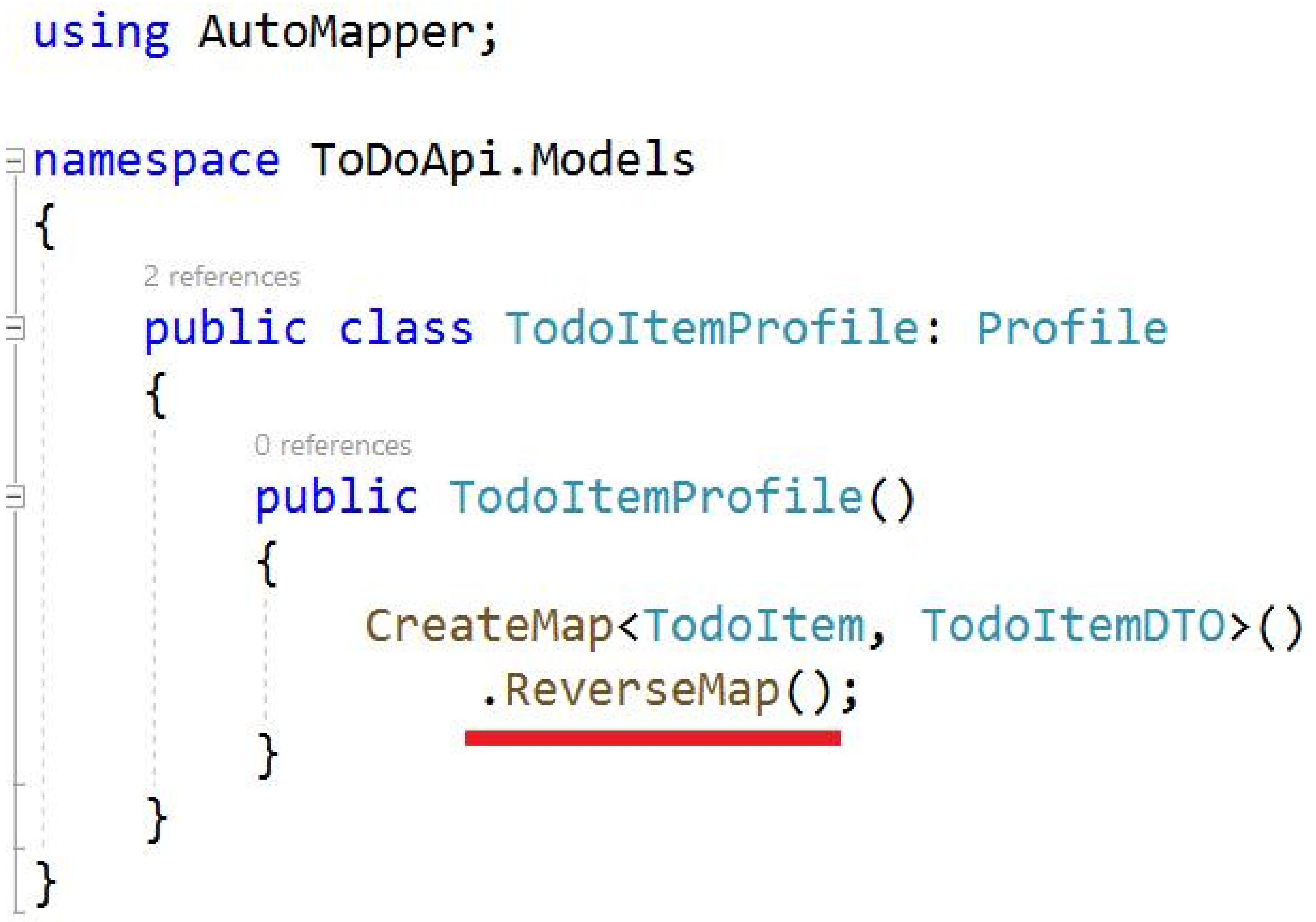
```
namespace ToDoApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    1 reference
    public class TodoItemsController : ControllerBase
    {
        private readonly TodoContext _context;
        private readonly IMapper _mapper;

        0 references
        public TodoItemsController(TodoContext context, IMapper mapper)
        {
            _context = context;
            _mapper = mapper;
        }
```



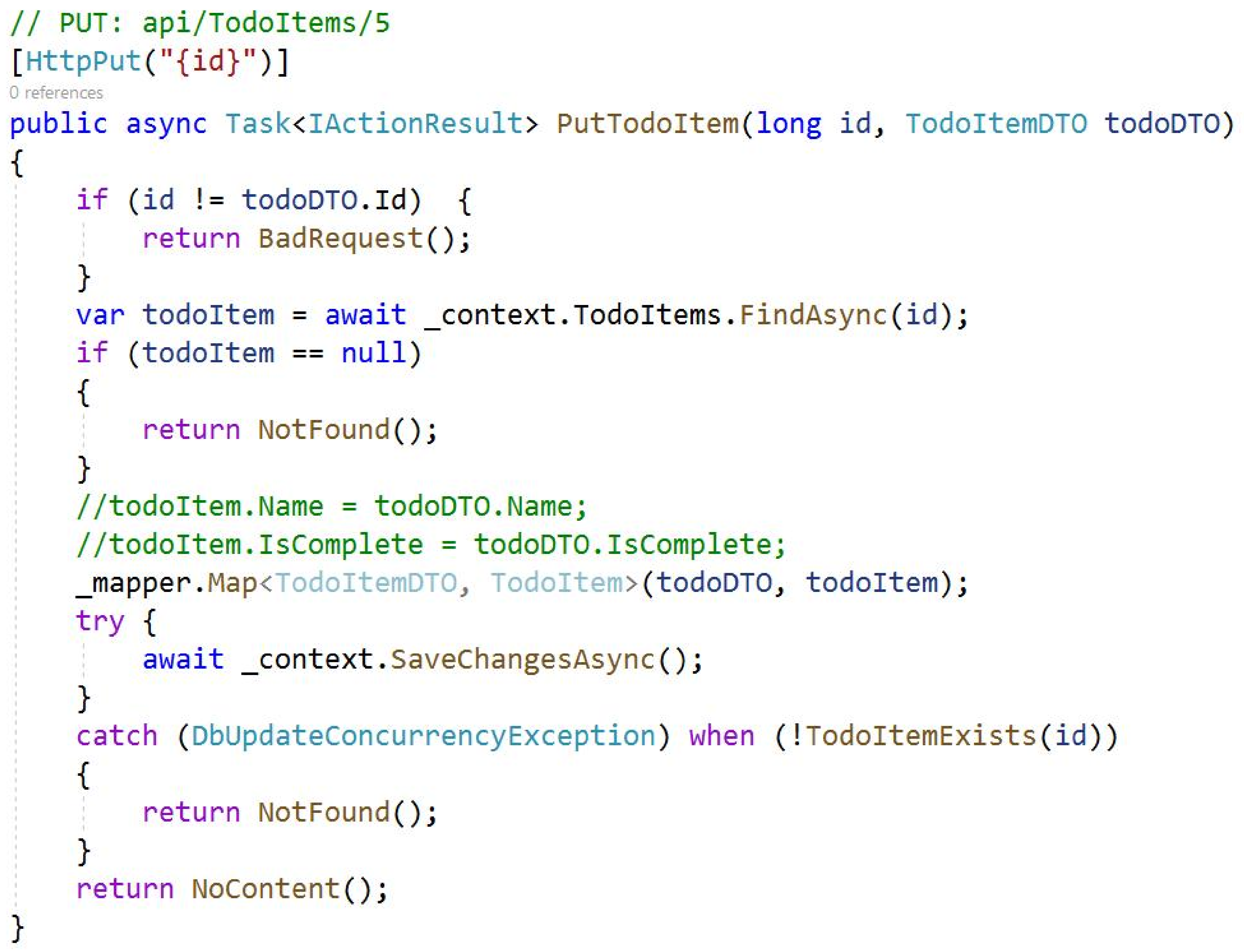
```
// GET: api/TodoItems/5
[HttpGet("{id}")]
1 reference
public async Task<ActionResult<TodoItemDTO>> GetTodoItem(long id)
{
  if (_context.TodoItems == null)
  {
      return NotFound();
  }
    var todoItem = await _context.TodoItems.FindAsync(id);

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

    //return Ok(ItemToDTO(todoItem));
    // var todoItemDTO = _mapper.Map<TodoItem, TodoItemDTO>(todoItem);
    var todoItemDTO = _mapper.Map<TodoItemDTO>(todoItem);
    return Ok(todoItemDTO);
}
```



