Artsofte

Применение АОП в .net

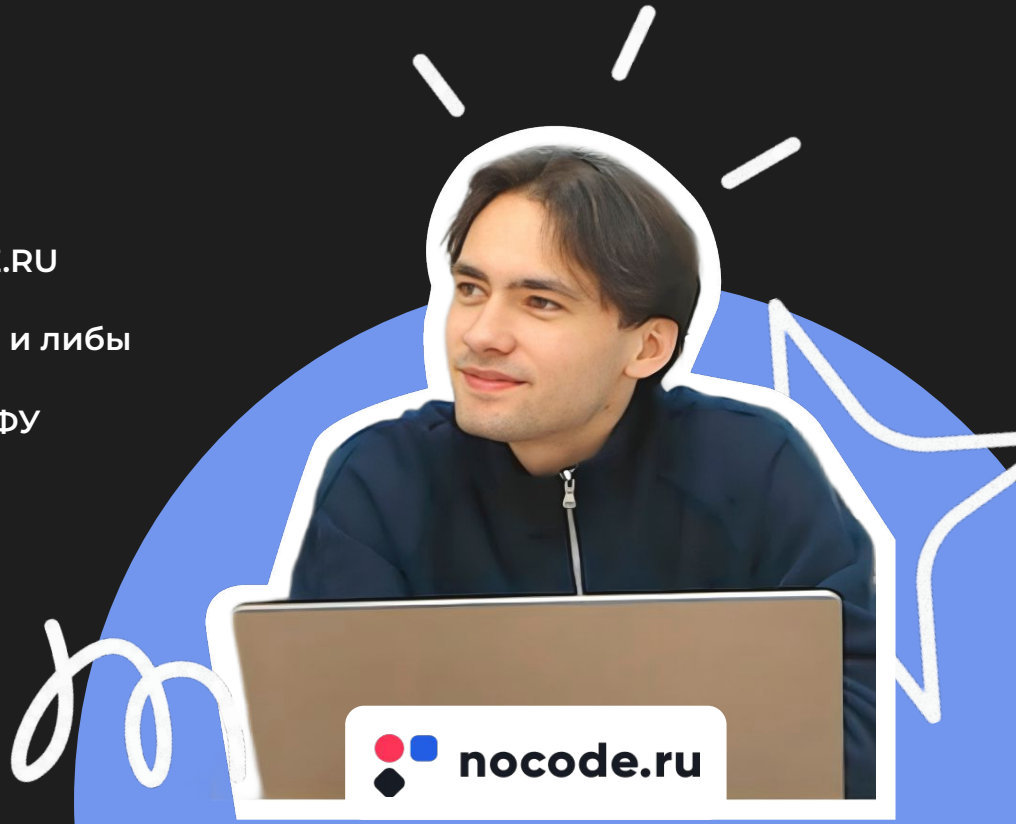
Немного рекламы себя

1. Работаю в компании Artsofte
2. Занимаюсь разработкой продукта NOCODE.RU
3. Разрабатываю инфраструктурные сервисы и либы
4. Читаю лекции по программированию в УрФУ

Подписывайтесь и ставьте лайки

@ArtsofteEducation

@dimoner1



Artsofte

Artsofte

Российская IT-компания с экспертизой разработки продуктов для цифровизации бизнеса

3000
Компаний используют решения Artsofte

16
городов, в которых есть наша команда

>20
лет на рынке IT

топ-20
поставщиков SaaS-решений в России

>20
стран используют продукты Artsofte

abanking
Платформа для построения экосистемы банка
FINTECH

nopaper
Сервис мобильного документооборота
ЭДО

nocode.ru
Zerocode-платформа для цифровой трансформации
ЦИФРОВИЗАЦИЯ

profit base
Цифровая экосистема для застройщика
НЕДВИЖИМОСТЬ

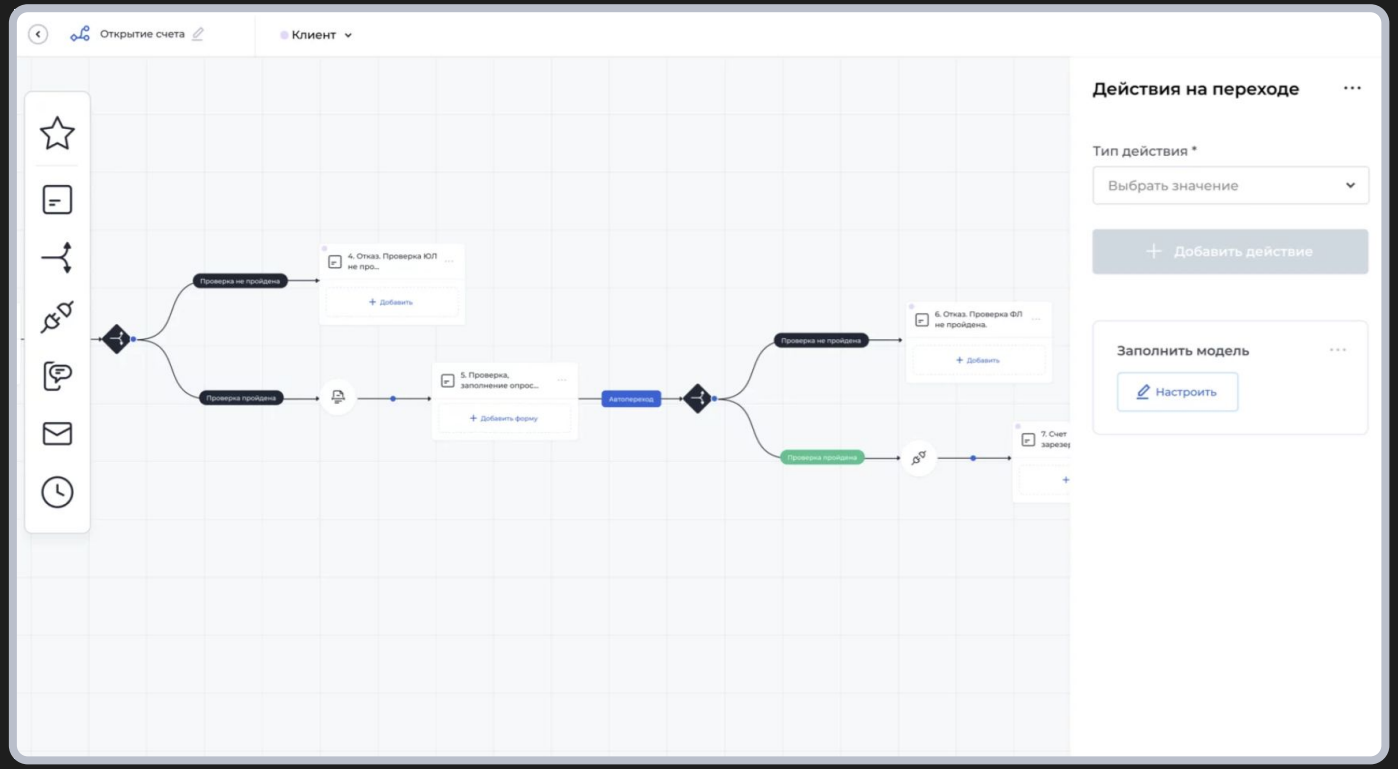
DIGITAL DEVELOPER
Исследовательский центр
PROPTECH

TradeDealer.ru
Платформа для автобизнеса
АВТО

Artsofte digital
Агентство интернет-маркетинга
МАРКЕТИНГ

Artsofte Consulting
Консалтинг по цифровой трансформации бизнеса
CONSULTING

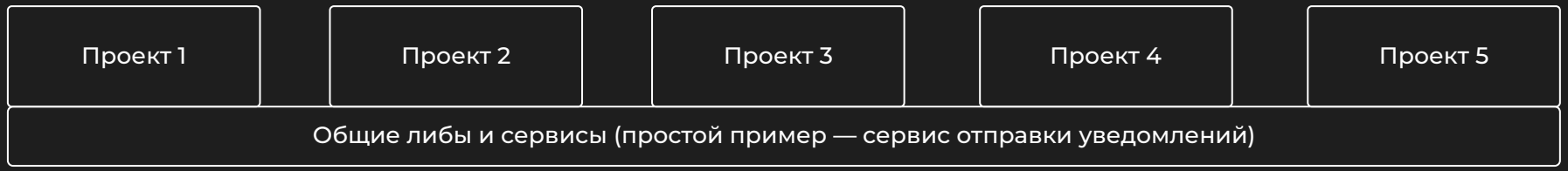
NOCODE



Artsofte

Как оно всё вместе?

Монорепозиторий



Команды в монорепозитории

Artsofte

Что в коде приводило к проблемам

Artsofte

Пример проблемного кода

Нужный технический код, который не влияет
на бизнес-логику, но захламляет всё

```
/// <inheritdoc />
0+1 usages
public async Task DeleteByTransitionIdListAsync(IEnumerable<Guid> transitionIdList)
{
    var logEnd :Action = LogHelper.StartLog(transitionIdList);

    if (transitionIdList is null || !transitionIdList.Any())
    {
        return;
    }

    var transaction = _actionRepository.BeginTransactionOrDefault();
    var searchFieldList = new SearchFieldModel[]
    {
        new(nameof(ActionDal.LinkEntityId), transitionIdList)
        {
            SearchFieldComparerType = SearchFieldComparerType.IN
        }
    };
    await _actionRepository.DeleteByFieldListAsync(searchFieldList, transaction);

    logEnd();
}
```


Artsofte

Примеры «проблемного» кода

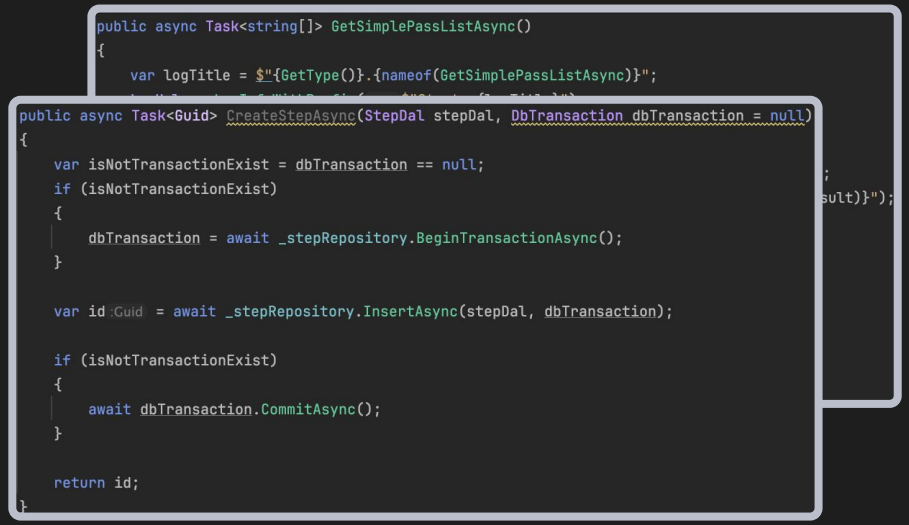
- Логирование
- Транзакции
- Вход и выход из multitenant
- Мониторинг

```
public async Task<string[]> GetSimplePassListAsync()
{
    var logTitle = $"{GetType()}.{nameof(GetSimplePassListAsync)}";
    LogHelper.LogInfoWithPrefix( text: $"Start: {logTitle}");
    try
    {
        var result :string[] = await _simplePasswordRepository.GetAllSimplePasswordAsync();
        LogHelper.LogInfoWithPrefix( text: $"End: {logTitle} Result: {JsonAb.Serialize(result)}");
        return result;
    }
    catch (Exception e)
    {
        LogHelper.LogInfoWithPrefix( text: $"End ERROR: {logTitle} Result: {e.Message}");
        throw;
    }
}
```


Artsofte

Примеры «проблемного» кода

- Логирование
- Транзакции
- Вход и выход из multitenant
- Мониторинг



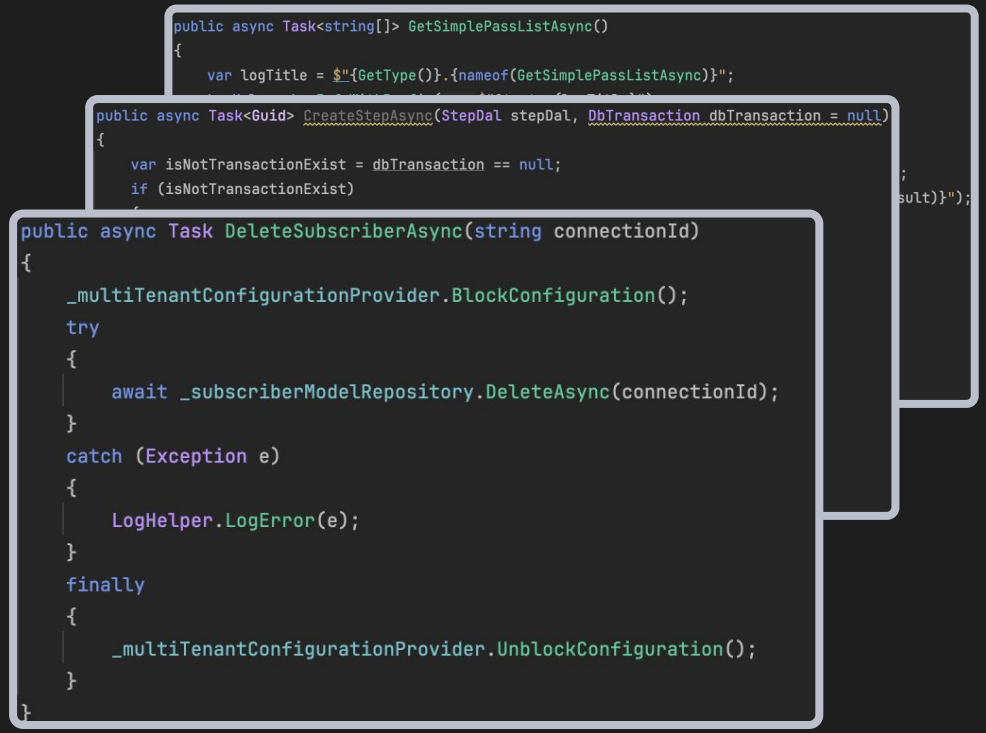
```
public async Task<string[]> GetSimplePassListAsync()
{
    var logTitle = $"{GetType()}.{nameof(GetSimplePassListAsync)}";
```

```
public async Task<Guid> CreateStepAsync(StepDal stepDal, DbTransaction dbTransaction = null)
{
    var isNotTransactionExist = dbTransaction == null;
    if (isNotTransactionExist)
    {
        dbTransaction = await _stepRepository.BeginTransactionAsync();
    }

    var id :Guid  = await _stepRepository.InsertAsync(stepDal, dbTransaction);

    if (isNotTransactionExist)
    {
        await dbTransaction.CommitAsync();
    }

    return id;
}
```


Artsofte

Примеры «проблемного» кода

- Логирование
- Транзакции
- Вход и выход из multitenant
- Мониторинг



```
public async Task<string[]> GetSimplePassListAsync()
{
    var logTitle = $"{GetType()}.{nameof(GetSimplePassListAsync)}";
```

