

Service Lifetime

Transient

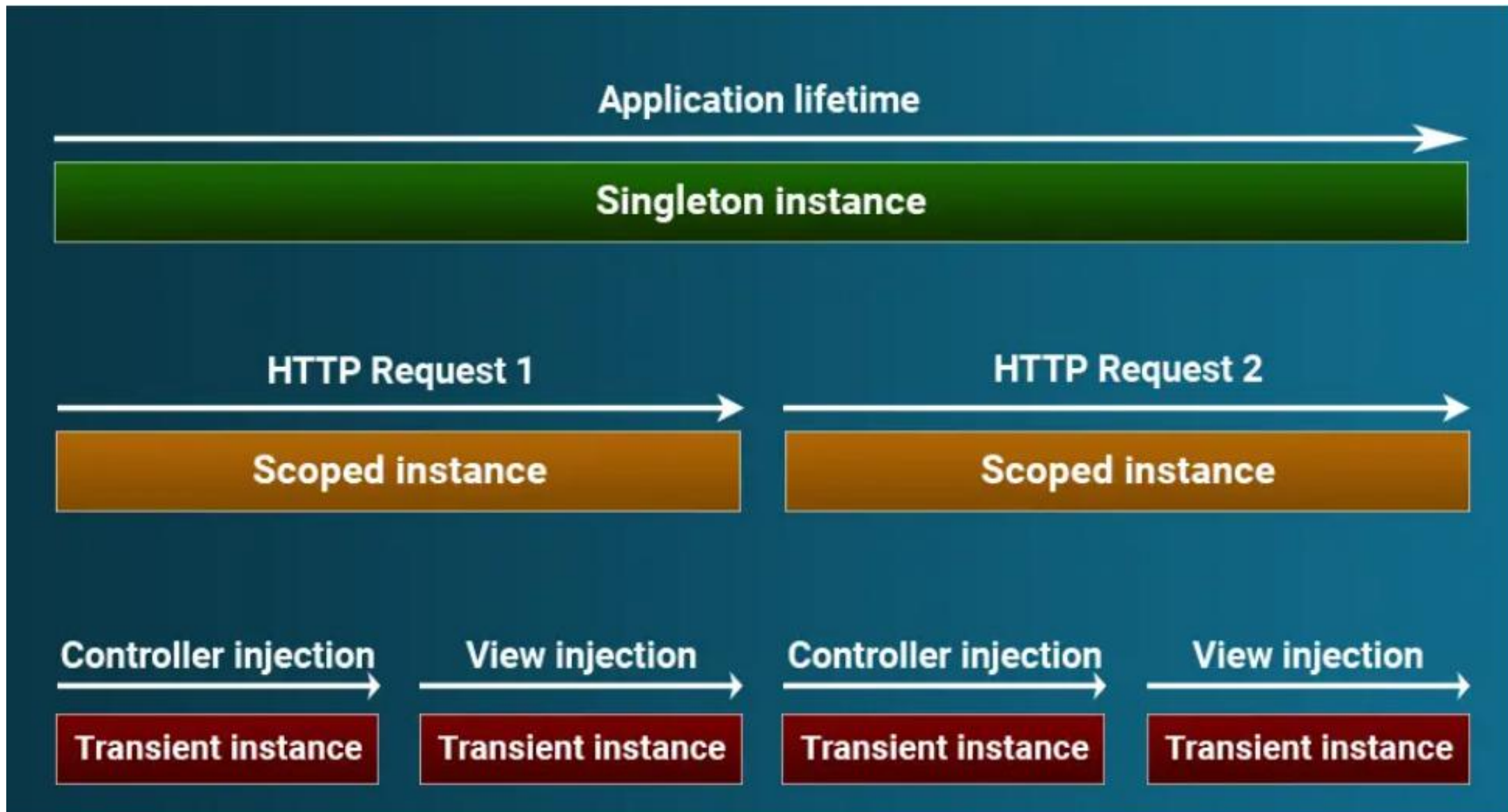
- An object is created every time it is requested from the DI container.
- Every time the caller asks for the type, the DI container will create a new instance of the type and it will return.
- The DI container disposes the object at the end of each request.

Scoped

- The instance will be created once per the request lifecycle.
- Meaning once for an HTTP request.
- Even if you ask the DI container five times, the same object created will be returned across the lifetime of this HTTP request.
- DI container disposes the scoped object at the end of the HTTP request.