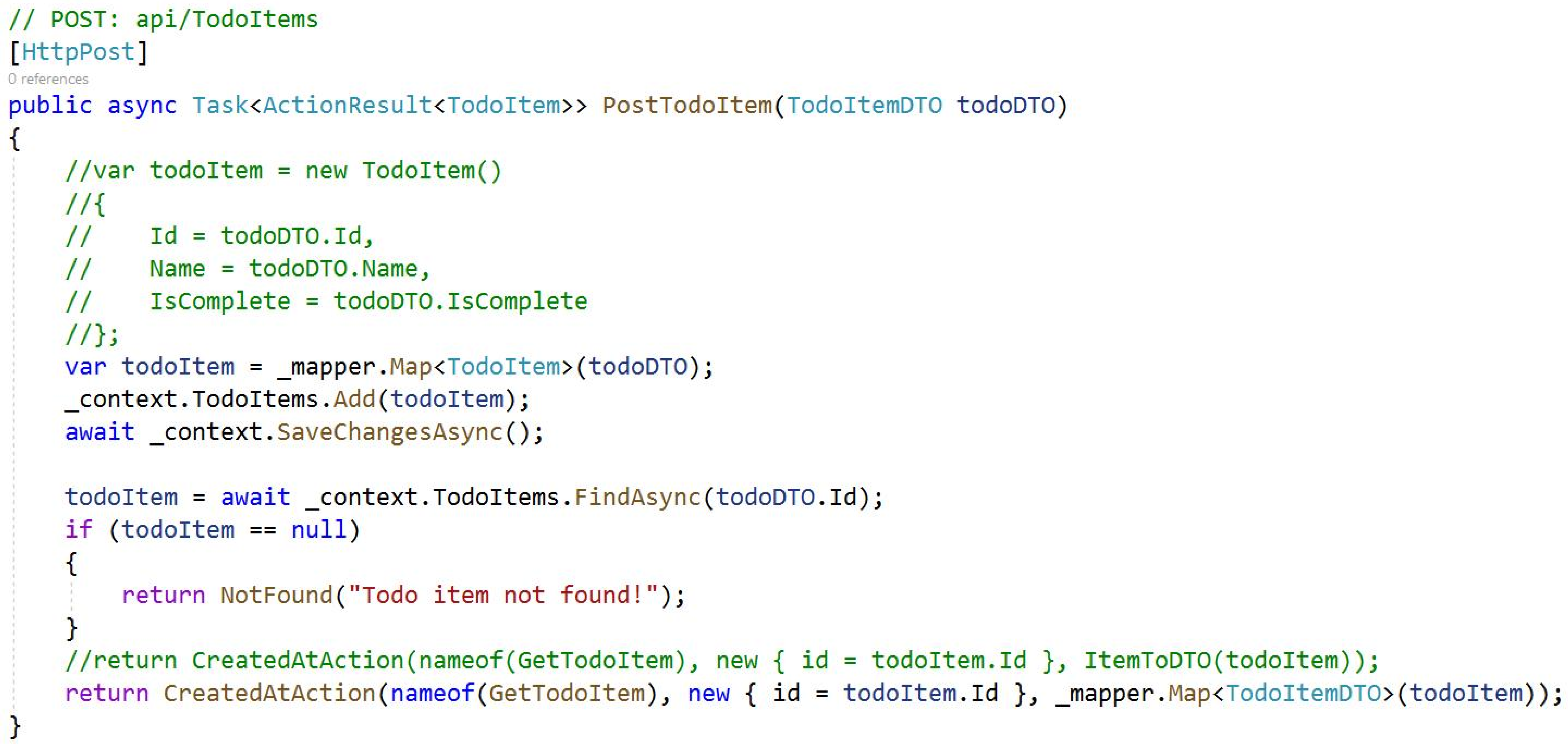
```
// GET: api/TodoItems
[HttpGet]
0 references
public async Task<ActionResult<IEnumerable<TodoItemDTO>>> GetTodoItems()
{
  if (_context.TodoItems == null)
  {
      return NotFound();
  }
    //return await _context.TodoItems.Select(x=> ItemToDTO(x)).ToListAsync();
    // return await _context.TodoItems.Select(x => _mapper.Map<TodoItemDTO>(x)).ToListAsync();
    var res = await _context.TodoItems.ToListAsync();
    return Ok(_mapper.Map<List<TodoItemDTO>>(res));
}
```



