SolarLab>_

Аспектно-ориентированное

программирование в .NET

План

1. Основные концепции парадигмы аспектно-ориентированного программирования
2. Пример использования парадигмы
3. Возможные реализации АОП в .NET

Зачем мне это ваше АОП?

01 Улучшение читаемости и поддерживаемости кода

02 Эффективное решение задач декомпозиции

03 Добавление функциональности без изменения модулей

Определения

Аспектно-ориентированное программирование

Это парадигма программирования, направленная на решение задач декомпозиции кода посредством использования конструкций АОП – **аспектов**

Основная задача

Улучшение модульности программы, путем решения проблемы **“сквозной функциональности”**

Функциональность (concern)

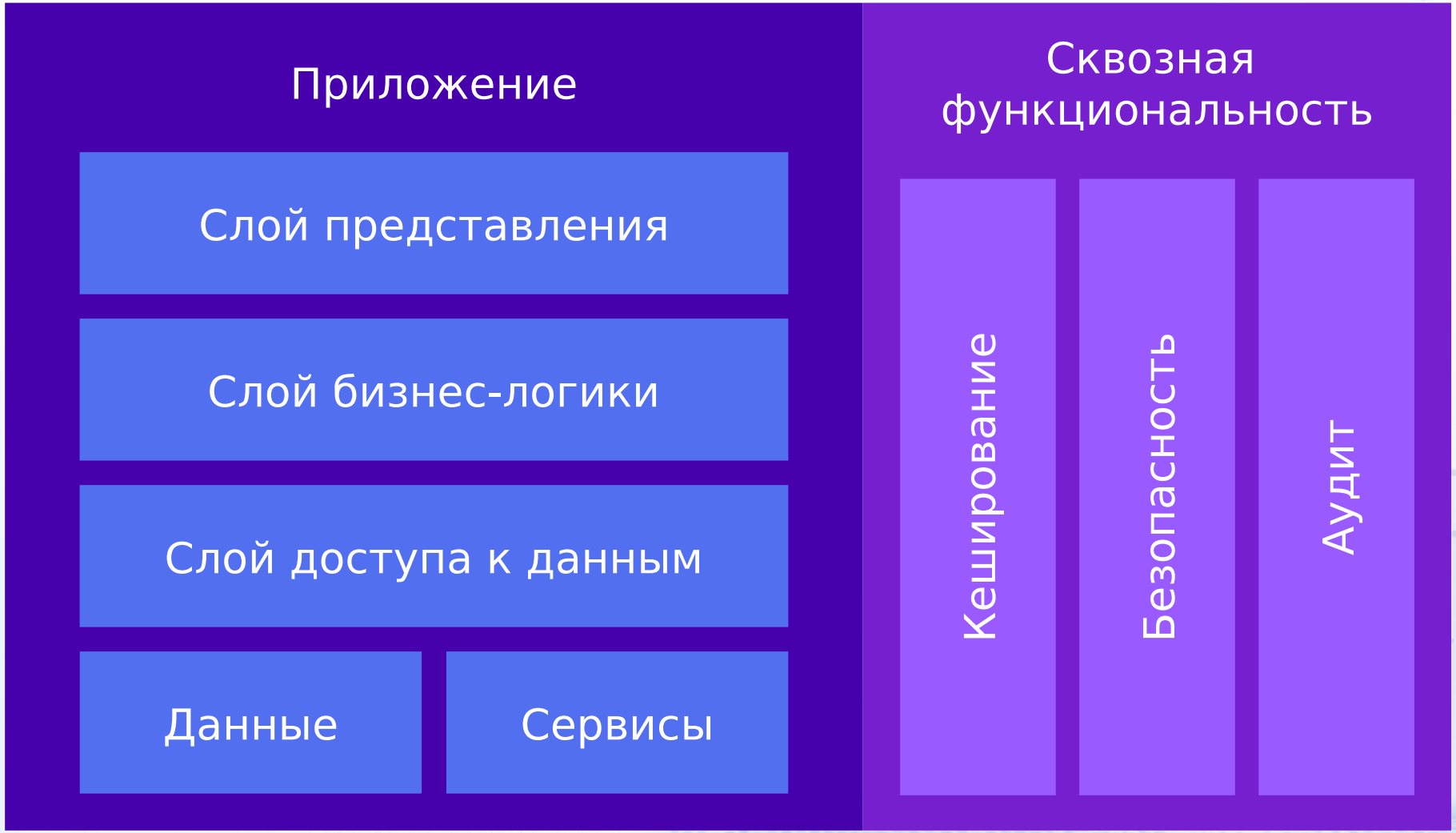
Функциональность – это некоторая задача (функция) системы