Singleton

- A Singleton instance will be created only once for the entire life cycle of the application.
- Every subsequent request for the object from the dependency injection container will give the same instance.



```
namespace ToDoApi.Services
```

```
{
```

3 references

```
public interface IGetGuid
```

```
{
```

12 references

```
    string GetGuid();
```

```
}
```

```
}
```

```
namespace ToDoApi.Services
```

```
{
```

7 references

```
public interface IGuidTransient : IGetGuid
```

```
{
```

```
}
```

```
}
```

```
namespace ToDoApi.Services
```

```
{
```

7 references

```
public interface IGuidScoped : IGetGuid
```

```
{
```

```
}
```

```
}
```

```
namespace ToDoApi.Services
```

```
{
```

8 references

```
public interface IGuidSingleton : IGetGuid
```

```
{
```

```
}
```

```
}
```

```
namespace ToDoApi.Services
```

```
{
```

2 references

```
public class TransientService : IGuidTransient
```

```
{
```

```
    private readonly Guid Id;
```

0 references

```
    public TransientService()
```

```
    {
```

```
        Id= Guid.NewGuid();
```

```
    }
```

10 references

```
    public string GetGuid()
```

```
    {
```

```
        return Id.ToString();
```

```
    }
```

```
}
```

```
}
```

```
namespace ToDoApi.Services
```

```
{
```

2 references

```
public class ScopedService : IGuidScoped
{
```

```
    private readonly Guid Id;
```

0 references

```
    public ScopedService()
    {
```

```
        Id = Guid.NewGuid();
```

```
    }
```

10 references

```
    public string GetGuid()
    {
```

```
        return Id.ToString();
```

```
    }
```

```
}
```

```
}
```



```
namespace ToDoApi.Services
```

```
{
```

2 references

```
public class SingletonService : IGuidSingleton
```

```
{
```

```
    private readonly Guid Id;
```

0 references

```
    public SingletonService()
```

```
    {
```

```
        Id = Guid.NewGuid();
```

```
    }
```

10 references

```
    public string GetGuid()
```

```
    {
```

```
        return Id.ToString();
```

```
    }
```

```
}
```

```
}
```

```
builder.Services.AddTransient<IGuidTransient, TransientService>();  
builder.Services.AddScoped<IGuidScoped, ScopedService>();  
builder.Services.AddSingleton<IGuidSingleton, SingletonService>();
```

```

using Microsoft.AspNetCore.Mvc;
using System.Text;

namespace ToDoApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    1 reference
    public class SvcLifeTimeController : ControllerBase
    {
        private readonly IGuidTransient _transientService1, _transientService2;
        private readonly IGuidScoped _scopedService1, _scopedService2;
        private readonly IGuidSingleton _singletonService1, _singletonService2;
        0 references
        public SvcLifeTimeController(
            IGuidTransient t1, IGuidTransient t2,
            IGuidScoped sco1, IGuidScoped sco2,
            IGuidSingleton s1, IGuidSingleton s2 )
        {
            _transientService1 = t1; _transientService2 = t2; _scopedService1 = sco1;
            _scopedService2 = sco2; _singletonService1 = s1; _singletonService2 = s2;
        }
        [HttpGet]
        0 references
        public IActionResult GetGuid()
        {

```

[HttpGet]

0 references

public IActionResult GetGuid()

{

 StringBuilder sb = new StringBuilder();

 sb.Append(\$"Transient 1 : {_transientService1.GetGuid()}\n");

 sb.Append(\$"Transient 2 : {_transientService2.GetGuid()}\n");

 sb.Append(\$"Scoped 1 : {_scopedService1.GetGuid()}\n");

 sb.Append(\$"Scoped 2 : {_scopedService2.GetGuid()}\n");

 sb.Append(\$"Singleton 1 : {_singletonService1.GetGuid()}\n");

 sb.Append(\$"Singleton 2 : {_singletonService2.GetGuid()}\n");

 return Ok(sb.ToString());