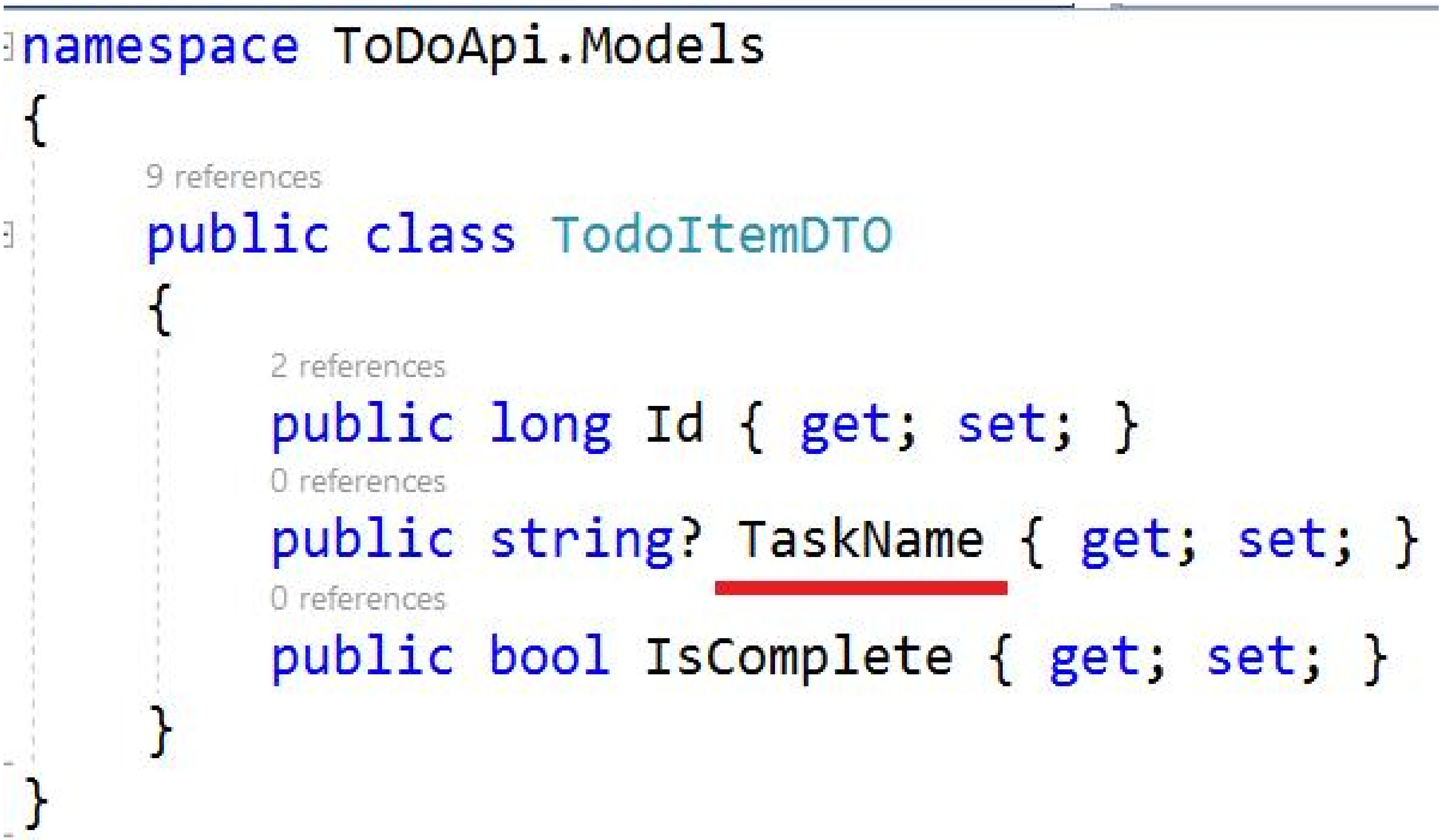
```
using AutoMapper;

namespace ToDoApi.Models
{
    2 references
    public class TodoItemProfile: Profile
    {
        0 references
        public TodoItemProfile()
        {
            CreateMap<TodoItem, TodoItemDTO>()
                .ReverseMap();
        }
    }
}
```



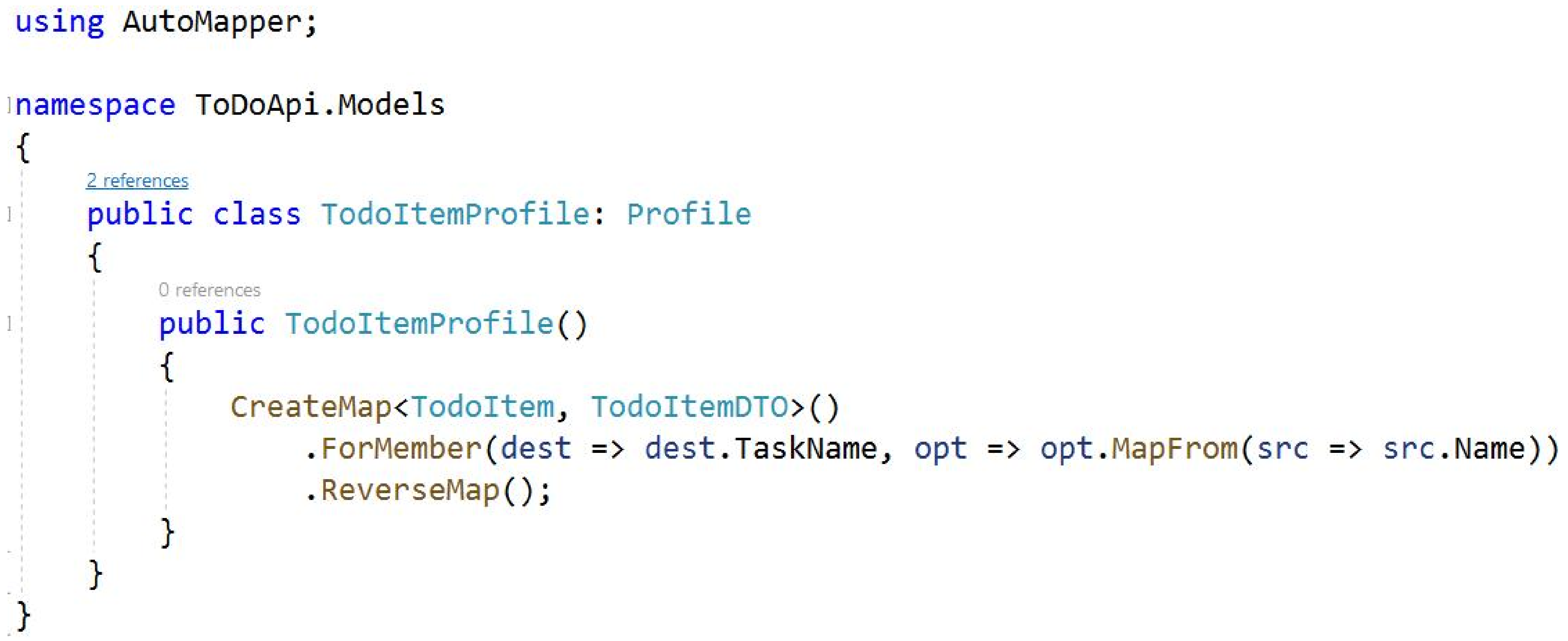
```
// PUT: api/TodoItems/5
[HttpPut("{id}")]
0 references
public async Task<IActionResult> PutTodoItem(long id, TodoItemDTO todoDTO)
{
    if (id != todoDTO.Id)  {
        return BadRequest();
    }
    var todoItem = await _context.TodoItems.FindAsync(id);
    if (todoItem == null)
    {
        return NotFound();
    }
    //todoItem.Name = todoDTO.Name;
    //todoItem.IsComplete = todoDTO.IsComplete;
    _mapper.Map<TodoItemDTO, TodoItem>(todoDTO, todoItem);
    try {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException) when (!TodoItemExists(id))
    {
        return NotFound();
    }
    return NoContent();
}
```