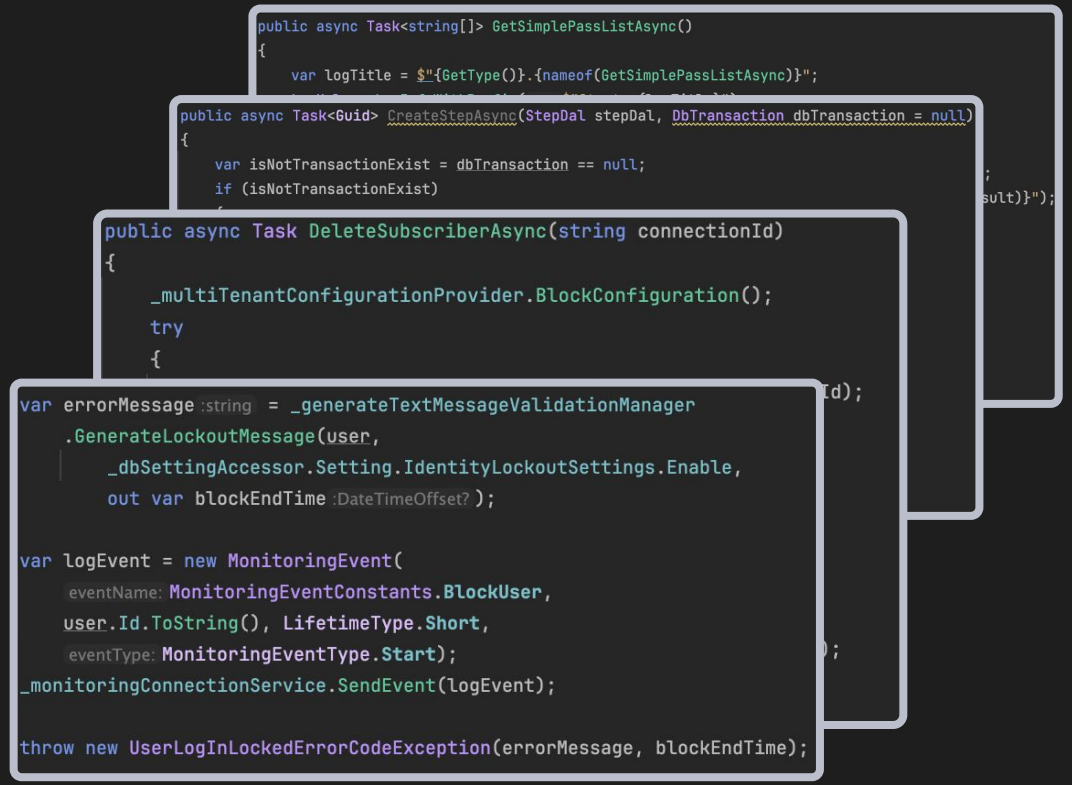
```
public async Task<Guid> CreateStepAsync(StepDal stepDal, DbTransaction dbTransaction = null)
{
    var isNotTransactionExist = dbTransaction == null;
    if (isNotTransactionExist)
```

```
public async Task DeleteSubscriberAsync(string connectionId)
{
    _multiTenantConfigurationProvider.BlockConfiguration();
    try
    {
        await _subscriberModelRepository.DeleteAsync(connectionId);
    }
    catch (Exception e)
    {
        LogHelper.LogError(e);
    }
    finally
    {
        _multiTenantConfigurationProvider.UnblockConfiguration();
    }
}
```


Artsofte

Примеры «проблемного» кода

- Логирование
- Транзакции
- Вход и выход из multitenant
- Мониторинг



Artsofte

Декомпозирование группы классов

Если заголовок есть, то работа метода должна трекаться
Нужно делать что-то до и после работы метода

Artsofte

Декорирование группы классов
(через ООП)

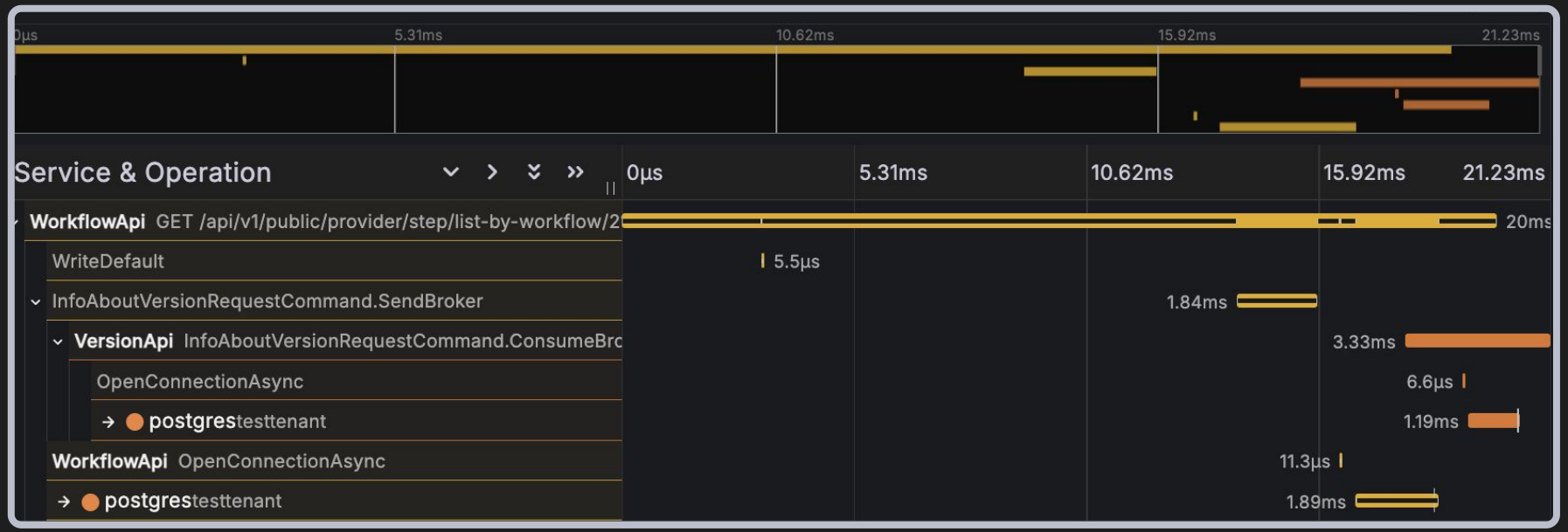
Много повторяющегося кода, надо для каждого класса репозитория писать декоратор, а потом делать сложные регистрации

```
services.TryAddScoped<StepRepository>();
services.TryAddScoped<IStepRepository>( implementationFactory: service =>
    {
        var repository = service.GetRequiredService<IStepRepository>();

        // если мы не передали заголовок версии, то выходим
        var isNotVersionAction = service.CheckIsVersionAction();
        if (isNotVersionAction)
        {
            return repository;
        }

        return new DecorateRepository(repository);
    });
```

```
public interface IStepRepository;

6 usages  new *
public class StepRepository : IStepRepository;

new *
public class DecorateRepository : IStepRepository
{
    private readonly IStepRepository _stepRepository;

    new *
    public DecorateRepository(IStepRepository stepRepository)
    {
        _stepRepository = stepRepository;
    }
}
```

Использование для добавления трейсов

Типичным решением является опять же интеграция
в работу Controller, например, через Filter и Item



Artsofte

Использование для добавления трейсов

Минусы

- Код повторяется
- Мы опять пишем лишний код

```
private async Task OpenConnectionAsync(DbConnection dbConnection)
{
    var activityStatusCode = ActivityStatusCode.Ok;
    var activivty :Activity = _serviceDiagnostic.StartActivity(nameof(OpenConnectionAsync), ActivityKind.Internal);
    try
    {
        await dbConnection.OpenAsync();
    }
    catch
    {
        activityStatusCode = ActivityStatusCode.Error;
        throw;
    }
    finally
    {
        activivty?.SetStatus(activityStatusCode);
        activivty?.Stop();
    }
}
```

Использование для добавления трейсов

Минусы

- Аллокация на func
- Может быть замыкание
- Лишний код

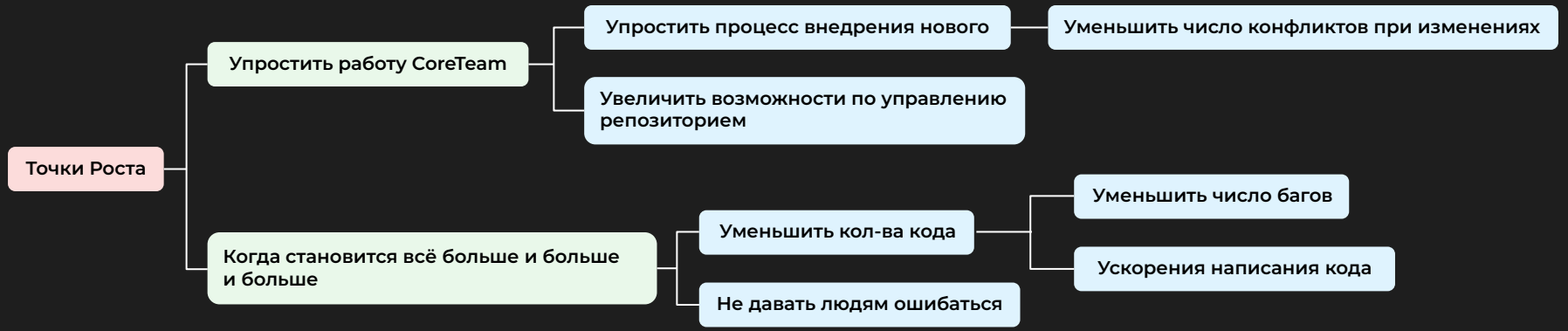
```
private async Task OpenConnectionAsync(DbConnection dbConnection)
{
    await _serviceDiagnostic.StartActivity( func: async () =>
    {
        await dbConnection.OpenAsync();

    }, nameof(OpenConnectionAsync), ActivityKind.Internal); // Task
}
```

Управление инфраструктурой из CoreLib

- Переход с Start на StartAsync CoreTeam 100500 конфликтов

```
private async Task<List<TableDependency>> GetTableSqlHierarchyAsync(IDbConnecti
{
    var connection = await connectionFactory.CreateConnectionAsync();
    var tableHierarchy :IEnumerable<TableDependency> = await connection.GetTableHier

    var tableSqlHierarchy :List<TableDependency> = tableHierarchy.Where(x :TableDepen

    return tableSqlHierarchy;
}
```

Задачи, которые вытекают из этого кода



Artsofte

А как делают другие?

Использование декораторов в Angular

```
@Component({
    selector: 'sado-cl-cabinet-menu',
    templateUrl: './cabinet-menu.adaptive.component.html',
    styleUrls: ['./styles/cabinet-menu.master.adaptive.scss'],
    changeDetection: ChangeDetectionStrategy.OnPush,
})
export class CabinetMenuAdaptiveComponent extends CabinetMenuBaseComponent implements AfterViewInit {
    @Output()
    public closeMenuEmitter: EventEmitter<void> = new EventEmitter;

    @ViewChild('container')
    public containerElement!: ElementRef<HTMLDivElement>;

    public isDefaultLogo: boolean = false;
```

\+ Сразу видишь всё, что относится к поведению компоненты

* Аналогичный опыт можно почерпнуть из UI/UX, плохо, когда пользователю надо что-то запоминать между страницами

Artsofte

Использование декораторов в Spring

Пример использование
декораторов/атрибутов в backend

```
@Service
public class InvoiceService {

    @Transactional(propagation = Propagation.REQUIRES_NEW)
    public void createPdf() {
        // ...
    }
}
```

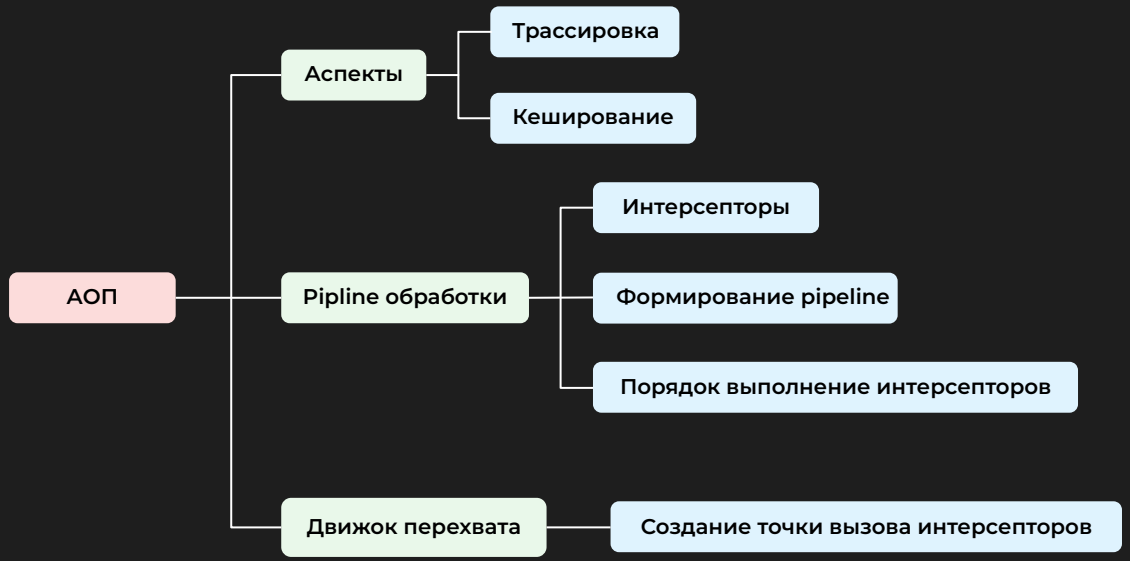
АОП

АОП — аспектно-ориентированное программирование


```
public async Task<string[]> GetSimplePassListAsync()
{
    var logTitle = $"{GetType()}.{nameof(GetSimplePassListAsync)}";
    LogHelper.LogInfoWithPrefix( text: $"Start: {logTitle}");
    try
    {
        var result :string[] = await _simplePasswordRepository.GetAllSimplePasswordAsync();
        LogHelper.LogInfoWithPrefix( text: $"End: {logTitle} Result: {JsonAb.Serialize(result)}");
        return result;
    }
    catch (Exception e)
    {
        LogHelper.LogInfoWithPrefix( text: $"End ERROR: {logTitle} Result: {e.Message}");
        throw;
    }
}
```


Artsofte

Из чего состоит АОП и его реализация



Artsofte

Применение

```
[Tracing]
0+3 usages   D Egorov +1 *
public async Task<string[]> GetSimplePassListAsync()
{
    var result :string[] = await _simplePasswordRepository
        .GetAllSimplePasswordAsync(); // Task<string[]>
    return result;
}
```

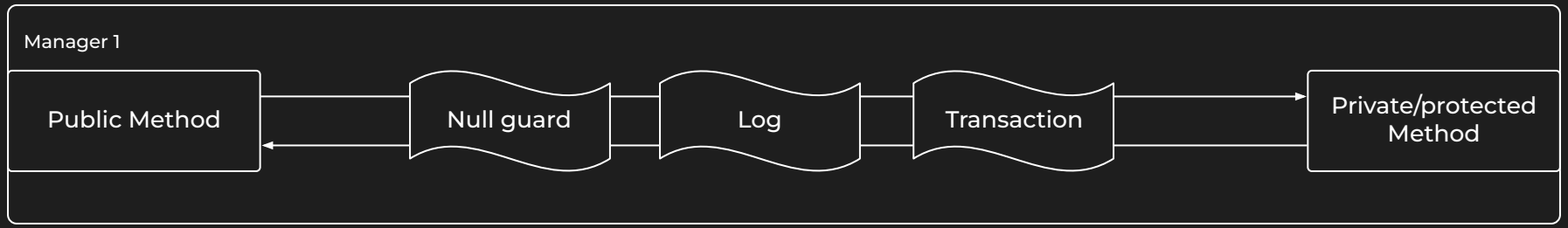
Бизнес-код

```
public void InterceptSynchronousHandlerVoid(IInvocation invocation)
{
    var activity = _serviceDiagnostic.StartActivity(invocation.Method.Name, ActivityKind.Internal);
    var activityStatusCode = ActivityStatusCode.Ok;
    try
    {
        invocation.Proceed();
    }
    catch
    {
        activityStatusCode = ActivityStatusCode.Error;
        throw;
    }
    finally
    {
        activity?.SetStatus(activityStatusCode);
        activity?.Stop();
    }
}
```