}

}

}

Swagger UI

× +

https://localhost:7051/swagger/index.html

120%


☆

🛡️

📄

🔖

☰

 **Swagger**
Supported by SMARTBEAR

Select a definition

ToDoApi v1

ToDoApi

1.0 OAS3

<https://localhost:7051/swagger/v1/swagger.json>

SvcLifeTime

^

GET /api/SvcLifeTime

^

Parameters

Cancel

No parameters

Execute

Clear

Request URL

`https://localhost:7051/api/SvcLifeTime`

Server response

Code	Details
------	---------

200

Response body

```
Transient 1 : 0af58691-4bbd-4e12-8684-4a235263c931
Transient 2 : ce4e0b04-5c5b-4493-98c5-ab0975638996
Scoped 1 : a2adbe53-a52c-4b3b-b5f3-53f5a220cd7e
Scoped 2 : a2adbe53-a52c-4b3b-b5f3-53f5a220cd7e
Singleton 1 : a47447b2-ae12-4713-8461-fe395ed189cc
Singleton 2 : a47447b2-ae12-4713-8461-fe395ed189cc
```



Download

Response headers

```
content-type: text/plain; charset=utf-8
date: Thu,02 Feb 2023 10:06:25 GMT
server: Kestrel
x-firefox-spdy: h2
```

SvcLifeTime



GET

/api/SvcLifeTime



Parameters

Cancel

No parameters

Execute

Clear

Server response

Code	Details
------	---------

200	
-----	--

Response body

```
Transient 1 : 9e52eb7c-cdb6-4580-a892-5638f4033725
Transient 2 : 13cd3c2a-46ee-4283-aa0e-c89073eca897
Scoped 1 : ad81878e-4e9f-4910-a8e5-9b5b8a4cd3ef
Scoped 2 : ad81878e-4e9f-4910-a8e5-9b5b8a4cd3ef
Singleton 1 : a47447b2-ae12-4713-8461-fe395ed189cc
Singleton 2 : a47447b2-ae12-4713-8461-fe395ed189cc
```

[Download](#)

Response headers

```
content-type: text/plain; charset=utf-8
date: Thu,02 Feb 2023 10:35:48 GMT
server: Kestrel
x-firefox-spdy: h2
```


Controller with TodoService

```
using AutoMapper;
using Microsoft.EntityFrameworkCore;

namespace ToDoApi.Services
{
    // Todo Service which uses Automapper library
    2 references
    public class TodoService2 : ITodoService
    {
        private readonly TodoContext _context;
        private readonly IMapper _mapper;

        0 references
        public TodoService2(TodoContext todoContext, IMapper mapper)
        {
            _context = todoContext;
            _mapper = mapper;
        }
    }
}
```

```
public async Task<TodoItemDTO> GetTodoItem(long id)
{
    if (_context.TODOItems == null)
    {
        return null;
    }
    var todoItem = await _context.TODOItems.FindAsync(id);

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

    //return ItemToDTO(todoItem);
    return _mapper.Map<TodoItemDTO>(todoItem);
}
```

```
public async Task<IEnumerable<TodoItemDTO>> GetTodoItems()
{
    return await _context.TodoItems.Select(x => _mapper.Map<TodoItemDTO>(x)).ToListAsync();
}
```

```
public async Task<TodoItemDTO> AddTodoItem(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();

    return todoDTO;
}
```



```
public async Task<TodoItemDTO> UpdateTodoItem(long id, TodoItemDTO todoDTO)
{
    var todoItem = await _context.TodoItems.FindAsync(id);

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

    //todoItem.Name = todoDTO.Name;
    //todoItem.IsComplete = todoDTO.IsComplete;
    _mapper.Map<TodoItemDTO, TodoItem>(todoDTO, todoItem);

    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException) when (!TodoItemExists(id))
    {
        return null;
    }
    return todoDTO;
}
```

```
public async Task<TodoItemDTO> DeleteTodoItem(long id)
{
    if (_context.TODOItems == null)
    {
        return null;
    }
    var todoItem = await _context.TODOItems.FindAsync(id);
    if (todoItem == null)
    {
        return null;
    }

    _context.TODOItems.Remove(todoItem);
    await _context.SaveChangesAsync();

    //return ItemToDTO(todoItem);
    return _mapper.Map<TodoItemDTO>(todoItem);
}
```

```
private bool TodoItemExists(long id)
{
    return (_context.TodoItems?.Any(e => e.Id == id)).GetValueOrDefault();
}
```



```
builder.Services.AddScoped<ITodoService, TodoService2>();
```

```
using Microsoft.AspNetCore.Mvc;
```

```
namespace ToDoApi.Controllers  
{
```

```
    [Route("api/[controller]")]  
    [ApiController]
```