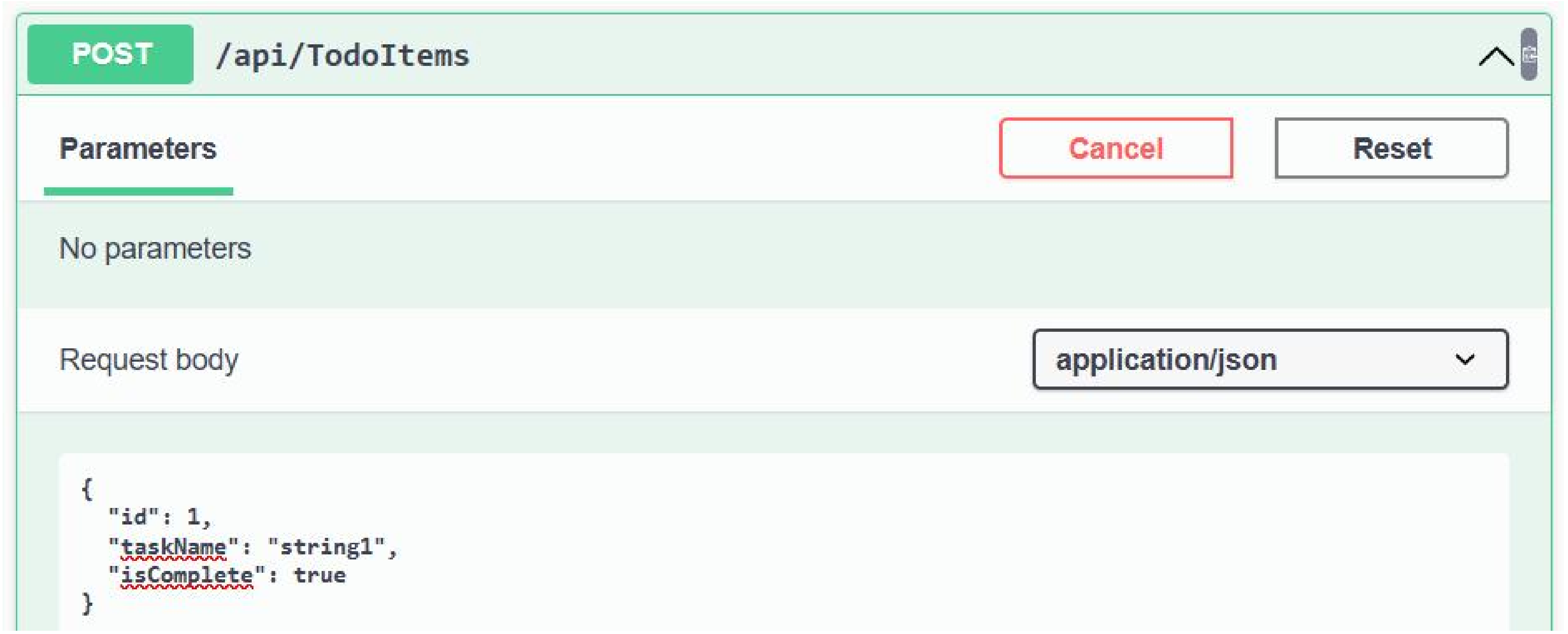
```csharp
// POST: api/TodoItems
[HttpPost]
0 references
public async Task<ActionResult<TodoItem>> PostTodoItem(TodoItemDTO todoDTO)
{
    //var todoItem = new TodoItem()
    //{
    //    Id = todoDTO.Id,
    //    Name = todoDTO.Name,
    //    IsComplete = todoDTO.IsComplete
    //};
    var todoItem = _mapper.Map<TodoItem>(todoDTO);
    _context.TodoItems.Add(todoItem);
    await _context.SaveChangesAsync();

    todoItem = await _context.TodoItems.FindAsync(todoDTO.Id);
    if (todoItem == null)
    {
        return NotFound("Todo item not found!");
    }
    //return CreatedAtAction(nameof(GetTodoItem), new { id = todoItem.Id }, ItemToDTO(todoItem));
    return CreatedAtAction(nameof(GetTodoItem), new { id = todoItem.Id }, _mapper.Map<TodoItemDTO>(todoItem));
}
```



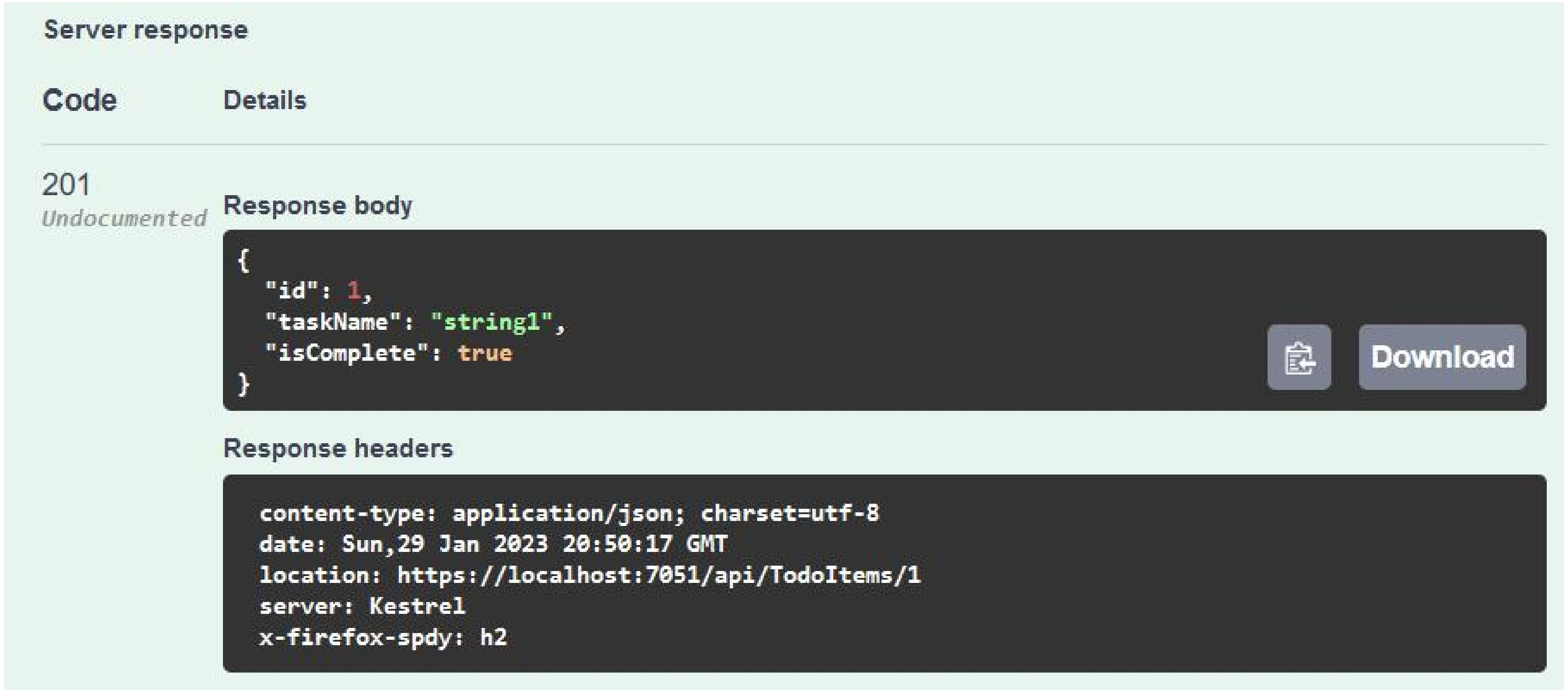
```
namespace ToDoApi.Models
{
        9 references
        public class TodoItemDTO
        {
            2 references
            public long Id { get; set; }
            0 references
            public string? TaskName { get; set; }
            0 references
            public bool IsComplete { get; set; }
        }
}
```



```
using AutoMapper;

namespace ToDoApi.Models
{
    2 references
    public class TodoItemProfile: Profile
    {
        0 references
        public TodoItemProfile()
        {
            CreateMap<TodoItem, TodoItemDTO>()
                .ForMember(dest => dest.TaskName, opt => opt.MapFrom(src => src.Name))
                .ReverseMap();
        }
    }
}
```