Сервис, инкапсулирующий логику

Типы вызова методов

Между классами



Между методами в одном классе

Artsofte

АОП в .NET - Filter Controller

Плюсы

- Определяем атрибут
- По атрибуту как маркеру вызывается сервис
- Логика сервиса легко подключается и инкапсулируется

```
public class TestController : Controller
{
    [TestResourceFilter]
    new *
    public IActionResult Test()
    {
        return Ok();
    }
}
```

```
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Policy = RequireClientPolice.Name)]
public class ChatPublicController : BasePublicController
```

АОП в .NET - Diagnostic System

Пример перехвата начала и конца логики

```
private void StopActivity(Activity activity, HttpContext httpContext)
{
  if (activity.Duration == TimeSpan.Zero)
    activity.SetEndTime(DateTime.UtcNow);
  HostingApplicationDiagnostics.WriteDiagnosticEvent<HttpContext>((DiagnosticSource) this._diagnosticListener, name: "Microsoft.AspNetCore.Hosting.HttpRequestIn.Stop", httpContext);
  activity.Stop();
}
```

```
public void OnNext(KeyValuePair<string, object> value)
{
    if (value.Key == "Microsoft.AspNetCore.Hosting.HttpRequestIn.Start")
    {
        Console.WriteLine(value.Key);
        Console.WriteLine(value.Value);
        Console.WriteLine();
    }

    if (value.Key == "Microsoft.AspNetCore.Hosting.HttpRequestIn.Stop")
    {
        Console.WriteLine(value.Key);
        Console.WriteLine(value.Value);
        Console.WriteLine();
    }
}
```

АОП в .NET - Mock

Плюсы

- \+ Быстро и удобно описывает методы
- \+ Не нужно делать декоратор под каждый класс!!!

```
IL code
private void InitializeInstance()
{
    // Determine the set of interfaces that the proxy object should additionally implement.
    var additionalInterfaceCount :int = this.AdditionalInterfaces.Count;
    var interfaces = new Type[1 + additionalInterfaceCount];
    interfaces[0] = typeof(IMocked<T>);
    this.AdditionalInterfaces.CopyTo( index: 0, interfaces, arrayIndex: 1, additionalInterfaceCount);

    this.instance = (T)ProxyFactory.Instance.CreateProxy(
        typeof(T),
        interceptor: this,
        interfaces,
        this.constructorArguments); // object
}
```

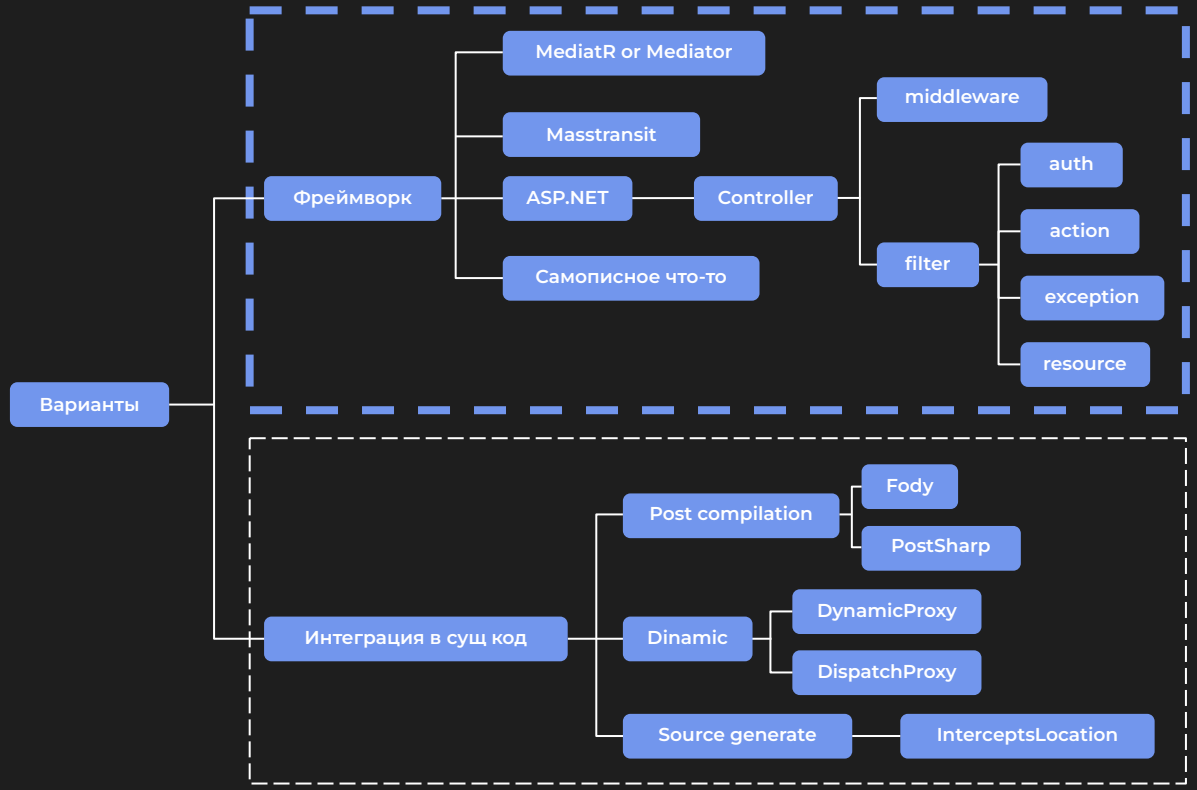
```
WorkflowApplicationFactory.ConnectMonitoringServiceMock // Mock<IMonitoringConnectionService>
    .Setup( expression: x :IMonitoringConnectionService => x.SendEvent(It.IsAny<MonitoringEvent>()));

WorkflowApplicationFactory.ConnectDocumentSchemaServiceMock // Mock<IConnectDocumentSchemaService>
    .Setup( expression: x :IConnectDocumentSchemaService => x.CreateDocumentSchemaAsync( request: It.IsAny<CreateDocumentSchemaBrokerRequest>()))
    .ReturnsAsync(new CreateDocumentSchemaBrokerResponse()
    {
        Id = ObjectId.GenerateNewId()
    });
```

Какими путями можно внедрять АОП в код

Движок для создания точек перехвата

Фреймворк

- \+ Легко реализовать
- \- Нужен Фреймворк

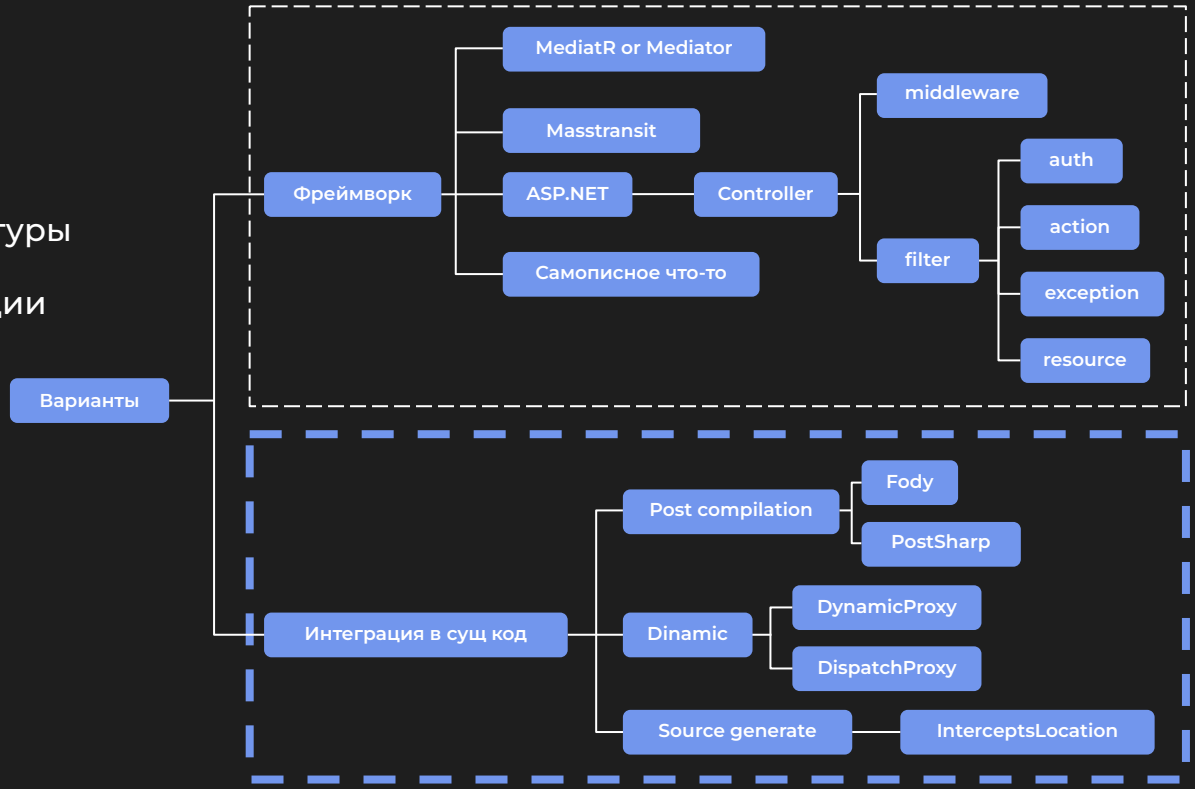


Artsofte

Движок для создания точек перехвата

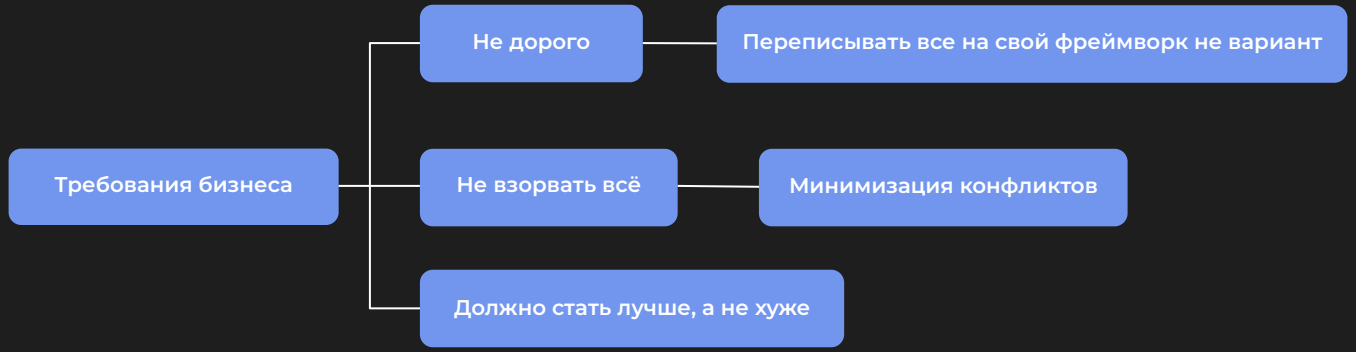
Интеграция в код

- \+ Подходит для любой архитектуры
- \- Много нюансов при реализации



Artsofte

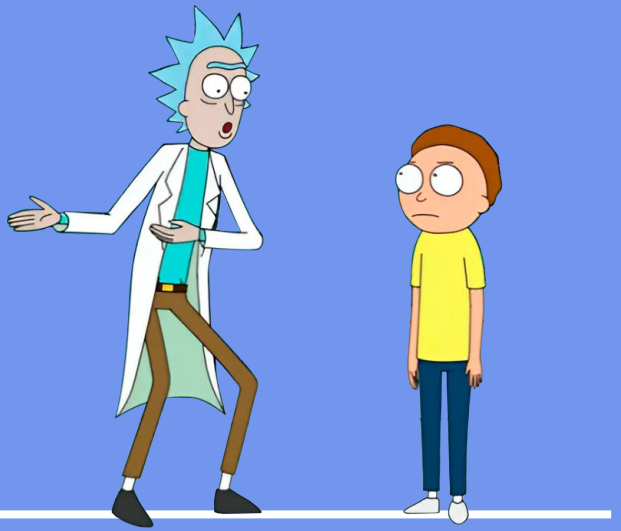
Требования бизнеса к улучшениям



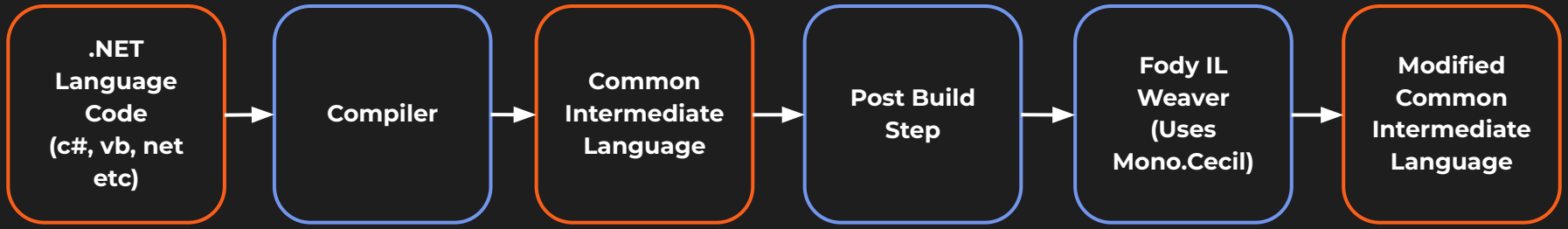
Технические требования

Artsofte

Post Compilation — Fody



Fody



Плюсы

- \+ Хорошая поддержка сообществом
- \+ Много решений
- \+ Перехват любого типа методов

Дополнительно мы использовали библиотеку MethodBoundaryAspect.Fody

Fody: пример

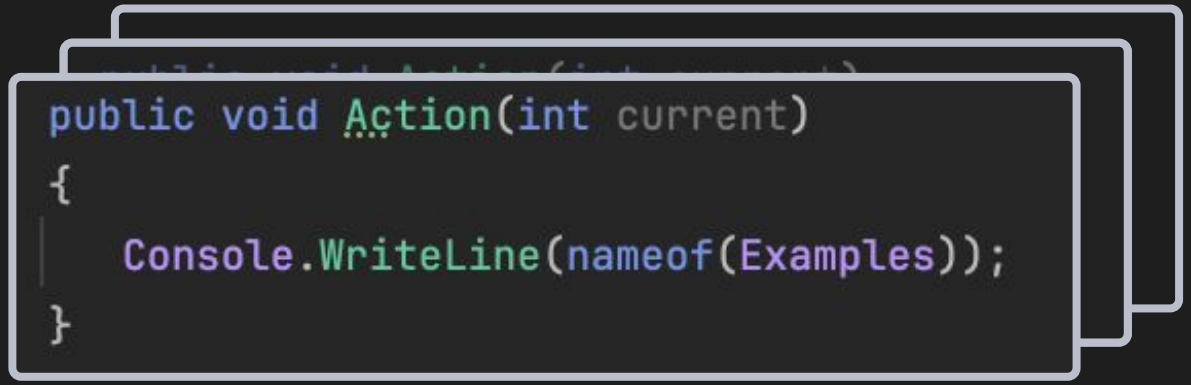
```
[LogExample]
3 usages
public class FodyExample
{
    2 usages
    public async Task Action(int current)
    {
        Console.WriteLine(HelperGetInt());
    }
    1 usage
    private int HelperGetInt() => 1;
```

```
OnEntry Action
OnEntry MoveNext
OnEntry HelperGetInt
OnExit HelperGetInt
1
OnExit MoveNext
OnExit Action
```