1 reference

```
    public class TodoItems2Controller : ControllerBase  
    {
```

```
        private IToDoService _todoService;
```

0 references

```
        public TodoItems2Controller(IToDoService todoService)  
        {  
            _todoService = todoService;  
        }  
    }
```

```
// GET: api/ToDoItems2
```

```
[HttpGet]
```

```
0 references
```

```
public async Task<ActionResult<IEnumerable<ToDoItemDTO>>> GetToDoItems()  
{  
    var res = await _todoService.GetToDoItems();  
    if (res == null)  
    {  
        return NotFound();  
    }  
    return Ok(res);  
}
```

```
// GET: api/ToDoItems2/5
```

```
[HttpGet("{id}")]
```

1 reference

```
public async Task<ActionResult<ToDoItemDTO>> GetToDoItem(long id)
{
    var todoItemDTO = await _todoService.GetToDoItem(id);

    if (todoItemDTO == null)
    {
        return NotFound(id);
    }
    return Ok(todoItemDTO);
}
```

```
// PUT: api/ToDoItems2/
```

```
[HttpPut("{id}")]
```

0 references

```
public async Task<IActionResult> PutToDoItem(long id, ToDoItemDTO todoDTO)
{
    if (id != todoDTO.Id)
    {
        return BadRequest();
    }

    var todoItem = await _todoService.UpdateToDoItem(id, todoDTO);

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

    return NoContent();
}
```

```
// POST: api/ToDoItems2
```

```
[HttpPost]
```

0 references

```
public async Task<ActionResult<ToDoItem>> PostToDoItem(ToDoItemDTO todoDTO)
```

```
{
```

```
    var todoItemDTO = await _todoService.AddToDoItem(todoDTO);
```

```
    return CreatedAtAction(nameof(GetToDoItem), new { id = todoDTO.Id }, todoItemDTO);
```

```
}
```



```
// DELETE: api/TodoItems2/5
```

```
[HttpDelete("{id}")]
```

0 references

```
public async Task<IActionResult> DeleteTodoItem(long id)
{
    var todoItemDTO = await _todoService.DeleteTodoItem(id);

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

    return NoContent();
}
```

Action Method Parameter Binding


```
// POST: api/ToDoItems2
```

```
[HttpPost]
```

```
//public async Task<ActionResult<ToDoItem>> PostToDoItem(ToDoItemDTO todoDTO)
```

0 references

```
public async Task<ActionResult<ToDoItem>> PostToDoItem([FromHeader] ToDoItemDTO todoDTO)
{
    var todoItemDTO = await _todoService.AddToDoItem(todoDTO);

    return CreatedAtAction(nameof(GetToDoItem), new { id = todoDTO.Id }, todoItemDTO);
}
```

POST

https://localhost:7051/api/todoitems2?id=33&name=test&iscomplete=false

Send

Params • Authorization Headers (7) Body Pre-request Script Tests Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	id	33			
<input checked="" type="checkbox"/>	name	test			
<input checked="" type="checkbox"/>	iscomplete	false			
	Key	Value	Description		

Body Cookies Headers (5) Test Results



Status: 201 Created

Time: 105 ms

Size: 247 B

Save Response

Pretty

Raw

Preview

Visualize

JSON



```
1 {  
2   "id": 33,  
3   "name": "test",  
4   "isComplete": false  
5 }
```

GET



https://localhost:7051/api/todoitems2/

Send

[Params](#) [Auth](#) [Headers \(6\)](#) [Body](#) [Pre-req.](#) [Tests](#) [Settings](#)[Cookies](#)

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body



200 OK 85 ms 192 B

[Save Response](#)

Pretty

Raw

Preview

Visualize

JSON



```
1  [
2    {
3      "id": 33,
4      "name": "test",
5      "isComplete": false
6    }
7  ]
```

```
// PUT: api/ToDoItems2
```

```
[HttpPut]
```