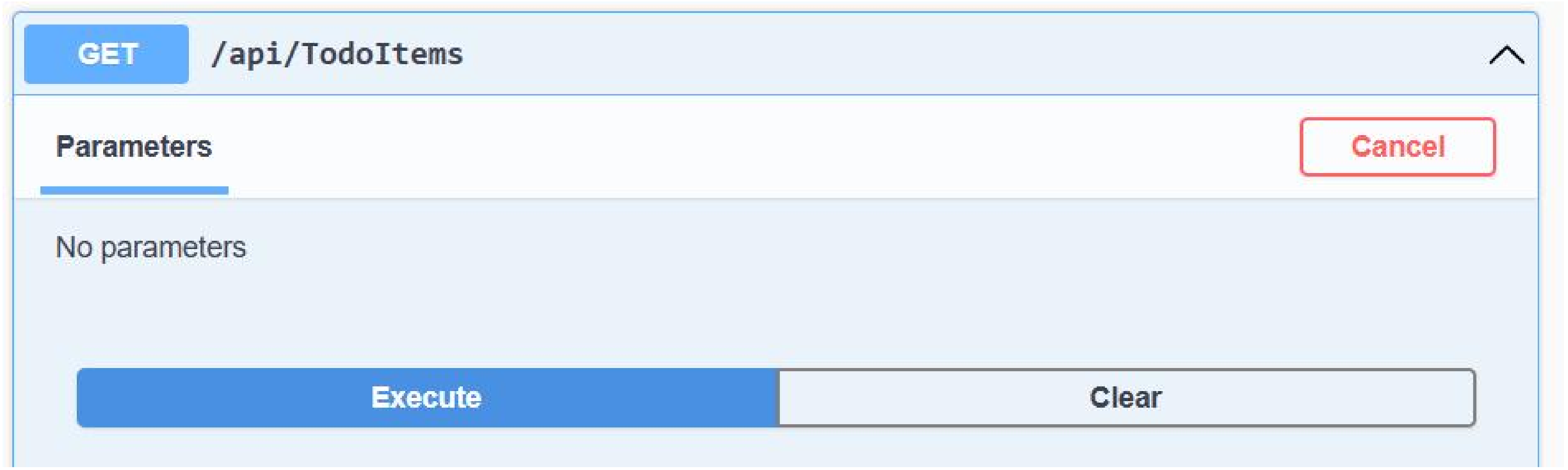


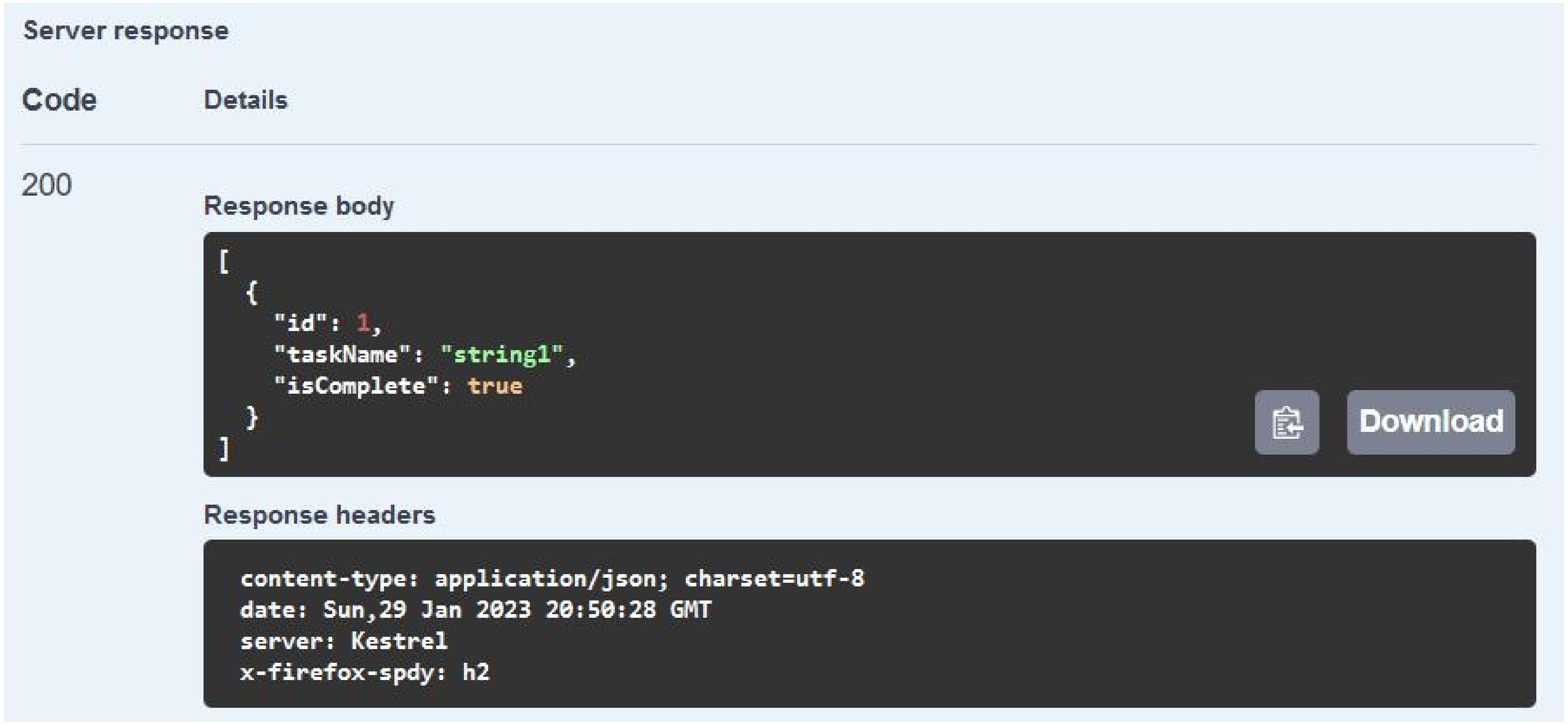
```
{
  "id": 1,
  "taskName": "string1",
  "isComplete": true
}
```



```
{
  "id": 1,
  "taskName": "string1",
  "isComplete": true
}
```

```
content-type: application/json; charset=utf-8
date: Sun,29 Jan 2023 20:50:17 GMT
location: https://localhost:7051/api/TodoItems/1
server: Kestrel
x-firefox-spdy: h2
```