Основная (Core Concern)	**Второстепенная** (Non-core concern)
Аренда машин	Логирование
Проведение закупок	Кеширование

Функциональность (concern)

Сквозная функциональность (cross-cutting concern)
это функциональность, реализация которой рассредоточена на несколько модулей или слоев приложения

Сквозная функциональность



Определения

Аспект - это некоторая (не основная) задача программы, вынесенная в отдельный модуль

Совет (Advice) - код, который выполняет задачу сквозной функциональности (например логирование)

Точка прикрепления (join point) - место действия аспекта

Срез (point cut) - описание всех мест действия аспекта

SolarLab>_

Простой пример

аренда машины

Пример

Запрос на аренду

Валидация

Обновление записи в БД

Ответ

```
public class CarRentService : ICarRentService
{
    private readonly IRepository<Car> _carRepository;

    public CarRentService(IRepository<Car> carRepository)
    {
        _carRepository = carRepository;
    }

    public void RentCar(CarRentRequest request)
    {
        var car = _carRepository.Get(request.CarId);

        car.RentEndDate = DateTime.UtcNow.Add(request.RentDuration);
        car.RenterId = request.RenterId;

        _carRepository.Update(car);
    }
}
```

Добавили логирование

```
public void RentCar(CarRentRequest request)
{
    _logger.Log($"{DateTime.Now:u} | Начало выполнения метода '{nameof(RentCar)}'");

    var car = _carRepository.Get(request.CarId);

    car.RentEndDate = DateTime.UtcNow.Add(request.RentDuration);
    car.RenterId = request.RenterId;

    _carRepository.Update(car);

    _logger.Log($"{DateTime.Now:u} | Завершено выполнение метода '{nameof(RentCar)}'");
}
```

```
public void RentCar(CarRentRequest request)
{
    _logger.Log($"{DateTime.Now:u} | Начало выполнения метода '{nameof(RentCar)}'");

    if (request == null)
        throw new ArgumentNullException(nameof(request));

    if (request.CarId <= 0)
        throw new ArgumentException("Идентификатор машины не задан", nameof(request.CarId));

    if (request.RenterId <= 0)
        throw new ArgumentException("Идентификатор арендатора не задан", nameof(request.CarId));

    if (request.RentDuration < TimeSpan.FromHours(1))
        throw new ArgumentException("Время проката не может быть меньше часа", nameof(request.RentDuration));

    var car = _carRepository.Get(request.CarId);

    car.RentEndDate = DateTime.UtcNow.Add(request.RentDuration);
    car.RenterId = request.RenterId;

    _carRepository.Update(car);

    _logger.Log($"{DateTime.Now:u} | Завершено выполнение метода '{nameof(RentCar)}'");
}
```

```
public void RentCar(CarRentRequest request)
{
    _logger.Log($"{DateTime.Now:u} | Начало выполнения метода '{nameof(RentCar)}'");

    if (request == null)
        throw new ArgumentNullException(nameof(request));

    if (request.CarId <= 0)
        throw new ArgumentException("Идентификатор машины не задан", nameof(request.CarId));

    if (request.RenterId <= 0)
        throw new ArgumentException("Идентификатор арендатора не задан", nameof(request.CarId));

    if (request.RentDuration < TimeSpan.FromHours(1))
        throw new ArgumentException("Время проката не может быть меньше часа", nameof(request.RentDuration));

    try
    {

        var car = _carRepository.Get(request.CarId);

        car.RentEndDate = DateTime.UtcNow.Add(request.RentDuration);
        car.RenterId = request.RenterId;

        _carRepository.Update(car);
    }
    catch (Exception e)
    {
        var carRentException = new Exception(e.Message, e);
        _logger.Error("Произошло исключение при аренде машины. " +
                      $"Запрос: {request}. Сообщение: {e.Message}", carRentException);

        throw carRentException;
    }
    finally
    {
        _logger.Log($"{DateTime.Now:u} | Завершено выполнение метода '{nameof(RentCar)}'");
    }
}
```

```
public void RentCar(CarRentRequest request)
{
    _logger.Log($"{DateTime.Now:u} | Начало выполнения метода '{nameof(RentCar)}'");

    if (request == null)
        throw new ArgumentNullException(nameof(request));

    if (request.CarId <= 0)
        throw new ArgumentException("Идентификатор машины не задан", nameof(request.CarId));

    if (request.RenterId <= 0)
        throw new ArgumentException("Идентификатор арендатора не задан", nameof(request.CarId));

    if (request.RentDuration < TimeSpan.FromHours(1))
        throw new ArgumentException("Время проката не может быть меньше часа", nameof(request.RentDuration));

    try
    {
        var car = _carRepository.Get(request.CarId);

        car.RentEndDate = DateTime.UtcNow.Add(request.RentDuration);
        car.RenterId = request.RenterId;

        _carRepository.Update(car);
    }
    catch (Exception e)
    {
        var carRentException = new Exception(e.Message, e);
        _logger.Error("Произошло исключение при аренде машины. " +
                      $"Запрос: {request}. Сообщение: {e.Message}", carRentException);

        throw carRentException;
    }
    finally
    {
        _logger.Log($"{DateTime.Now:u} | Завершено выполнение метода '{nameof(RentCar)}'");
    }
}
```