0 references

```
public async Task<IActionResult> PutToDoItem2([FromBody] long Id, string Name, Boolean isComplete)
{
    var todoDTO = new ToDoItemDTO() { Id = Id, Name = Name, IsComplete = isComplete};
    var todoItem = await _todoService.UpdateToDoItem(todoDTO.Id, todoDTO);

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

    return Ok(todoDTO);
}
```

PUT ▼ https://localhost:7051/api/todoitems2?name=test-updated&iscomplete=true

Send ▼

Params ● Auth Headers (9) Body ● Pre-req. Tests Settings

Cookies

Headers 👁 8 hidden

	KEY	VALUE	DESCR	...	Bulk Edit	Presets ▼
<input checked="" type="checkbox"/>	Content-Type	application/json				
	Key	Value	Description			

PUT ▼ https://localhost:7051/api/todoitems2?name=test-updated&iscomplete=true

Send ▼

Params ● Auth Headers (9) Body ● Pre-req. Tests Settings

Cookies

raw ▼ JSON ▼

Beautify

1 33

Body Cookies Headers (4) Test Results



200 OK

73 ms

197 B

Save Response ▾

Pretty

Raw

Preview

Visualize

JSON ▾



```
1 {  
2   "id": 33,  
3   "name": "test-updated",  
4   "isComplete": true  
5 }
```

GET



https://localhost:7051/api/todoitems2

Send



Params

Auth

Headers (7)

Body

Pre-req.

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (4)

Test Results



200 OK

52 ms

199 B

Save Response



Pretty

Raw

Preview

Visualize

JSON



```
1  [
2    {
3      "id": 33,
4      "name": "test-updated",
5      "isComplete": true
6    }
7  ]
```



```
// PUT: api/ToDoItems2/  
[HttpPut("{id}")]  
//public async Task<IActionResult> PutToDoItem(long id, ToDoItemDTO todoDTO)  
0 references  
public async Task<IActionResult> PutToDoItem([FromBody] long todo_id,  
                                              [FromHeader] ToDoItemDTO todoDTO)  
{  
    if (todo_id != todoDTO.Id)  
    {  
        return BadRequest();  
    }  
  
    //var todoItem = await _todoService.UpdateToDoItem(id, todoDTO);  
    var todoItem = await _todoService.UpdateToDoItem(todo_id, todoDTO);  
  
    if (todoItem == null)  
    {  
        return NotFound();  
    }  
  
    return Ok(todoDTO);  
}
```

PUT



https://localhost:7051/api/todoitems2?name=test-new&iscomplete=false&id=33

Send



Params ●

Auth

Headers (9)

Body ●

Pre-req.

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	test-new			
<input checked="" type="checkbox"/>	iscomplete	false			
<input checked="" type="checkbox"/>	id	33			
	Key	Value	Description		

PUT



https://localhost:7051/api/todoitems2?name=test-new&iscomplete=false&id=33

Send



Params ●

Auth

Headers (9)

Body ●

Pre-req.

Tests

Settings

Cookies

raw



JSON



Beautify

1 33

2

PUT



https://localhost:7051/api/todoitems2?name=test-new&iscomplete=false&id=33

Send

[Params](#) [Auth](#) [Headers \(9\)](#) [Body](#) [Pre-req.](#) [Tests](#) [Settings](#)[Cookies](#)

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	test-new			
<input checked="" type="checkbox"/>	iscomplete	false			
<input checked="" type="checkbox"/>	id	33			
	Key	Value	Description		

[Body](#) [Cookies](#) [Headers \(4\)](#) [Test Results](#)

200 OK 56 ms 194 B

[Save Response](#)

Pretty

Raw

Preview

Visualize

JSON



```
1 {  
2   "id": 33,  
3   "name": "test-new",  
4   "isComplete": false  
5 }
```