GET
/api/TodoItems
Parameters
Cancel
No parameters
Execute
Clear



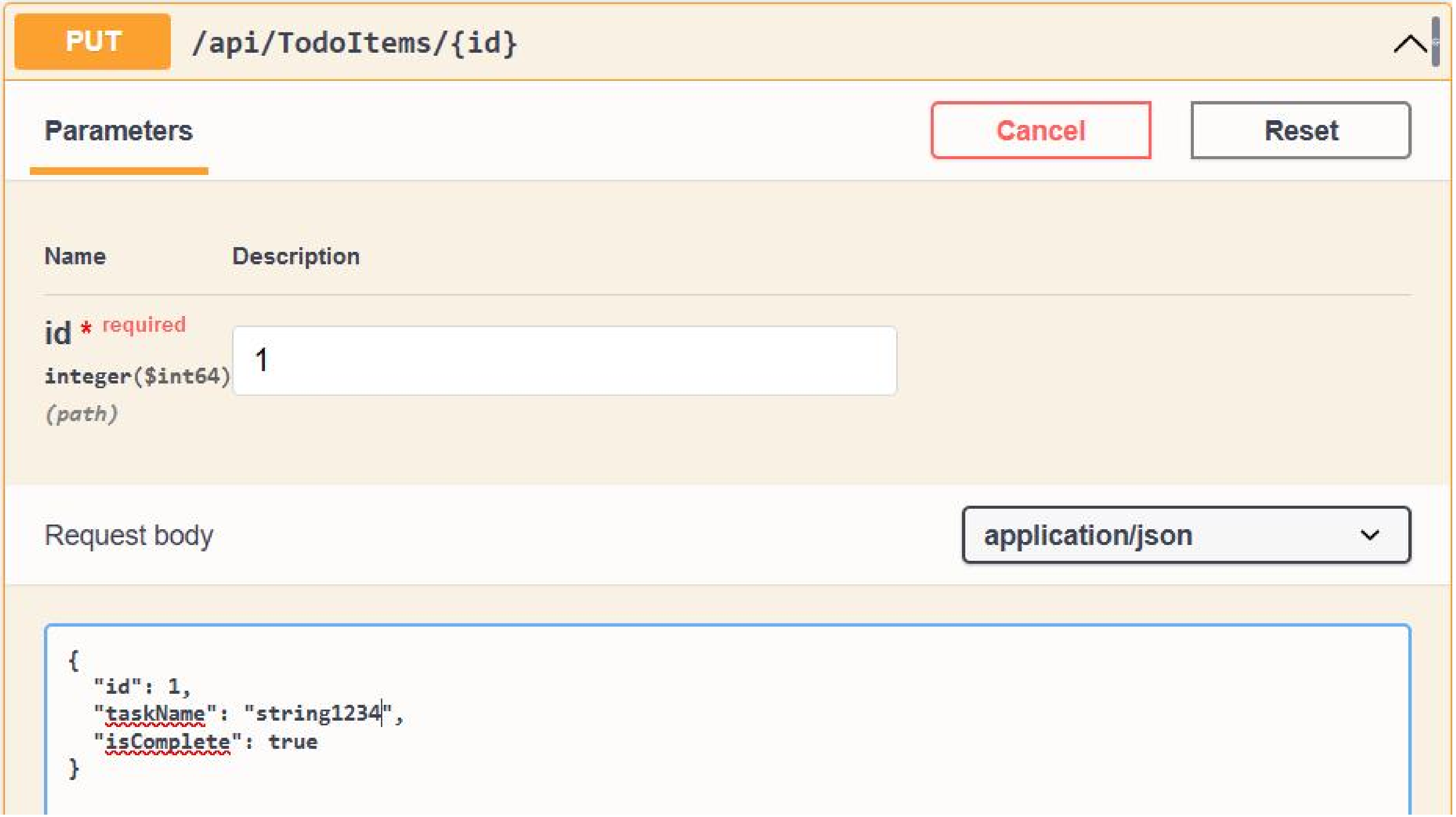
Server response

Code	Details
200	Response body

```
[
  {
    "id": 1,
    "taskName": "string1",
    "isComplete": true
  }
]
```

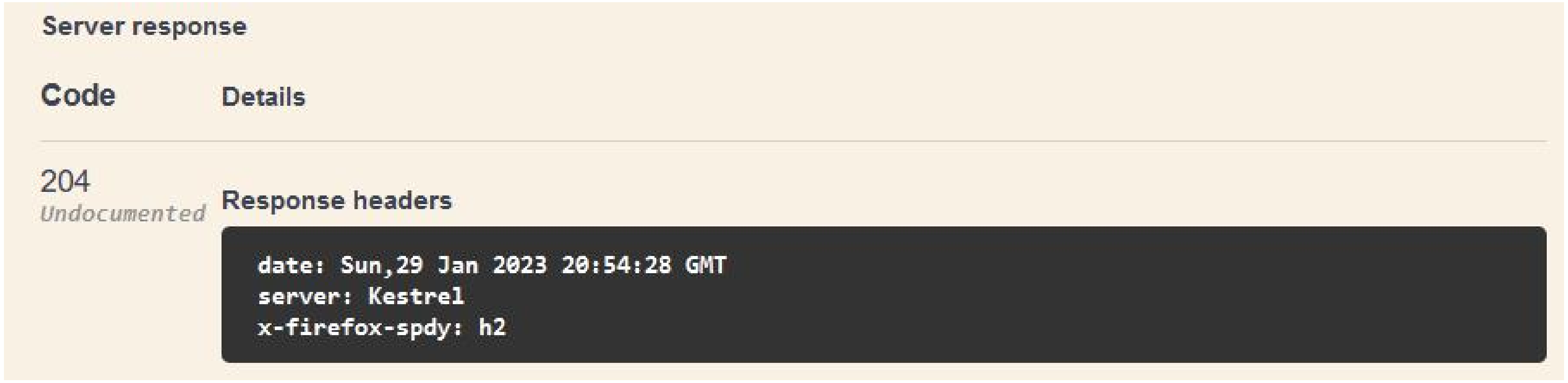
Response headers

```
content-type: application/json; charset=utf-8
date: Sun,29 Jan 2023 20:50:28 GMT
server: Kestrel
x-firefox-spdy: h2
```



id * required
integer($int64)
(path)
1

```
{
  "id": 1,
  "taskName": "string1234",
  "isComplete": true
}
```