Fody: до и после

Было

5 классов на запрос,
каждый вызывал по 2 метода



Artsofte

Fody: до и после

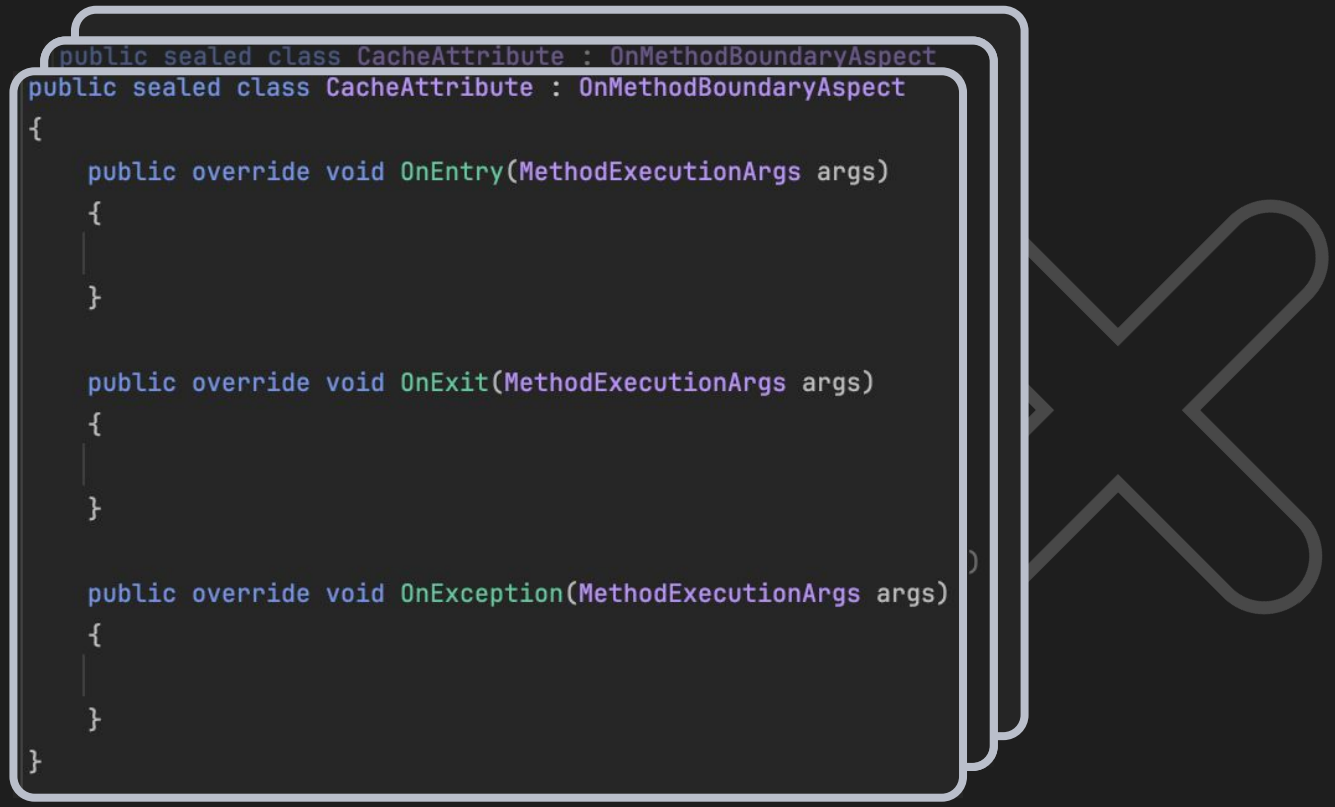
Стало

5 * 2 * 3 = 30 классов на запрос

```
public void Action(int current)
{
  object[] objArray = new object[1]{ (object) current };
  MethodExecutionArgs args = new MethodExecutionArgs();
  args.Arguments = objArray;
  MethodBase b01Ca1D2B354AbeC2B = MethodInfos._methodInfo_8D48D52327
  args.Method = b01Ca1D2B354AbeC2B;
  FodyExample fodyExample1 = this;
  args.Instance = (object) fodyExample1;
  CustomLogAttribute customLogAttribute = new CustomLogAttribute();
  customLogAttribute.OnEntry(args);
  if (args.FlowBehavior == FlowBehavior.Return)
    return;
  FodyExample fodyExample2 = this;
  try
  {
    fodyExample2.$_executor_Action(current);
  }
  catch (Exception ex)
  {
    args.Exception = ex;
    customLogAttribute.OnException(args);
    switch (args.FlowBehavior)
    {
      case FlowBehavior.Continue:
        return;
      case FlowBehavior.Return:
        return;
      default:
        throw;
    }
  }
  customLogAttribute.OnExit(args);
}
```


Artsofte

Сам перехватчик



1 способ: ContinueWith

Минусы

- Не подходит, если нужно ожидание
- Остается проблема с Task<T>

```
public override void OnExit(MethodExecutionArgs args)
{
    if (args.ReturnValue is Task t && !args.ReturnValue.GetType().IsGenericType)
    {
        t.ContinueWith( continuationAction: _ =>
        {
            Console.WriteLine($"OnExit {Thread.CurrentThread.ManagedThreadId}");
        }, TaskContinuationOptions.ExecuteSynchronously);
    }
    else
    {
        Console.WriteLine($"OnExit {Thread.CurrentThread.ManagedThreadId}");
    }
}
```

2 способ: TaskCompletionSource

Минусы

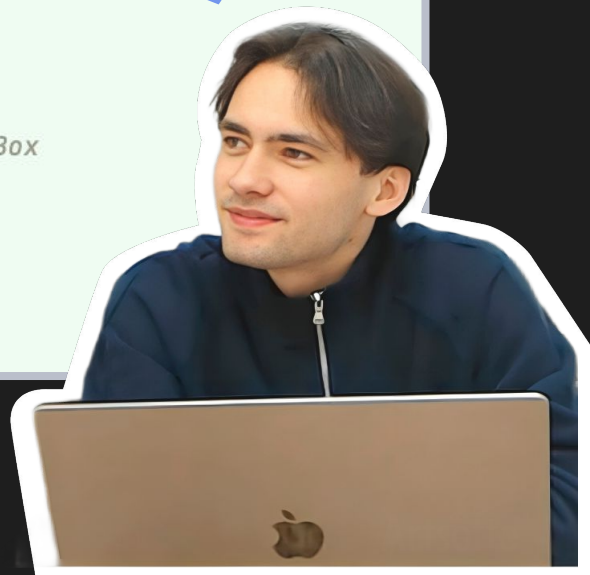
- Производительность
- Замыкание
- Если класс Singleton — будут проблемы с многопоточностью
- Остается проблема с Task<T>

можно решить через source generate

```
private readonly TaskCompletionSource _source = new TaskCompletionSource();

public override void OnExit(MethodExecutionArgs args)
{
    if (args.ReturnValue is Task t && !args.ReturnValue.GetType().IsGenericType)
    {
        args.ReturnValue = _source.Task;
        t.ContinueWith( continuationFunction: async task =>
        {
            await task;
            Console.WriteLine($"OnExit {Thread.CurrentThread.ManagedThreadId}");
            _source.SetResult();
        }, TaskContinuationOptions.ExecuteSynchronously);
    }
    else
    {
        Console.WriteLine($"OnExit {Thread.CurrentThread.ManagedThreadId}");
    }
}
```

```
if (t.IsCompletedSuccessfully)
{
    //System.Threading.Tasks.VoidTaskResult - объект, который возвращает Task.Result без результата
    var isResultTask = !genericArguments.First().FullName.Equals("System.Threading.Tasks.VoidTaskResult");

    if (isResultTask)
    {
        var returnValueTask = arg.ReturnValue as Task;
        var result = type.GetProperty("Result")?.GetValue(returnValueTask);
        arg.ReturnValue = result;
    }
    else
    {
        //Если таска не возвращает результат, она возвращает AsyncStateMachineBox
        //Смысла в этом, думаю, нету
        arg.ReturnValue = null;
    }
    HandleExit(arg);
}
```

Делать так **не надо**



3 способ: Dynamic

- Определяем, что это Task<T>
- Создаём delegate под вызов
- Логика создания delegate
- Мы попали в Task<T>
- Мы попали в Task<T> с ожиданием

```
private static bool IsGenericTask(Type returnType)
{
    return returnType != typeof(void)
           && typeof(Task).IsAssignableFrom(returnType)
           && returnType.GetTypeInfo().IsGenericType;
}
```

3 способ: Dynamic

- Определяем, что это Task<T>
- Создаём delegate под вызов
- Логика создания delegate
- Мы попали в Task<T>
- Мы попали в Task<T> с ожиданием

```
private static bool IsGenericTask(Type returnType)
{
    return returnType != typeof(void)
```

```
private static readonly ConcurrentDictionary<Type, GenericAsyncHandler> GenericAsyncHandlers = new();
1 usage
private static GenericAsyncHandler GetHandler(Type returnType)
{
    GenericAsyncHandler handler = GenericAsyncHandlers.GetOrAdd(returnType, CreateHandler);
    return handler;
}
```


Artsofte

3 способ: Dynamic

- Определяем, что это Task<T>
- Создаём delegate под вызов
- Логика создания delegate
- Мы попали в Task<T>
- Мы попали в Task<T> с ожиданием

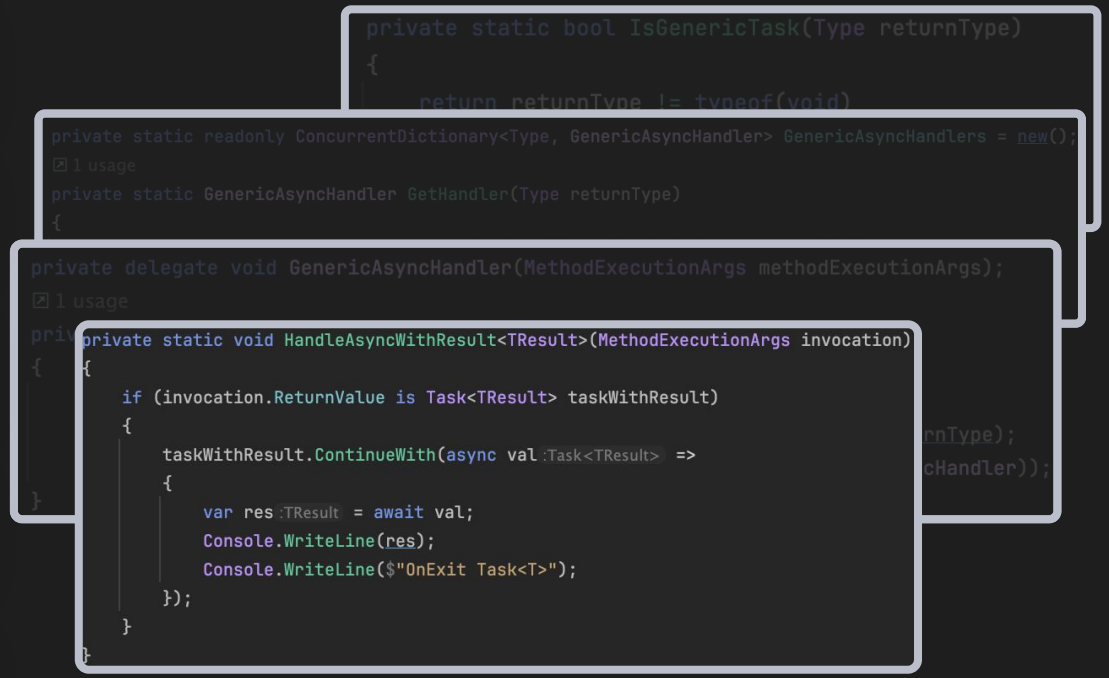
```
private static bool IsGenericTask(Type returnType)
{
    return returnType != typeof(void)
```

```
private static readonly ConcurrentDictionary<Type, GenericAsyncHandler> GenericAsyncHandlers = new();
1 usage
private static GenericAsyncHandler GetHandler(Type returnType)
{
```

```
private delegate void GenericAsyncHandler(MethodExecutionArgs methodExecutionArgs);
1 usage
private static GenericAsyncHandler CreateHandler(Type returnType)
{
    Type taskReturnType = returnType.GetGenericArguments()[0];
    MethodInfo methodInfo = HandleAsyncMethodInfo.MakeGenericMethod(taskReturnType);
    return (GenericAsyncHandler)methodInfo.CreateDelegate(typeof(GenericAsyncHandler));
}
```

3 способ: Dynamic

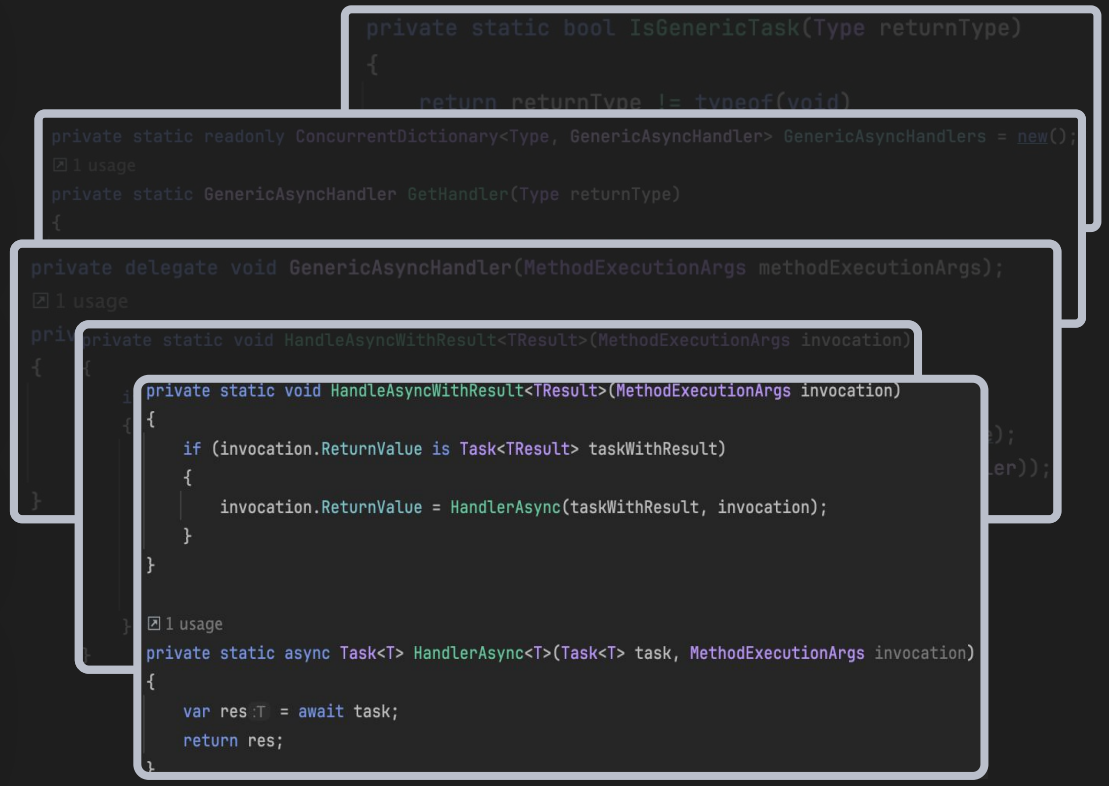
- Определяем, что это Task<T>
- Создаём delegate под вызов
- Логика создания delegate
- Мы попали в Task<T>
- Мы попали в Task<T> с ожиданием