GET

▼

https://localhost:7051/api/todoitems2

Send

▼

ParamsAuthHeaders (6)BodyPre-req. TestsSettingsCookies

Query Params

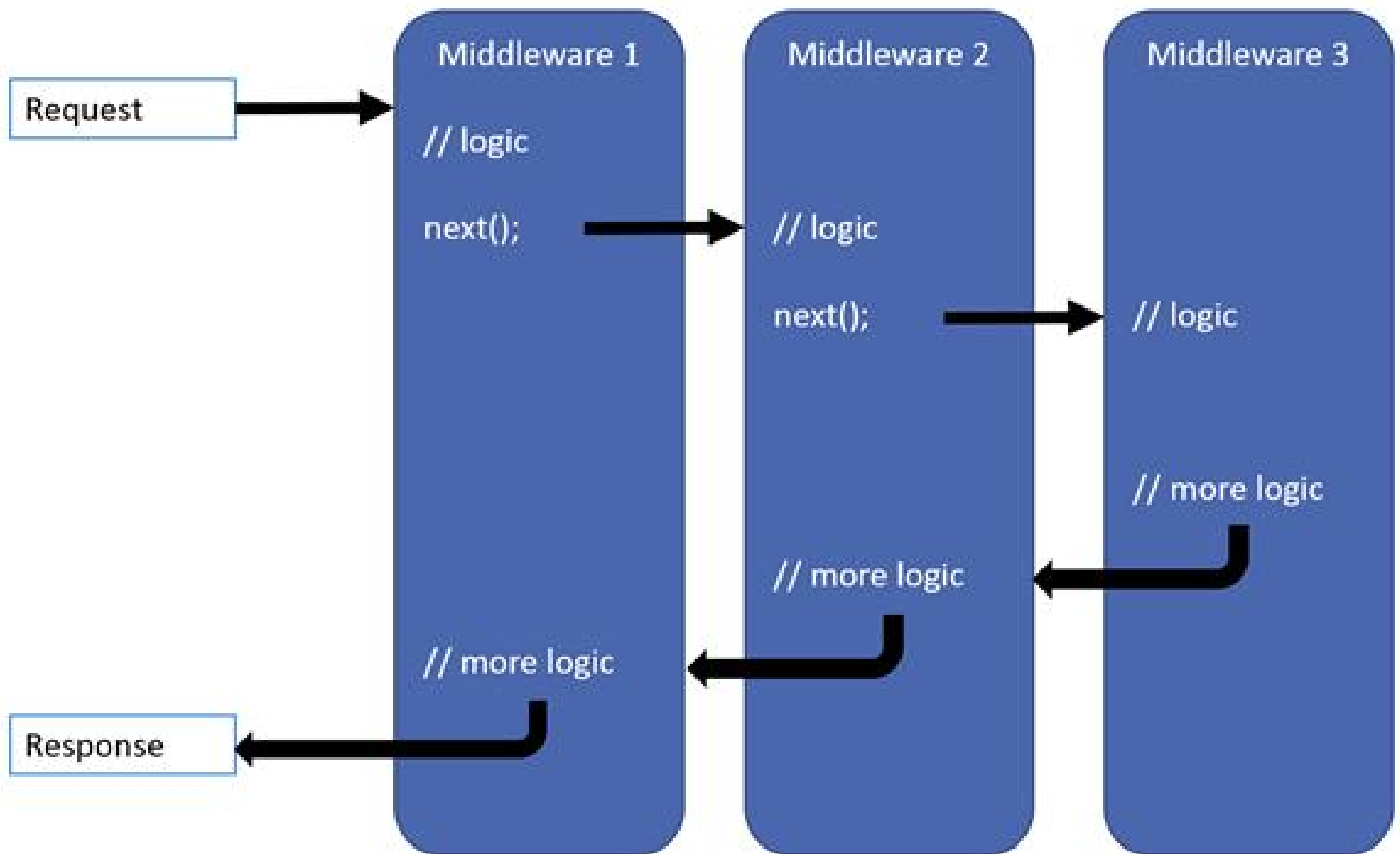
	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