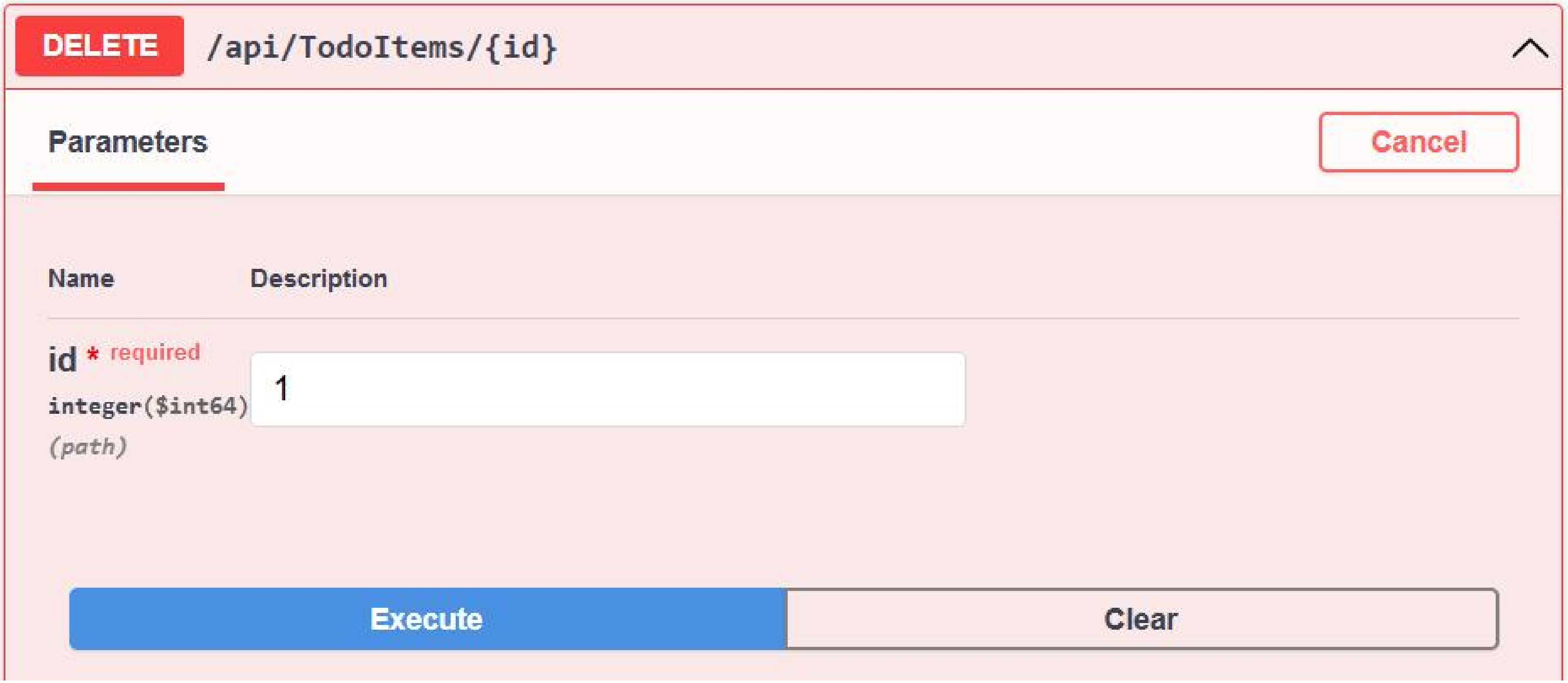
Server response

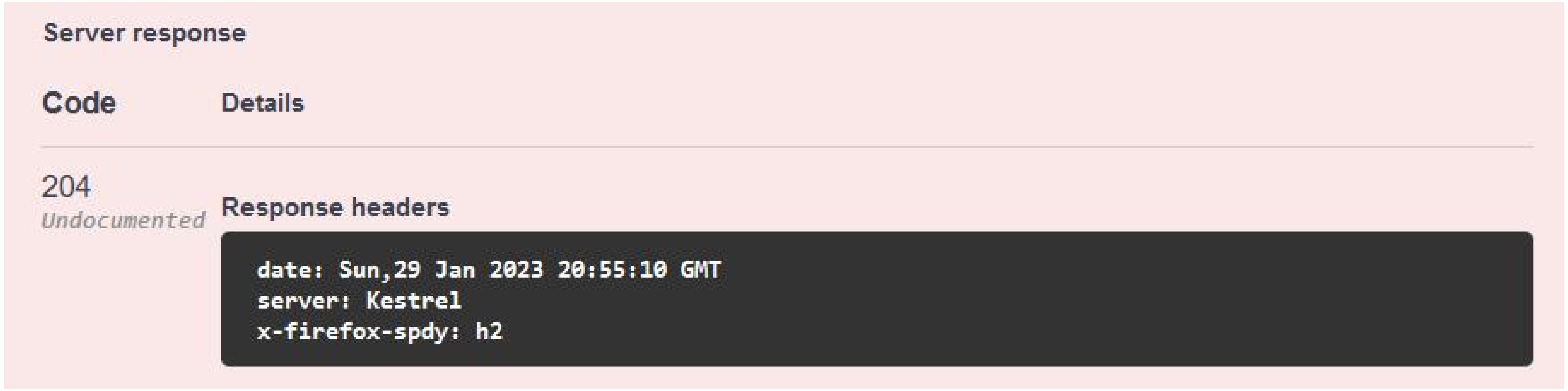
Code Details

204
Undocumented

Response headers

```
date: Sun,29 Jan 2023 20:54:28 GMT
server: Kestrel
x-firefox-spdy: h2
```


DELETE
/api/TodoItems/{id}
Parameters
Cancel
Name
Description
id * required
integer($int64)
(path)
1
Execute
Clear