Artsofte

3 способ: Dynamic

- Определяем, что это Task<T>
- Создаём delegate под вызов
- Логика создания delegate
- Мы попали в Task<T>
- **Мы попали в Task<T> с ожиданием**



Artsofte

Формирование pipeline

Artsofte

А как делает MediatR?
Черпаем вдохновение

```
public override Task<TResponse> Handle(
    IRequest<TResponse> request,
    IServiceProvider serviceProvider,
    CancellationToken cancellationToken)
{
    // все эти вычисления каждый раз
    return serviceProvider
        .GetServices<IPipelineBehavior<TRequest, TResponse>>() // находим все обработчики между вызовами
        .Reverse() // определение порядке через порядок регистрации !!!
        .Aggregate( // ЗАМЫКАНИЕ
            () => serviceProvider
                .GetRequiredService<IRequestHandler<TRequest, TResponse>>() // ПЛЮС что есть DI
                .Handle((TRequest) request, cancellationToken),
            (Func<RequestHandlerDelegate<TResponse>, IPipelineBehavior<TRequest, TResponse>, RequestHandlerDelegate<TResponse>>)
            ((next :RequestHandlerDelegate<TResponse> , pipeline) => (RequestHandlerDelegate<TResponse>) (() => pipeline.Handle((TRequest) request, next, cancellationToken))))();
}
```

1. Аллокации/замыкания
2. Каждый раз пайплайн в рантайме

Хотелось бы не создавать pipeline каждый раз (ASP)

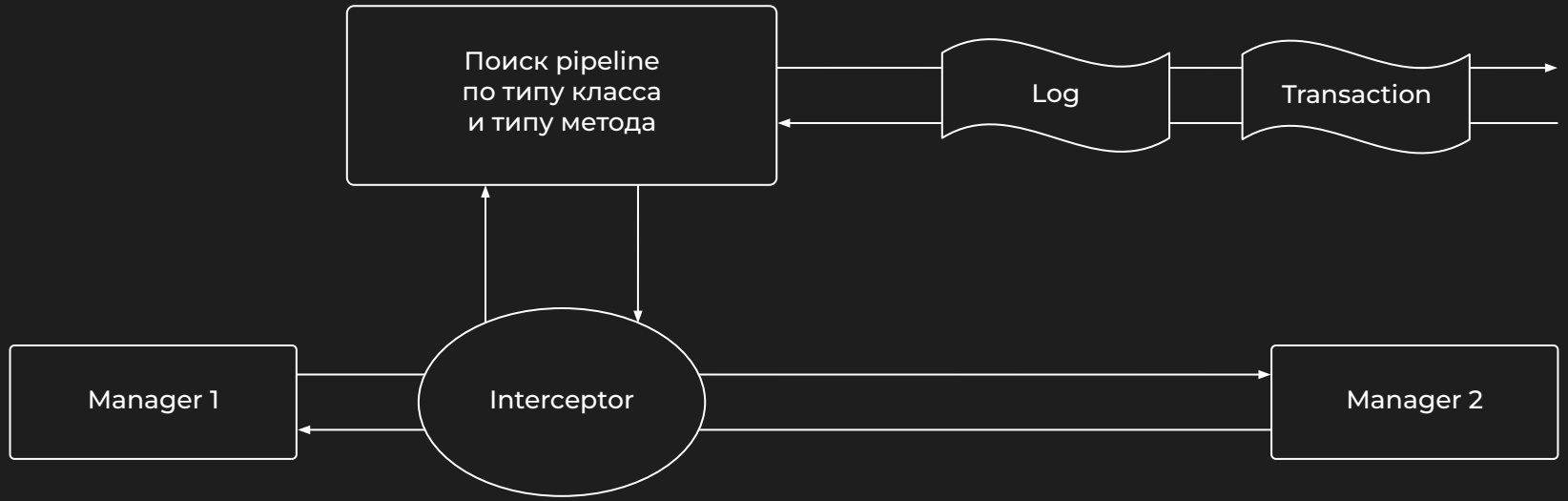
```
public RequestDelegate Build()
{
  RequestDelegate requestDelegate = (RequestDelegate) (context =>
  {
    Endpoint endpoint = context.GetEndpoint();
    if (!context.Response.HasStarted)
      context.Response.StatusCode = 404;
    context.Items[(object) "__RequestUnhandled"] = (object) true;
    return Task.CompletedTask;
  });
  for (int index = this._components.Count - 1; index >= 0; --index)
    requestDelegate = this._components[index](requestDelegate);
  return requestDelegate;
}
```

```
public class Middleware
{
    private readonly RequestDelegate next;

    public Middleware(RequestDelegate next)
    {
        this.next = next;
    }

    public async Task InvokeAsync(HttpContext context)
    {
        var timer = context.RequestServices.GetRequiredService<ITimer>();
        await next.Invoke(context);
    }
}
```

1. ASP.NET: один раз при Build формируем pipeline
2. Как следствие, DI передаётся через параметры

Но у него один конвейер, а как сделать несколько?

Pipeline + interceptor chain

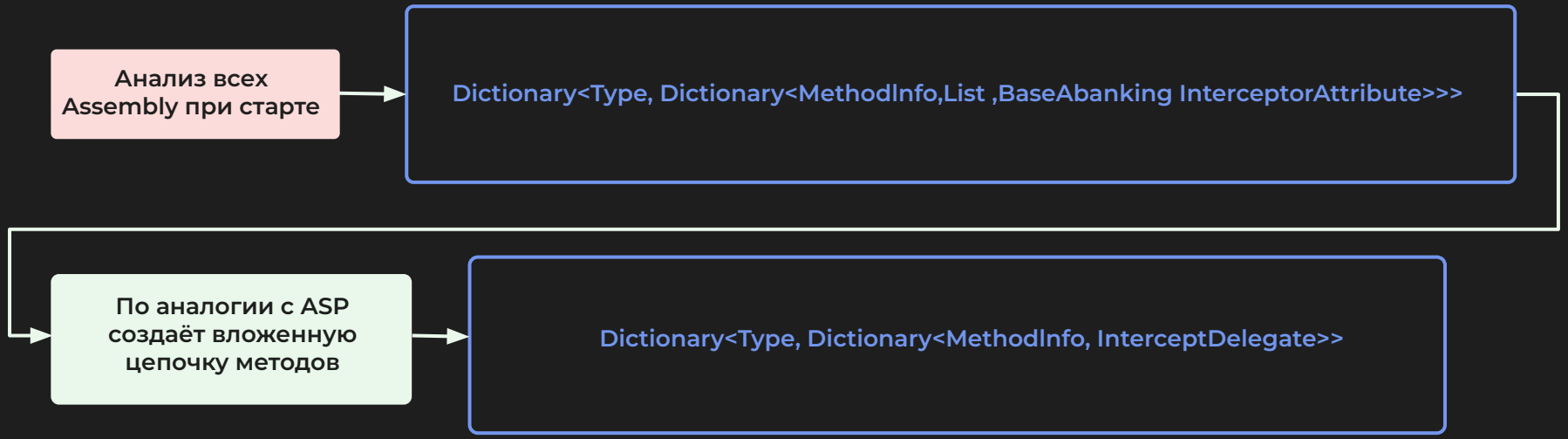


Artsofte

Вид интеграции

```
public class Test1Interceptor : IAbankingInterseptor
{
    private readonly InterceptTask _interceptTask;
    private readonly InterceptVoid _interceptVoid;

    public Test1Interceptor() { }
    public Test1Interceptor(InterceptVoid interceptVoid) { _interceptVoid = interceptVoid; }
    public Test1Interceptor(InterceptTask interceptTask) { _interceptTask = interceptTask; }

    public void InterceptSynchronousHandlerVoid(MethodExecutionArgs arg) { _interceptVoid(arg); }
    public async Task InterceptSynchronousHandler(MethodExecutionArgs arg) { await _interceptTask(arg);}

    0+1 usages
    public async Task InterceptSynchronousHandlerRes<TResult>(MethodExecutionArgs arg)
    {
        await _interceptTask(arg);
        if (arg.ReturnValue is Task<TResult> returnValue)
        {
            var resData :TResult = await returnValue;
            Console.WriteLine(resData);
        }
    }
}
```

Реализация через runtime



Artsofte

Реализация (Как middleware)

```
if (typeof(Task).IsAssignableFrom(methodType.ReturnType) && methodType.ReturnType.GetTypeInfo().IsGenericType)
{
    var zero = (InterceptTask)(static async arg :MethodExecutionArgs =>
    {
        if (arg.ReturnValue is Task task)
        {
            await task;
        }
    });

    foreach (var interceptor :BaseAbankingInterseptorAttribute? in interseptorAttributeList)
    {
        // тут кеш
        var interceptorInstance :object? = Activator.CreateInstance(interceptor.InterseptorType, zero);

        zero = (InterceptTask)interceptor
            .InterseptorType?
            .GetMethod(nameof(ICustomInterseptor.InterceptSynchronousHandlerRes))!
            .MakeGenericMethod(methodType.ReturnType.GetGenericArguments()[0])
            .CreateDelegate(typeof(InterceptTask), interceptorInstance)!;
    }

    dictionary.Add(methodType, zero);
}
```

Вызов внутри Fody

```
private static async Task<T> HandlerAsync<T>(Task<T> task, MethodExecutionArgs invocation)
{
    if (InterceptPipelineCreator.Dictionary.TryGetValue(
            invocation.Method.DeclaringType,
            out var methodList :Dictionary<MethodInfo,InterceptTask>? ))
    {
        if (methodList.TryGetValue((MethodInfo)invocation.Method, out var intercept))
        {
            await intercept(invocation);
        }
        else
        {
            return await task;
        }
    }
    else
    {
        return await task;
    }

    if (task.IsCompleted)
    {
        return task.Result;
    }

    throw new Exception();
}
```

Стек трейс

- DebuggerStepThroughAttribute отдаёт отладчику указание о сквозной обработке кода (вместо обработки изнутри)

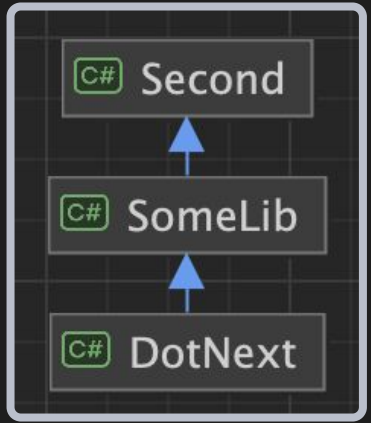
```
[DebuggerStepThrough]
0+1 usages  degorov +1 *
protected override async Task<TResult> InterceptSynchronousHandler<TResult>(Func<Task<TResult>> next, IInvocation invocation)
{
    var actionResult :Func<Task<...>> = next;
    var interseptor :List<IAbankingInterseptor> = GetCurrentForMethodList(invocation);

    for (var c :int = interseptor.Count - 1; c >= 0; c--)
    {
        actionResult = interseptor[c].InterceptSynchronousHandler(actionResult, invocation);
    }

    if (actionResult == null)
    {
        var res :TResult = await next();
        return res;
    }

    return await actionResult();
}
```


Artsofte

А почему мой класс не интерсептируется?

- Assembly Lazy, соответственно, если на момент построения дерева не было обращения к типу, то сборка просто не подгрузится!

```
foreach (var name in Assembly.GetEntryAssembly()?.GetReferencedAssemblies()!)
{
    var res :Assembly = Assembly.Load(name);
    foreach (var nameInner in res.GetReferencedAssemblies()!)
    {
        Assembly.Load(nameInner);
    }
}
```



Просто добавили это в CoreLib

Fody DI

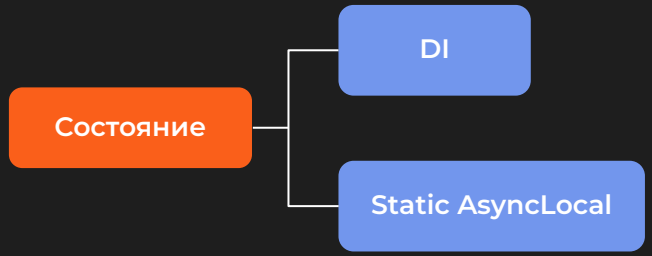
Artsofte

Как получить DI/внешнее состояние?