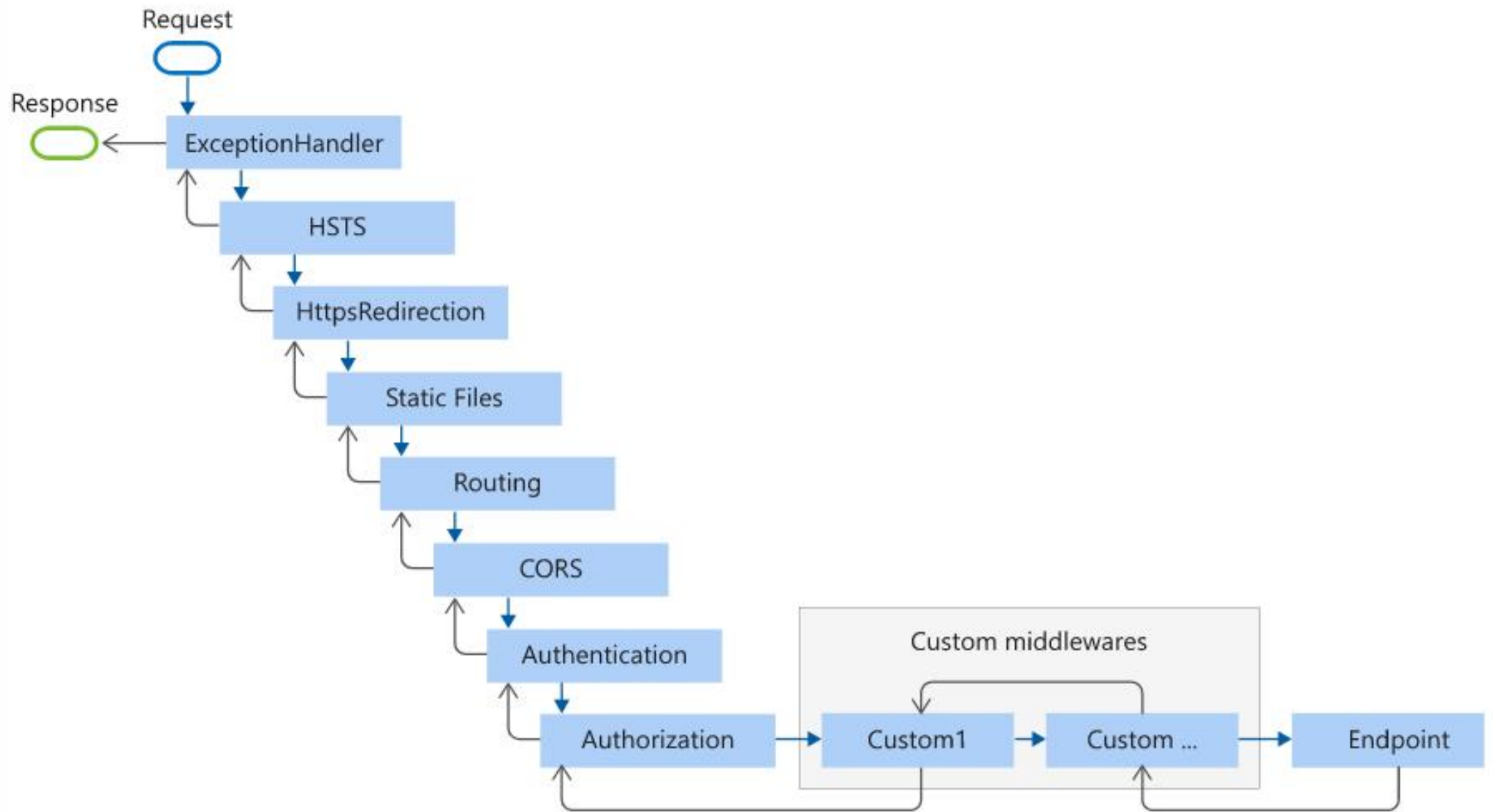
BodyCookiesHeaders (4)Test Results200 OK15 ms196 BSave Response ▼

PrettyRawPreviewVisualizeJSON ▼

```
1  [
2    {
3      "id": 33,
4      "name": "test-new",
5      "isComplete": false
6    }
7  ]
```

Writing Custom Middleware





Service lifetimes

See [Service lifetimes](#) in [Dependency injection in .NET](#)

To use scoped services in middleware, use one of the following approaches:

- Inject the service into the middleware's `Invoke` or `InvokeAsync` method. Using [constructor injection](#) throws a runtime exception because it forces the scoped service to behave like a singleton. The sample in the [Lifetime and registration options](#) section demonstrates the `InvokeAsync` approach.
- Use [Factory-based middleware](#). Middleware registered using this approach is activated per client request (connection), which allows scoped services to be injected into the middleware's constructor.

```
public class MyMiddleware
{
    private readonly RequestDelegate _next;
    private readonly ILogger          _logger;
    private readonly IGuidSingleton   _mySingletonService;
    0 references
    public MyMiddleware(RequestDelegate next,
                        ILogger<MyMiddleware> logger,
                        IGuidSingleton mySingletonService)
    {
        _logger = logger;
        _mySingletonService = mySingletonService;
        _next = next;
    }
    0 references
    public async Task InvokeAsync(HttpContext context,
                                IGuidScoped myScopedService,
                                IGuidTransient myTransientService)
    {
        _logger.LogInformation("Transient: " + myTransientService.GetGuid());
        _logger.LogInformation("Scoped:    " + myScopedService.GetGuid());
        _logger.LogInformation("Singleton: " + _mySingletonService.GetGuid());

        await _next(context);
    }
}
```

```
using ToDoApi.Middlewares;

namespace ToDoApi.Helper
{
    0 references
    public static class MyMiddlewareExtensions
    {
        0 references
        public static IApplicationBuilder UseMyMiddleware(this IApplicationBuilder builder)
        {
            return builder.UseMiddleware<MyMiddleware>();
        }
    }
}
```



```
var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseDefaultFiles();
app.UseStaticFiles();

app.UseHttpsRedirection();
app.UseAuthentication();
app.UseAuthorization();

app.UseMiddleware<MyMiddleware>();

app.MapControllers();

app.Run();
```


SvcLifeTime



GET

/api/SvcLifeTime



Parameters

Cancel

No parameters

Execute

Server response

Code	Details
------	---------

200	
-----	--

Response body

```
Transient 1 : 2935804a-3675-444e-b14e-e747c7223c77
Transient 2 : c05aa6b3-4f65-49f6-bacd-a467f526c68a
Scoped 1 : f5f0f502-5f2b-45c3-86e1-e13f74d27e0c
Scoped 2 : f5f0f502-5f2b-45c3-86e1-e13f74d27e0c
Singleton 1 : a124f312-e33d-489a-8b7f-ab08c0e735da
Singleton 2 : a124f312-e33d-489a-8b7f-ab08c0e735da
```

[Download](#)

Response headers

```
content-type: text/plain; charset=utf-8
date: Wed, 15 Feb 2023 19:49:45 GMT
server: Kestrel
x-firefox-spdy: h2
```

```
info: ToDoApi.Middlewares.MyMiddleware[0]
      Transient: 00e20548-1e0f-4ac5-9dd9-567459dfe7f3
info: ToDoApi.Middlewares.MyMiddleware[0]
      Scoped:    f5f0f502-5f2b-45c3-86e1-e13f74d27e0c
info: ToDoApi.Middlewares.MyMiddleware[0]
      Singleton: a124f312-e33d-489a-8b7f-ab08c0e735da
```

SvcLifeTime



GET

/api/SvcLifeTime



Parameters

Cancel

No parameters

Execute

Clear

Server response

Code

Details

200

Response body

```
Transient 1 : ecb842c9-abc5-47f8-bfa7-17fa14182e7e
Transient 2 : ed2172b5-241c-4e59-be6c-9251c21f701e
Scoped 1 : a686f1e6-f966-46b9-b92b-cfe3654d24ba
Scoped 2 : a686f1e6-f966-46b9-b92b-cfe3654d24ba
Singleton 1 : a124f312-e33d-489a-8b7f-ab08c0e735da
Singleton 2 : a124f312-e33d-489a-8b7f-ab08c0e735da
```



Download

Response headers

```
content-type: text/plain; charset=utf-8
date: Wed, 15 Feb 2023 19:51:22 GMT
server: Kestrel
x-firefox-spdy: h2
```

```
info: ToDoApi.Middlewares.MyMiddleware[0]
      Transient: 00e20548-1e0f-4ac5-9dd9-567459dfe7f3
info: ToDoApi.Middlewares.MyMiddleware[0]
      Scoped:    f5f0f502-5f2b-45c3-86e1-e13f74d27e0c
info: ToDoApi.Middlewares.MyMiddleware[0]
      Singleton: a124f312-e33d-489a-8b7f-ab08c0e735da
info: ToDoApi.Middlewares.MyMiddleware[0]
      Transient: 43762bd9-7537-469e-9e87-36eac6e5ed76
info: ToDoApi.Middlewares.MyMiddleware[0]
      Scoped:    a686f1e6-f966-46b9-b92b-cfe3654d24ba
info: ToDoApi.Middlewares.MyMiddleware[0]
      Singleton: a124f312-e33d-489a-8b7f-ab08c0e735da
```