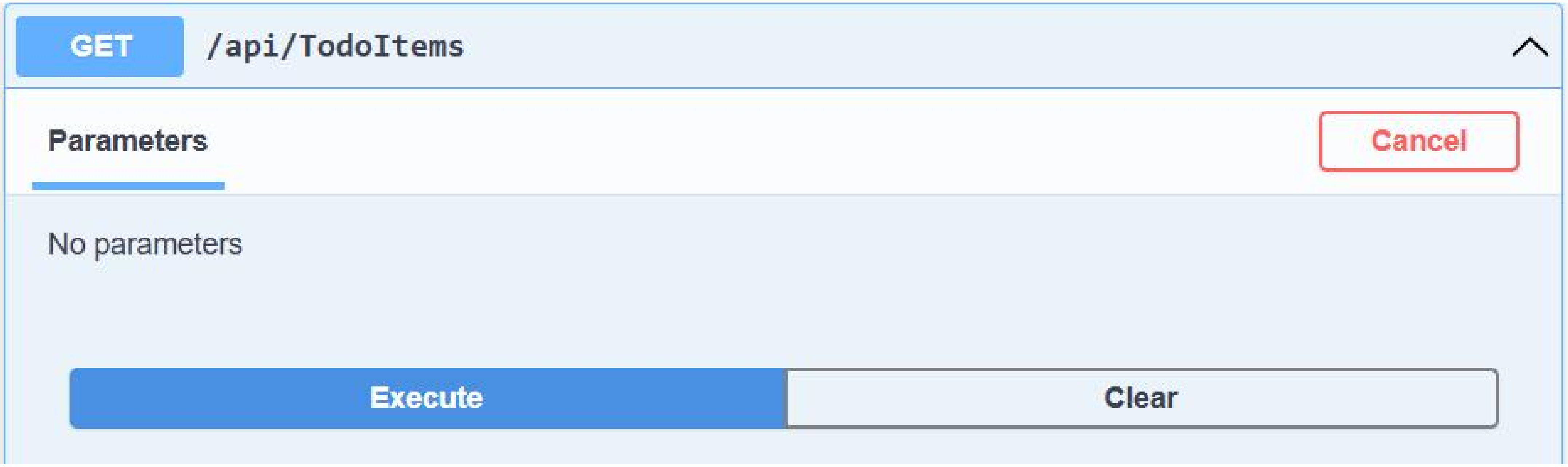

Server response

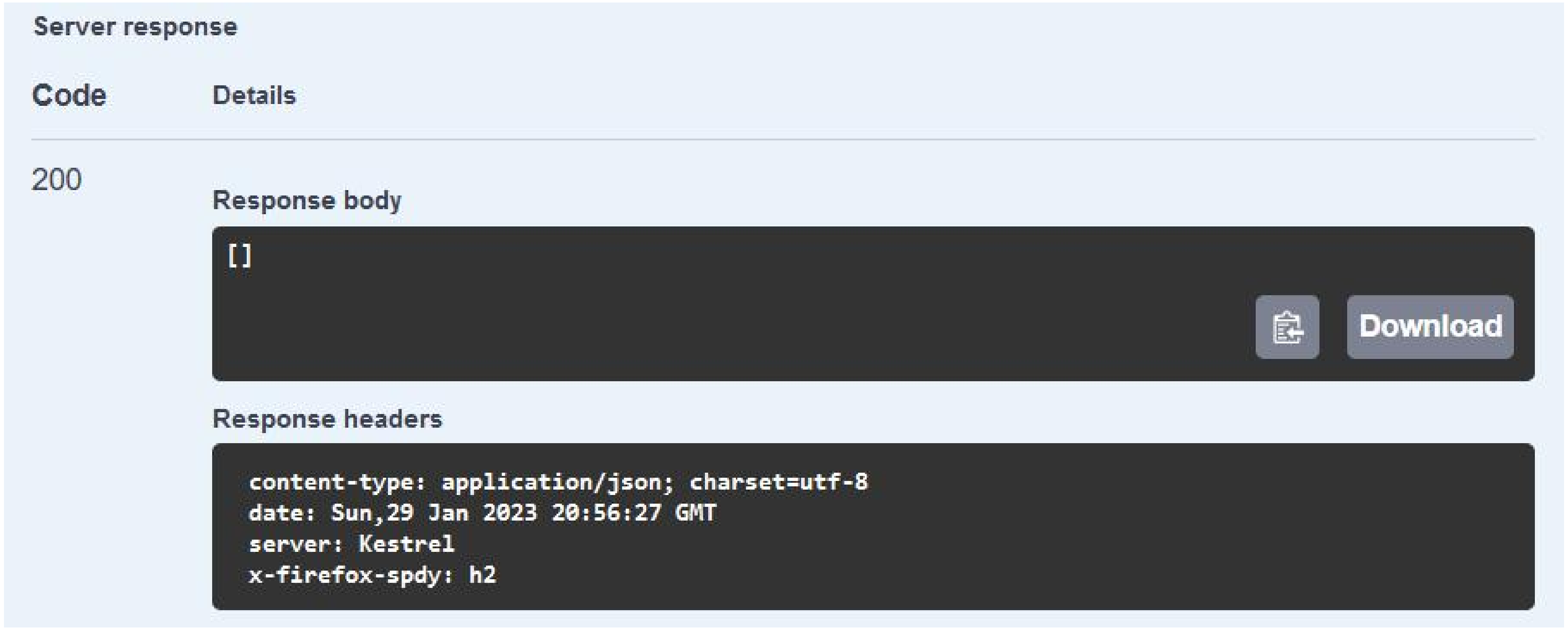
Code Details

204
Undocumented

Response headers

```
date: Sun,29 Jan 2023 20:55:10 GMT
server: Kestrel
x-firefox-spdy: h2
```

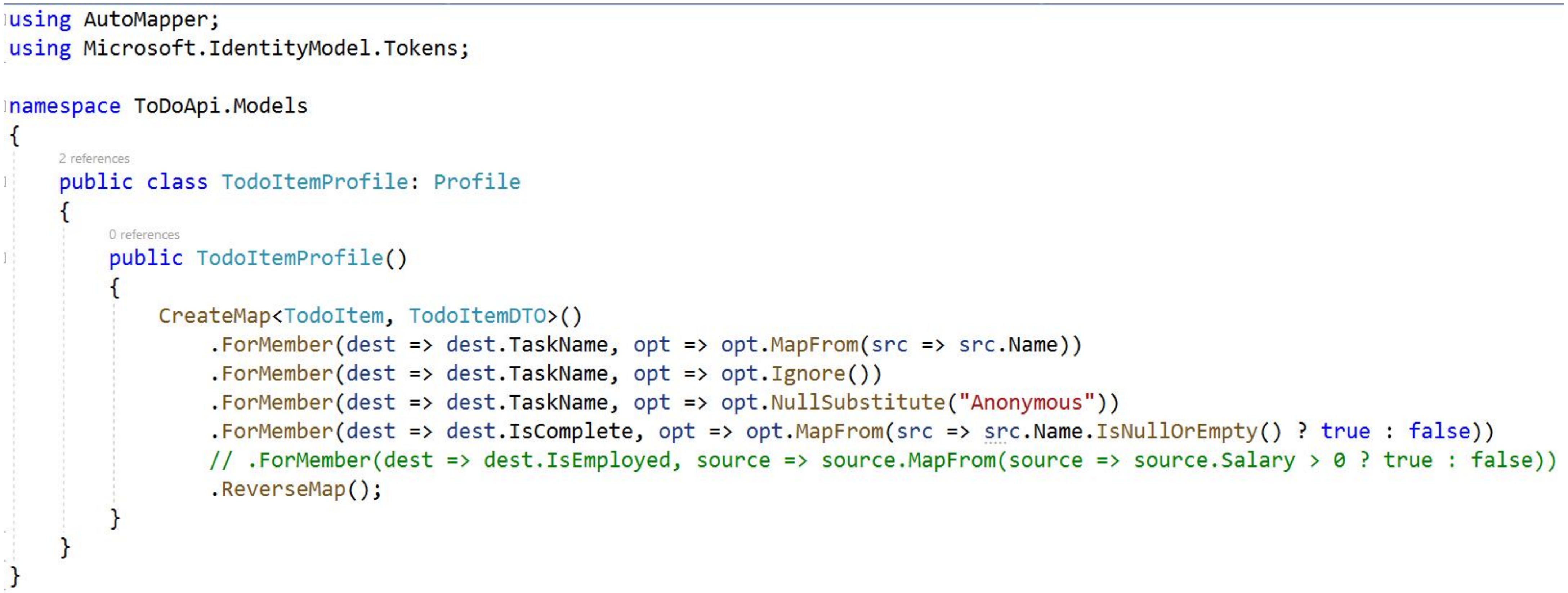

GET
/api/TodoItems
Parameters
Cancel
No parameters
Execute
Clear



200

```
[]
```

```
content-type: application/json; charset=utf-8
date: Sun,29 Jan 2023 20:56:27 GMT
server: Kestrel
x-firefox-spdy: h2
```



```
using AutoMapper;
using Microsoft.IdentityModel.Tokens;

namespace ToDoApi.Models
{
    2 references
    public class TodoItemProfile: Profile
    {
        0 references
        public TodoItemProfile()
        {
            CreateMap<TodoItem, TodoItemDTO>()
                .ForMember(dest => dest.TaskName, opt => opt.MapFrom(src => src.Name))
                .ForMember(dest => dest.TaskName, opt => opt.Ignore())
                .ForMember(dest => dest.TaskName, opt => opt.NullSubstitute("Anonymous"))
                .ForMember(dest => dest.IsComplete, opt => opt.MapFrom(src => src.Name.IsNullOrEmpty() ? true : false))
                // .ForMember(dest => dest.IsEmployed, source => source.MapFrom(source => source.Salary > 0 ? true : false))
                .ReverseMap();
        }
    }
}
```