1. Чтобы работать с DI, нужно получить доступ к IServiceProvider

2. Делать это через Fody мы не можем, так как сам обработчик создается через new

```
public void Action(int current)
{
  object[] objArray = new object[1]{ (object) current };
  MethodExecutionArgs args = new MethodExecutionArgs();
  args.Arguments = objArray;
  MethodBase b01Ca1D2B354AbeC2B = MethodInfos._methodInfo_8D48D52327
  args.Method = b01Ca1D2B354AbeC2B;
  FodyExample fodyExample1 = this;
  args.Instance = (object) fodyExample1;
  CustomLogAttribute customLogAttribute = new CustomLogAttribute();
```



Artsofte

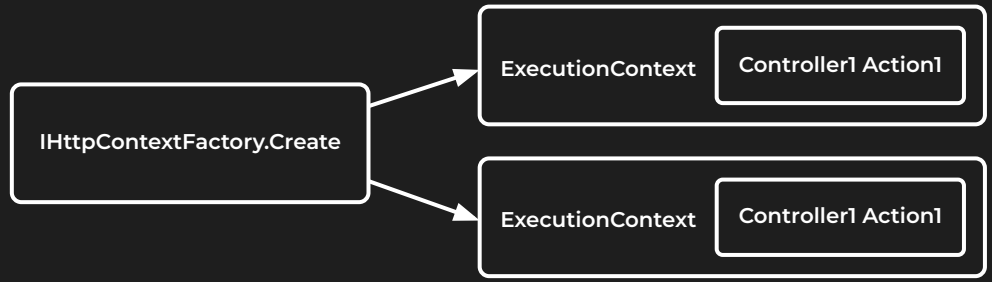
Пример решения проблемы из ASP

! Сам ASP.NET позволяет получать доступ из статики к текущему контексту запроса:

```
public class HttpContextAccessor : IHttpContextAccessor
{
  #nullable disable
  IL code
  private static readonly AsyncLocal<HttpContextAccessor.HttpContextHolder> _httpContextCurrent = new AsyncLocal<HttpContextAccessor.HttpContextHolder>();
```

Важно

Он создаётся один на запрос, не может быть случая перетирания через void



Artsofte

Тоже самое с Activity

```
private static readonly AsyncLocal<Activity> s_current;
```

```
public static Activity? Current
{
  get => Activity.s_current.Value;
  set
  {
    if (!Activity.ValidateSetCurrent(value))
      return;
    Activity.SetCurrent(value);
  }
}
```

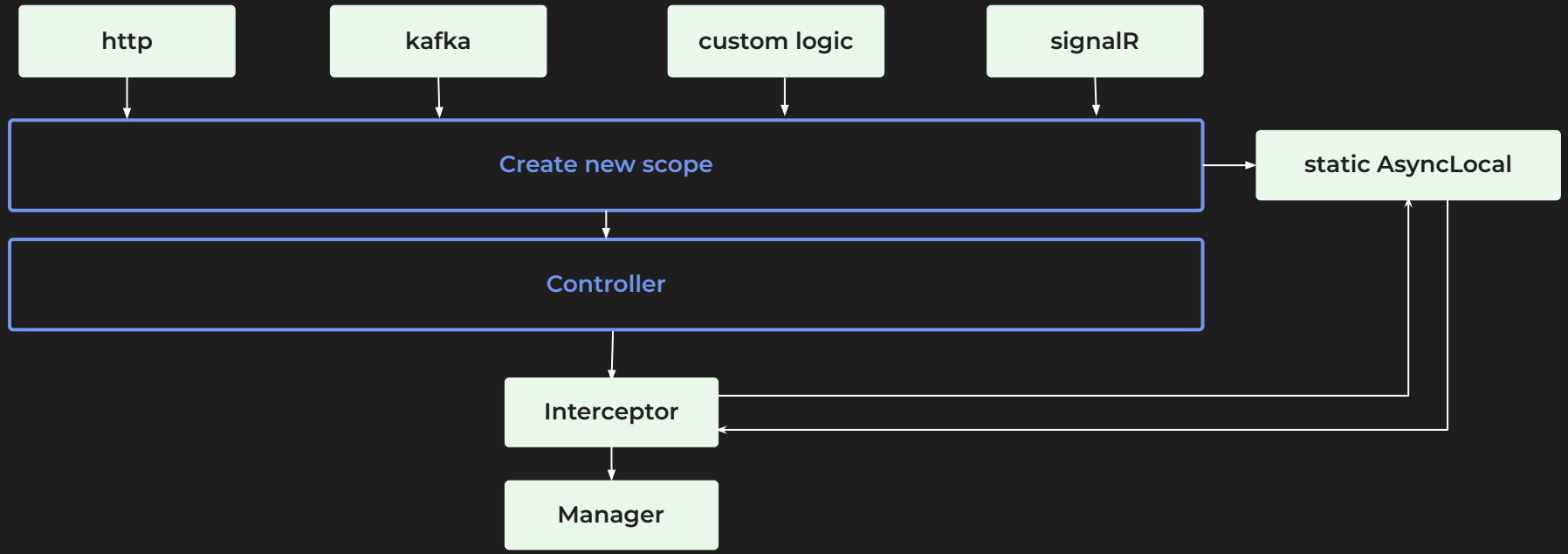
```
public void InterceptSynchronousHandlerVoid(IInvocation invocation)
{
    var activity = _serviceDiagnostic.StartActivity(invocation.Method.Name, ActivityKind.Internal);
    var activityStatusCode = ActivityStatusCode.Ok;
    try
    {
        invocation.Proceed();
    }
    catch
    {
        activityStatusCode = ActivityStatusCode.Error;
        throw;
    }
    finally
    {
        activity?.SetStatus(activityStatusCode);
        activity?.Stop();
    }
}
```

Serilog

```
public static class LogContext
{
    IL code
    private static readonly AsyncLocal<EnricherStack?> Data = new AsyncLocal<EnricherStack>();
```

- Хранит свойства, которые подставляет в строку лога в AsyncLocal

Реализация через runtime



Artsofte

Почему AsyncLocal может нас подвести

- Метод Void завладеет контекстом из метода MainAsync

```
[Transaction(CreateType.CreateNew)]
public async Task MainAsync(IServiceProvider root)
{
    _currentState.Value = 1;
    Console.WriteLine(_currentState.Value);
    Void();
    Console.WriteLine(_currentState.Value);
    await VoidAsync();
    Console.WriteLine(_currentState.Value);
}

[Transaction(CreateType.CreateNew)]
1 usage
private void Void()
{
    Console.WriteLine(_currentState.Value);
    _currentState.Value = 2;
    Console.WriteLine(_currentState.Value);
}

[Transaction(CreateType.UseExistOrCreateNew)]
1 usage
private Task VoidAsync()
{
    Console.WriteLine(_currentState.Value);
    _currentState.Value = 3;
    Console.WriteLine(_currentState.Value);

    return Task.CompletedTask;
}
```

```
1
1
2
2
2
3
3
```


Artsofte

Почему AsyncLocal может нас подвести

```
private static readonly AsyncLocal<IServiceProvider> _asyncLocal = new();
private async Task ActionTask()
{
    var scopeOne = _serviceProvider.CreateScope();
    _asyncLocal.Value = scopeOne.ServiceProvider;
    var scopeTwo = _serviceProvider.CreateScope();
    _asyncLocal.Value = scopeTwo.ServiceProvider;
}
```

Вывод

Если мы хотим получать доступ до DI через AsyncLocal,
то нужно запретить CreateScope, **проксировать тоже не вариант**

Как запретить создавать scope

```
private static void Analyze(SyntaxNodeAnalysisContext context)
{
    var memberAccess = (MemberAccessExpressionSyntax)context.Node;
    if (memberAccess.Name.ToString() == "CreateScope" && memberAccess.Expression is IdentifierNameSyntax identifier)
    {
        if (context.SemanticModel.GetSymbolInfo(identifier).Symbol is ILocalSymbol variableSymbol && variableSymbol.Type.ToString() != "IServiceProvider")
        {
            var diagnostic = Diagnostic.Create(_createScopeMethodUsageIsProhibitedDescriptor, memberAccess.GetLocation(), identifier.Identifier.Text);
            context.ReportDiagnostic(diagnostic);
        }
    }
}
```

```
public static async Task Main()
{
    IServiceProvider t = new ServiceCollection().BuildServiceProvider();
    using var scope = t.CreateScope();
}
```

DN1000: Don't use CreateScope method

```
public static IServiceScope CreateScope(this IServiceProvider provider)
 in class Microsoft.Extensions.DependencyInjection.ServiceProviderServiceExtensions
```

Creates a new IServiceScope that can be used to resolve scoped services.

Params: **provider** – The IServiceProvider to create the scope from.

Returns: A IServiceScope that can be used to resolve scoped services.

`ServiceProviderServiceExtensions.CreateScope` on docs.microsoft.com ↗

Нас начали засыпать в Jira вот таким

```
13:00:36:036 LEVEL:[Information] Task faulted. Exceptions:
InvalidProgramException => Common Language Runtime detected an invalid program.
```

Проблемы с Fody

1. Баги с IL
(аналогичное видели в dapper)
2. Накладные расходы: +3 класса на каждый метод (Object Pool туда не добавишь)
3. Не очень так и бесшовно его устанавливать приходится (править каждый csproj)
4. DI: если запретить создавать scope разработчикам — мы не умели и не решились
5. Async/await: можно сделать
6. Тип перехвата: любой

Вывод

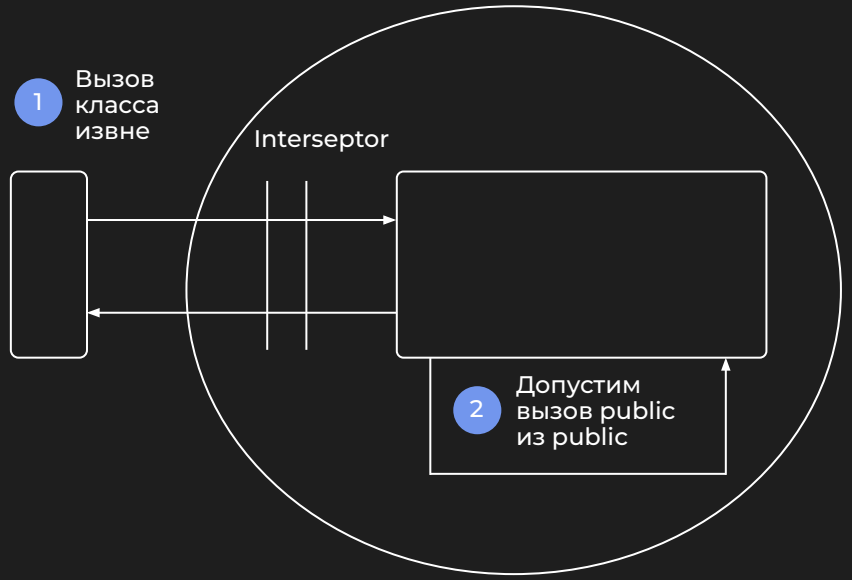
Из-за совокупности этих проблем мы применили его только на части ненагруженных сервисов и **двинулись искать новое решение**

Генерация Type в Runtime — DynamicProxy и DispatchProxy

Artsofte

Proxy Interceptor

крутая статья,
как этим пользоваться

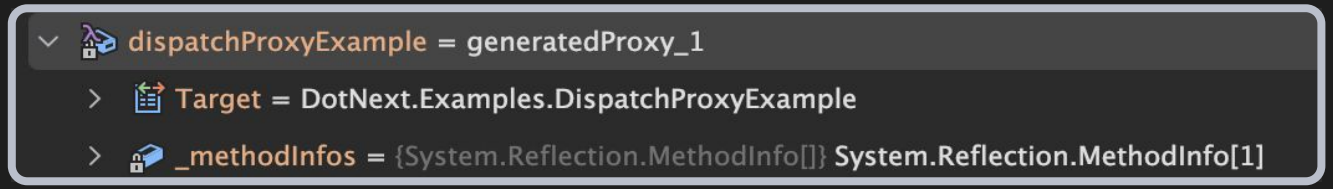


Важно

В отличие от посткомплиции методы, реализованные через такие интерсепторы, работают только на публичные вызовы!!

Artsofte

Что на выходе?




```
dispatchProxyExample.GetType().GetMethods()
{System.Reflection.MethodInfo[6]}
  [0]: {Int32 Action()}
  [1]: {DotNext.Examples.IDispatchProxyExample get_Target()}
  [2]: {System.Type GetType()}
  [3]: {System.String ToString()}
  [4]: {Boolean Equals(System.Object)}
  [5]: {Int32 GetHashCode()}
```

Главная разница DynamicProxy и DispatchProxy

	DynamicProxy	DispatchProxy
Возможность перехвата	По интерфейсу, по классу с virtual methods. Можно интерсептировать вообще без target	По интерфейсу
Дополнительные возможности	1) Можно настраивать фильтр на то, какие методы интерсептировать и какими перехватчиками из указанных 2) Можно использовать слабые ссылки 3) Можно явно кидать ошибку, если пометили Private метод (рантайм)	Нет сложных опций в коробке

Плюс класс

```
proxy.GetType().GetField("__target", BindingFlags.Instance | BindingFlags.NonPublic)
```

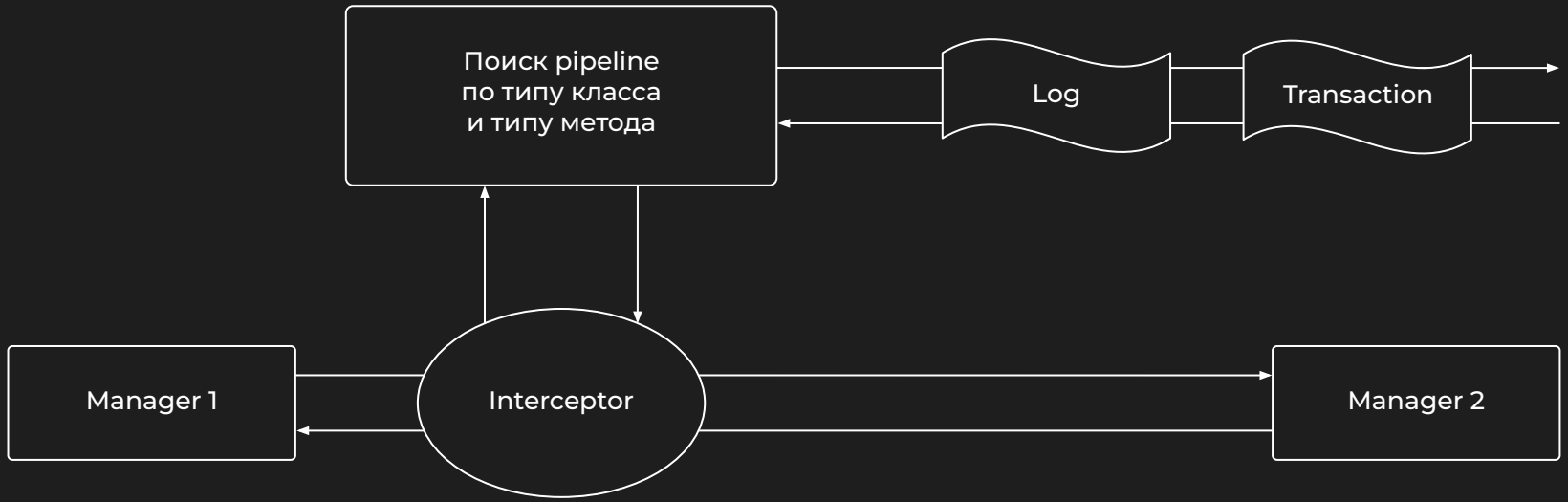
```
> $result = {System.Reflection.RtFieldInfo} IdentityDalLib.Repositories.Common.SimplePasswordRepository __target
```

1. Через рефлексию/expression (количество компиляций будет конечным) можно заменять заменять значения
2. Необходим ObjectPool, так как, пока класс, захваченный прокси, не произведет Dispose, мы можем создать гонку
3. Нет смысла использовать это для Singleton классов

Разные типы оберток

```
if (ensureRes == EnsureInterfaceInterceptionAppliesType.ByInterface)
{
    Очень выжный код
    var proxy :object = _proxyGenerator.CreateInterfaceProxyWithTarget(
        theInterface,
        interfaces,
        context.Instance,
        params interceptors: new Interseptor(existAttribute, serviceProvider));
    context.Instance = proxy;
}

if (ensureRes == EnsureInterfaceInterceptionAppliesType.ByClass)
{
    var proxy :object = _proxyGenerator.CreateClassProxyWithTarget(
        classToProxy: context.Instance.GetType(),
        context.Instance,
        params interceptors: new Interseptor(existAttribute, serviceProvider));
    context.Instance = proxy;
}
```

Pipeline + interceptor chain



Artsofte

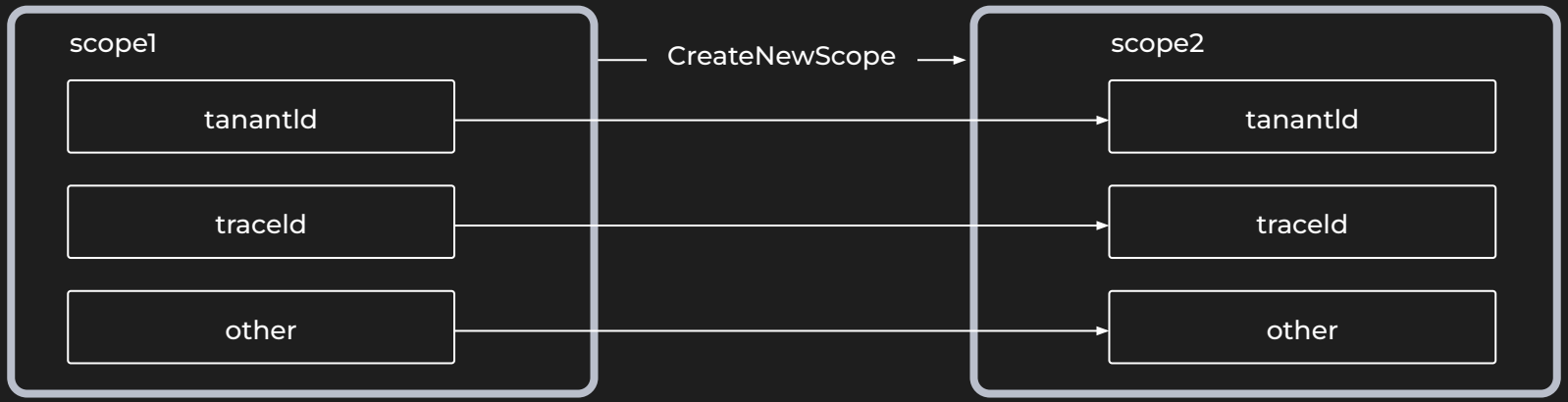
DI чтобы применить DynamicProxy

Artsofte

Что нам в совокупе нужно от DI?

(начало 2023 года)

1. Доступ к CreateScope для переноса трассировочных значений



2. Доступ к классу на выходе, чтобы обернуть его в проксю
3. Injection Key из коробки (делали его сами)

Artsofte

Как мы хотели ?

Наивные

```
services.AddScoped<IServiceProvider, CustomServiceProvider>();
```

```
public class CustomServiceProvider : IServiceProvider, ISupportRequiredService
{
    private readonly IServiceProvider _serviceProvider;

    degorov *
    public CustomServiceProvider(IServiceProvider serviceProvider)
    {
        _serviceProvider = serviceProvider;
    }

    2 usages  degorov
    internal IServiceProvider ServiceProvider => _serviceProvider;

    /// <inheritdoc />
    degorov *
    public object GetService(Type serviceType)
        => GetServiceInner(_serviceProvider.GetService(serviceType));

    /// <inheritdoc />
    degorov *
    public object GetRequiredService(Type serviceType)
        => GetServiceInner(((ISupportRequiredService)_serviceProvider).GetRequir

    /// <summary>
    /// Внутренняя логика создания сервиса
    /// </summary>
    2 usages  degorov *
    private object GetServiceInner(object service)
    {
        if (service is IServiceScopeFactory serviceScopeFactory)
        {
            var handlerServiceScopeFactories :IEnumerable<IHandlerServiceScopeFactory> =
                _serviceProvider.GetServices<IHandlerServiceScopeFactory>();
            return new CustomServiceScopeFactory(
                serviceScopeFactory,
                handlerServiceScopeFactories);
        }

        return service;
    }
```


Artsofte

Базовый DI внутри себя через IL Emit создаёт всё дерево

Получили

Вообще в базовом DI
нет логики декорирования
или интерсептирования!!!

Нет такого места, где можно
перехватить получение
внутренних зависимостей

```
protected override object? VisitConstructor(
  ConstructorCallSite constructorCallSite,
  ILEmitResolverBuilderContext argument)
{
  foreach (ServiceCallSite parameterCallSite in constructorCallSite.ParameterCallSites)
  {
    this.VisitCallSite(parameterCallSite, argument);
    if (parameterCallSite.ServiceType.IsValueType)
      argument.Generator.Emit( opcode: OpCodes.Unbox_Any, parameterCallSite.ServiceType);
  }
  argument.Generator.Emit( opcode: OpCodes.Newobj, constructorCallSite.ConstructorInfo);
  if (constructorCallSite.ImplementationType.IsValueType)
    argument.Generator.Emit( opcode: OpCodes.Box, constructorCallSite.ImplementationType);
  return (object) null;
}
```

А что с декорированием самого IServiceProvider?

- Нельзя декорировать IServiceProvider (можно частично, но легко ломается)

```
internal ServiceProvider(
  ICollection<ServiceDescriptor> serviceDescriptors,
  ServiceProviderOptions options)
{
  this.Root = new ServiceProviderEngineScope( provider: this,  isRootScope: true);
  this._engine = this.GetEngine();
  this._createServiceAccessor = new Func<ServiceIdentifier, ServiceProvider.ServiceAccessor>(this.Crea
  this._serviceAccessors = new ConcurrentDictionary<ServiceIdentifier, ServiceProvider.ServiceAccessor
  this.CallSiteFactory = new CallSiteFactory(serviceDescriptors);
  this.CallSiteFactory.Add(ServiceIdentifier.FromServiceType(typeof (IServiceProvider)), (ServiceCallS
  this.CallSiteFactory.Add(ServiceIdentifier.FromServiceType(typeof (IServiceScopeFactory)), (ServiceC
  this.CallSiteFactory.Add(ServiceIdentifier.FromServiceType(typeof (IServiceProviderIsService)), (Ser
  this.CallSiteFactory.Add(ServiceIdentifier.FromServiceType(typeof (IServiceProviderIsKeyedService)),
```

А что с декорированием самого IServiceProvider?

- Можно пойти следующим образом

```
var builder = WebApplication.CreateBuilder(args);
builder.Host.UseServiceProviderFactory(new CustomServiceProviderFactory());
```

IServiceScopeFactory, IServiceProvider, IServiceScope, ISupportRequiredService

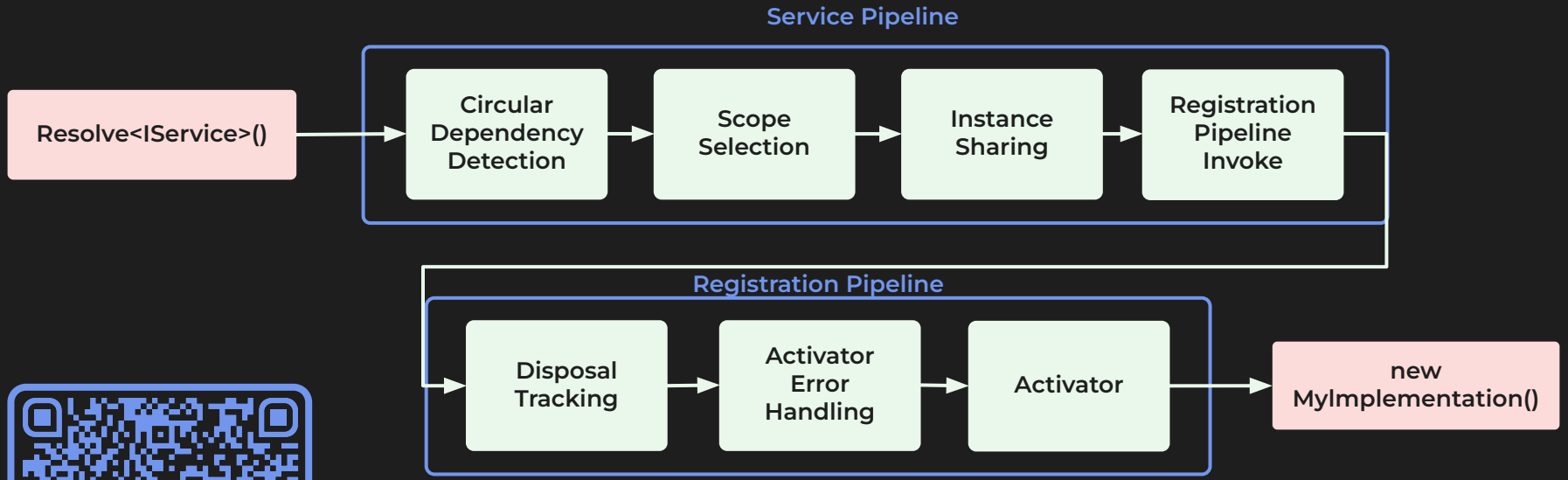
Переопределение IServiceProvider

- Не получат нашу реализацию:
  Controller, Endpoint, либы — в общем, все,
  у кого инжекция не от корня идёт

```
[Route( template: "setting-password")]
public class PasswordSettingInternalController : ControllerBase
{
    private readonly IServiceProvider _serviceProvider;  _servic

    public PasswordSettingInternalController(IServiceProvider ser
    {
        _serviceProvider = serviceProvider;  serviceProvider: Se
    }

    [HttpGet]
    public async Task<ActionResult> GetExamplePasswordAsync()
    {
        return Ok();
    }
}
```

- ! Контролируемую область кода можно
  обезопасить через анализатор!!!
  Запрет инжекции IServiceProvider

Мы пошли в сторону autofac



Конвейер получения класса из DI

Artsofte

Pure Di

```
partial class Composition: IInterceptor
{
    private static readonly ProxyGenerator ProxyGenerator
        = new();

    1 usage
    private partial T OnDependencyInjection<T>(
        in T value,
        object? tag,
        Lifetime lifetime)
    {
        return (T)ProxyGenerator // ProxyGenerator
            .CreateInterfaceProxyWithTargetInterface(
            interfaceToProxy: typeof(T),
            value,
            params interceptors: this); // object
    }

    public void Intercept(IInvocation invocation)
    {
        invocation.Proceed();
    }
}
```

Доклад того
года об этом

```
DI.Setup(nameof(Composition)).Bind().To<Service>().Root<IService>( name: "Root");
```

Проблема регистраций и не виртуальных методов

1. Пусть у нас есть такой класс и интерфейс

```
1 usage  1 implementation
public interface ISomeManager { Task ActionAsync(); }
1 usage
public class SomeManager : ISomeManager
{
    [Cache]
    public Task ActionAsync() { throw new NotImplementedException(); }
}
```

2. И такая регистрация

```
public static void TryAddDi(this IServiceCollection serviceCollection)
{
    serviceCollection.TryAddTransient<SomeManager>();
}
```

Проблема: Двойная прокся

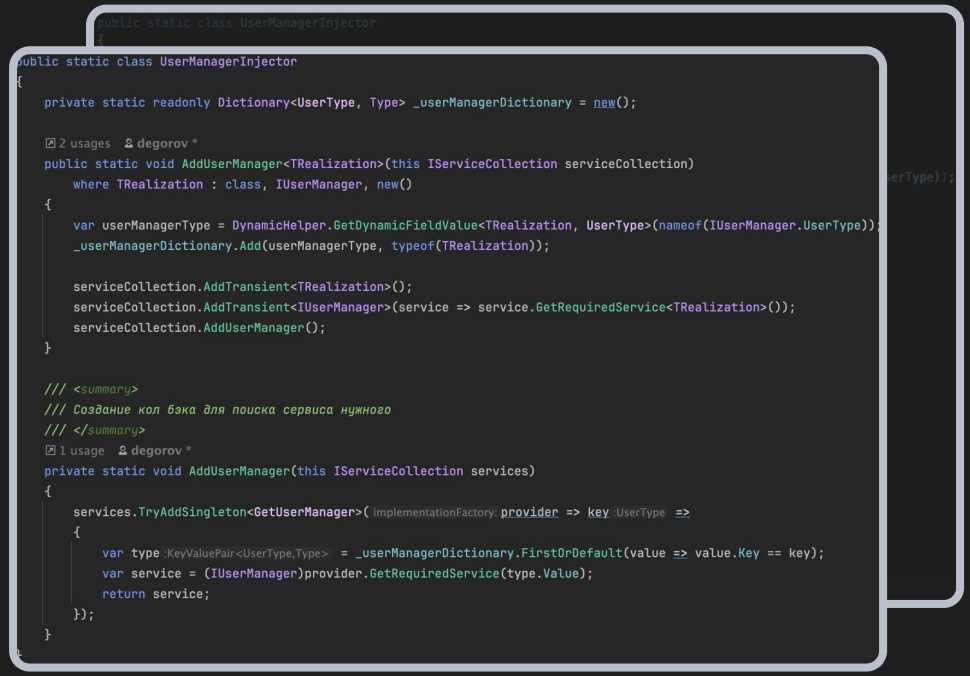
```
public static void TryAddDi(this IServiceCollection serviceCollection)
{
    serviceCollection.TryAddTransient<SomeManager>();
    serviceCollection.TryAddTransient<ISomeManager>( implementationFactory: sp :IServiceProvider  => sp.GetRequiredService<SomeManager>());
}
```

Аналогично тому, чтобы сделать вот так:

```
var proxyGenerator = new ProxyGenerator();
var castleProxyClass :object?  = proxyGenerator.CreateClassProxyWithTarget(typeof(DispatchProxyExample),  target: new DispatchProxyExample(),  interceptors: new []{ new LogStandardInterceptor()});
var castleProxyInterface :object?  = proxyGenerator.CreateInterfaceProxyWithTarget(typeof(IDispatchProxyExample), castleProxyClass,  interceptors: new []{ new LogStandardInterceptor()});
```

Проблема: Класс из DI извлекается по ключу типа, а не интерфейса

→ перешли на базовый InjectionKey



```
public static class UserManagerInjector
{
    private static readonly Dictionary<UserType, Type> _userManagerDictionary = new();

    2 usages  degorov *
    public static void AddUserManager<TRealization>(this IServiceCollection serviceCollection)
        where TRealization : class, IUserManager, new()
    {
        var userManagerType = DynamicHelper.GetDynamicFieldValue<TRealization, UserType>(nameof(IUserManager.UserType));
        _userManagerDictionary.Add(userManagerType, typeof(TRealization));

        serviceCollection.AddTransient<TRealization>();
        serviceCollection.AddTransient<IUserManager>(service => service.GetRequiredService<TRealization>());
        serviceCollection.AddUserManager();
    }

    /// <summary>
    /// Создание кол бэка для поиска сервиса нужного
    /// </summary>
    1 usage  degorov *
    private static void AddUserManager(this IServiceCollection services)
    {
        services.TryAddSingleton<GetUserManager>( implementationFactory: provider => key :UserType =>
        {
            var type :KeyValuePair<UserType,Type> = _userManagerDictionary.FirstOrDefault(value => value.Key == key);
            var service = (IUserManager)provider.GetRequiredService(type.Value);
            return service;
        });
    }
}
```


Artsofte

Проблема: Анализ через рефлексию

```
foreach (var externalActionFactory in _externalActionFactoryList)
{
    var actionName :string = externalActionFactory.Name;
    var action = externalActionFactory.GetExternalAction();
    var actionType = action.GetType();

    var actionContextType = ActionContextHelper.GetActionContextTypeOrDefault(actionType);
```

Вместо нужного типа мы получили тип обертки

```
var field = proxyType.GetField( name: "__target",  bindingAttr: BindingFlags.Instance | BindingFlags.NonPublic);
var type = field.GetMemberType();
```

Итоге по DynamicProxy и Autofac

1. Есть полная поддержка DI
2. Async/await — решение аналогично Fody
3. Перехват только публичных методов между классами.
4. Встраивание через прокси над классом, нет безумных багов с IL, но есть баги с Reflection и DI
5. Накладные расходы при transient: +1 класс прокси (решается через ObjectPool), +1 класс для Invocation с параметрами и ещё внутренние классы. В общем, лучше по числу не стало(

! На этом можно было бы остановиться, сказать, что решение красивое, но не подходит для нагруженных сервисов, но тут появился новый релиз .net!!!

Interceptors in C# 12

Artsofte

Interceptors in C# 12

Еще до появления interceptor можно было использовать source generate для реализации АОП

Но тут были минусы

- partial class

Interceptors in C# 12

```
var interseptorExample = new InterseptorExample();
await interseptorExample.Action2Async ();
```

```
[ InterceptsLocation( filePath: "/Users/dmitrijegorov/Desktop/DotNext/DotNext/Program.cs",  line: 66,  character: 26)]
2 usages
public static async Task Action2Async(this InterseptorExample example)
{
    Console.WriteLine(nameof(Action2Async) + "1");
    await example.Action2Async();
    Console.WriteLine(nameof(Action2Async) + "2");
}
```

1. Поддержка async/await
2. Работают через указание path line char
3. Если вы используете Rider есть интеграция

Никаких лишних классов не создаётся, всё максимально экономично через статику

```
var interseptorExample = new InterseptorExample();
await interseptorExample.ActionAsync ();
```

до

```
InterseptorExample interseptorExample = new InterseptorExample();
await GeneratedCode.ActionAsync(interseptorExample);
```

после

1. Конечно минус, что тут ситуация с DI аналогично Fody
2. Перехват только публичных методов, зато как между, так и внутри классов (в теории можно вообще любые)

Простой пример (source generate)

```
context.RegisterPostInitializationOutput(ctx :IncrementalGeneratorPostInitializationContext =>
{
    var sourceText = $$"""
                        using System.Runtime.CompilerServices;

                        namespace SourceGeneratorInCSharp;

                        public static class GeneratedCodeTwo
                        {
                            [InterceptsLocation("/Users/dmitrijegorov/Desktop/CodeMaz
                            public static string InterceptorMethod(this Example examp
                            {
                                return $"{text}, YES";
                            }
                        }

                        """;

    ctx.AddSource( hintName: "Inter.g", SourceText.From(sourceText, Encoding.UTF8));
});
```

Чуть сложнее

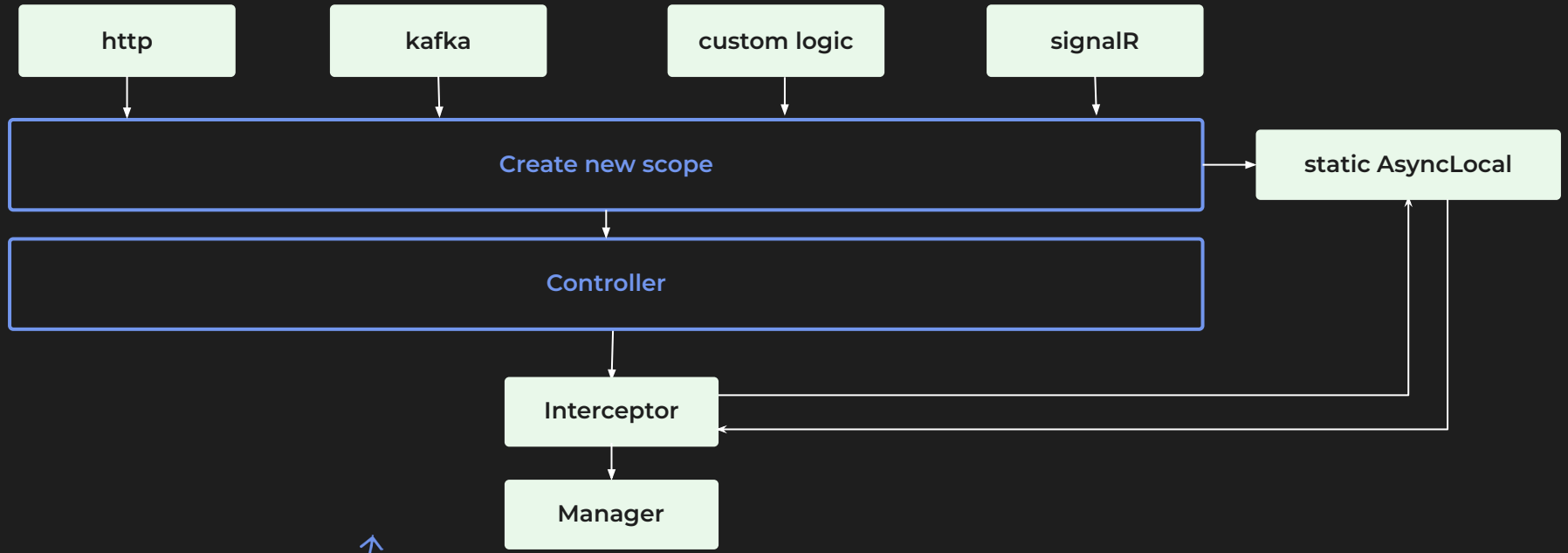

```
var providerInvocationExp = context.SyntaxProvider.CreateSyntaxProvider(
    predicate: (n :SyntaxNode , _) => n is MemberAccessExpressionSyntax,
    transform: (n :GeneratorSyntaxContext , cp :CancellationToken ) =>
    {
        var node = (MemberAccessExpressionSyntax)n.Node;
        var semanticModel = n.SemanticModel;
        var location = node.Name.GetLocation();
        var lineSpan = location.GetLineSpan();
        var startLine :int =  lineSpan.StartLinePosition.Line + 1;
        var startColumn :int  = lineSpan.StartLinePosition.Character + 1;
        var symbol = semanticModel.GetSymbolInfo(node, cp).Symbol;
        var hasAttribute :bool?  = symbol?.GetAttributes() // ImmutableArray<AttributeData
            .Any(attribute => attribute.AttributeClass?.Name == "InterceptAttribu
        if (hasAttribute != true)
        {
            return null;
        }
        var typeInfo = semanticModel.GetTypeInfo(node.Expression);
        var className = "";
        var namespaceName = "";
        if (typeInfo.Type is INamedTypeSymbol namedTypeSymbol)
        {
            className = namedTypeSymbol.Name;
            namespaceName = namedTypeSymbol.ContainingNamespace.ToDisplayString()
        }
        var filePath :string  = location.SourceTree?.FilePath;
        return new MethodInfoToIntercept(className, namespaceName,  methodName: noc
    });
```


Artsofte

От создателей таких хитов, как «ваше приложение больше не скомпилируется с namespace Newtonsoft»

Мы запрещаем использовать CreateScope вне CoreLib

И возвращаемся к этой концепции



И возвращаемся
к этой концепции

Artsofte

Итоговый вид перехватчика

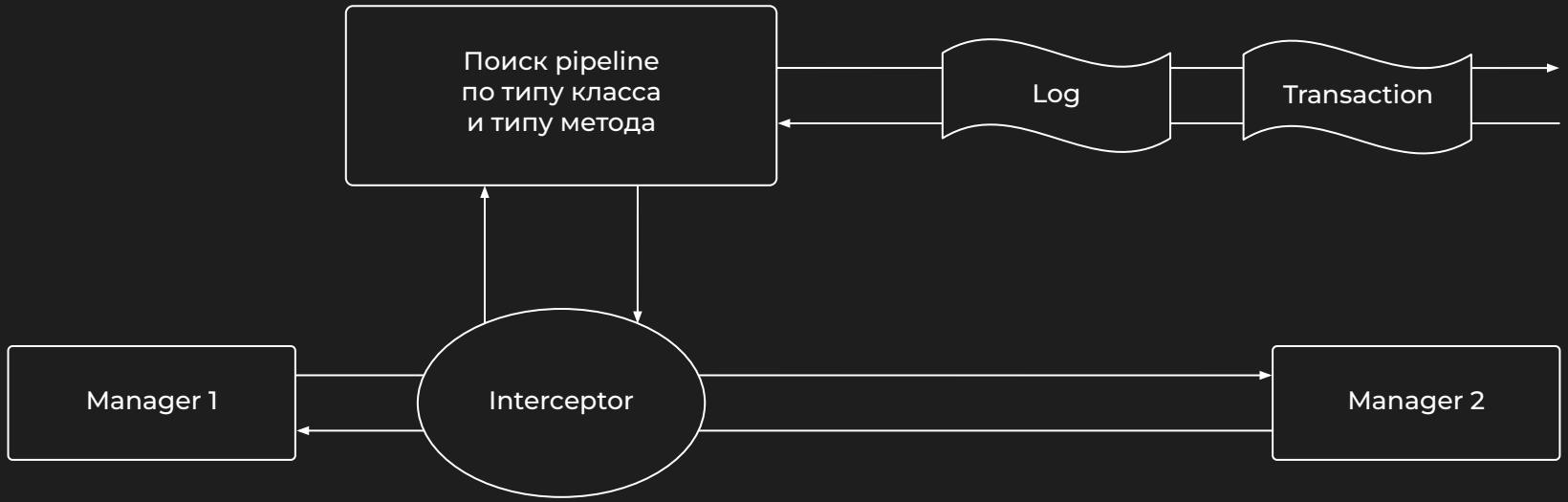
```
// ReSharper disable once InconsistentNaming
private static readonly Type _interceptorExampleType = typeof(InterseptorExample);
// ReSharper disable once InconsistentNaming
private static readonly MethodInfo _interceptorExampleTypeAction = typeof(InterseptorExample).GetMethod( name: "Actio

[ InterceptsLocation( filePath: "/Users/dmitrijegorov/Desktop/DotNext/DotNext/Program.cs", line: 66, character: 26)]
public static async Task ActionAsync(this InterseptorExample example)
{
    if (InterceptPipelineCreator.Store.TryGetValue(_interceptorExampleType, out var methodList :Dictionary<MethodInfo,Inter
    {
        if (methodList.TryGetValue(_interceptorExampleTypeAction, out var intercept))
        {
            var executeContext = ExecuteInterceptorContextPool.Pool.Get();
            try
            {
                executeContext.ServiceProvider = ServiceProviderStore.ServiceProvider.Value;
                await intercept(executeContext);
            }
            finally
            {
                ExecuteInterceptorContextPool.Pool.Return(executeContext);
            }

            return;
        }
    }

    await example.ActionAsync();
}
```

ExecuteInterceptorContext

```
public class ExecuteInterceptorContext
{
    2 usages
    public required IServiceProvider ServiceProvider { get; set; }

    internal object ExecuteClass { get; set; }
}
```

```
public class ExecuteInterceptorContext<T, U> : ExecuteInterceptorContext
{
    public T Param1;

    public U Param2;
}
```

Pipeline + interceptor chain



Artsofte

Итоги по Interceptor

1. Накладные расходы — сам по себе движок не создает их
2. Async/await — поддержка из коробки без магии
3. Перехват любых методов вызываемых из любых методов (expression)

Без трудностей не обошлось

Частичное использовать интерсепторов

1. Было

```
public async Task DeleteActionAsync(Guid actionId)
{
    await using var lockHandler :IAsyncDisposable = await _keyedSemaphoresCollection.LockAsync(actionId, timeout: null);
    var transaction = await _dbConnectionFactory.StartTransactionAsync();
```

2. Сломали

```
[Transaction]
⧖ 0+1 usages
public async Task DeleteActionAsync(Guid actionId)
{
    await using var lockHandler :IAsyncDisposable = await _keyedSemaphoresCollection.LockAsync(actionId, timeout: null);
```

3. Итог

```
[Transaction]
[Lock(nameof(actionId))]
⧖ 0+1 usages
public async Task DeleteActionAsync(Guid actionId)
{
```

Лейзи поведение для атрибута

Транзакция и соединение должны создаваться в момент первого обращения в БД, а не заранее, иначе нельзя делать так:

```
[Transaction]
public async Task<UpdateOnAutIdentityResponse> UpdateOnAuthIdentityAsync(UpdateOnAutIdentityRequest request)
{
    var identityUser = await _identityConnectionService.GetUserByIdAsync(new GetUserByIdRabbitRequest(request.Id));
    var isProvider = await _providerProfileRepository.ExistByAsync(nameof(ProviderProfileDal.IdentityUserId), request.Id);
```

Не всегда получается гладко использовать атрибуты

```
/// <inheritdoc/>
public async Task SaveFileAsync(byte[] fileContent, string fileNameWithExtension, string directoryPath)
{
    var endLog = LogHelper.StartLog(fileNameWithExtension, directoryPath, _fileStoreDirectoryAccess.DirectoryStore);

    var contentPath = Path.Combine(_fileStoreDirectoryAccess.DirectoryStore, directoryPath);
    await CreateFile(fileContent, fileNameWithExtension, contentPath);

    endLog();
}
```

- Как правило, это говорит о неправильной декомпозиции методов

Пример невалидного кода

```
public async Task DeleteAsync(ObjectId id)
{
    var transaction = await _conditionRepository.BeginTransactionAsync();
    var entity :ConditionSchemaDal = await _conditionRepository.GetAsync(id);

    await _conditionRepository.DeleteAsync(entity.Id, transaction);

    await transaction.CommitAsync();

    var conditionSchemaDeletedEvent = new ConditionSchemaDeletedEvent
    {
        ConditionSchemaId = id
    };

    _brokerSenderService.SendEvent(conditionSchemaDeletedEvent);
}
```

```
[Transaction]
public async Task DeleteAsync(ObjectId id)
{
    var entity :ConditionSchemaDal = await _conditionRepository.GetAsync(id);

    await _conditionRepository.DeleteAsync(entity.Id);

    var conditionSchemaDeletedEvent = new ConditionSchemaDeletedEvent
    {
        ConditionSchemaId = id
    };

    _brokerSenderService.SendEvent(conditionSchemaDeletedEvent);
}
```

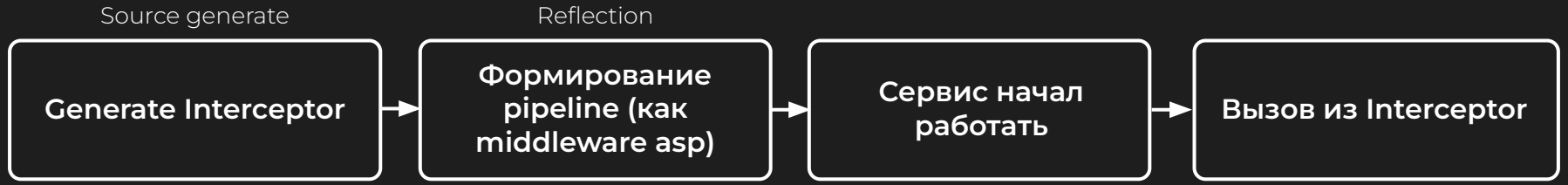
Подводим итоги

Artsofte

Сравнительная таблицы функциональныз возможностей

Тип	Плюс классов	Внешнее состояние	Async /await	Какие вызовы можно перехватить?	Где идет встраивание?
Fody + Default DI	3 + x	AsyncLocal	через delegate	любые	метод, который вызываем
DispatchProxy + Autofac	2 + n + x	DI	через delegate	public interface & virtual class	прокся
Interceptor source gen	1 (objectPool) + x	AsyncLocal	+	любые	метод, который вызывает

Итоговая схема



- Движок: Interceptor c#12
- Async/Await: проблем нет
- Pipeline: Как middleware
- Накладные расходы: ExecuteInterceptorContext для pipeline (ObjectPool)
- Di в interceptor: AsyncLocal
- Di движок: Откатились к базовому (переходим на Pure Di)
- Запретили: CreateScope
- Перехват любых методов из любых методов

Есть проблема, мы не можем интерсептировать внутри сторонних либ, которые расширяем через инверсию (но такое встречается у нас редко)

Artsofte