Основная задача

```
public void RentCar(CarRentRequest request)
{
    var car = _carRepository.Get(request.CarId);

    car.RentEndDate = DateTime.UtcNow.Add(request.RentDuration);
    car.RenterId = request.RenterId;

    _carRepository.Update(car);
}
```

```
public void RentCar(CarRentRequest request)
{
    var car = _carRepository.Get(request.CarId);

    car.RentEndDate = DateTime.UtcNow.Add(request.RentDuration);
    car.RenterId = request.RenterId;

    _carRepository.Update(car);
}
```

Точка прикрепления совета - “После вызова метода”

Аспект “Логирование выполнения метода”

Совет: записать в лог информацию о начале и завершении метода

```
public void RentCar(CarRentRequest request)
{
    var car = _carRepository.Get(request.CarId);

    car.RentEndDate = DateTime.UtcNow.Add(request.RentDuration);
    car.RenterId = request.RenterId;

    _carRepository.Update(car);
}
```

OnException

Аспект “Логирование выполнения метода”	Аспект “Валидация”	Аспект “Обработка исключений”
Совет: записать в лог информацию о начале и завершении метода	**Совет**: проверить входные параметры метода на корректность	Совет: обернуть исключение в новый тип и залогировать его

Определение

Связывание

это некоторый способ, с помощью которого можно применить аспекты в точках прикрепления

Определение

Связывание

это некоторый способ, с помощью которого можно применить аспекты в точках прикрепления

Статическое (перед запуском программы)		Динамическое (во время выполнения)	
Переписывание исходного кода	Переписывание IL кода	Проксирование	Внедрение в CLR

SolarLab>_

Реализация

решение задач с помощью АОП

Практическая работа (Before/After)

```
1 usage   1 inheritor
public interface ICarRentService
{
    1 implementation
    void RentCar(int carId);
}

public class CarRentService : ICarRentService
{
    public void RentCar(int carId)
    {
        //...
    }
}
```

Декоратор = обертка

```
public class CarRentServiceWrapper : ICarRentService
{
    private readonly ICarRentService _original;

    public CarRentServiceWrapper(ICarRentService original)
    {
        _original = original;
    }

    0+1 usages
    public void RentCar(int carId)
    {
        Logger.Log( message: "Before");
        _original.RentCar(carId);
        Logger.Log( message: "After");
    }
}
```

Решение невозможно переиспользовать, т.к. на каждый интерфейс нужна своя обертка

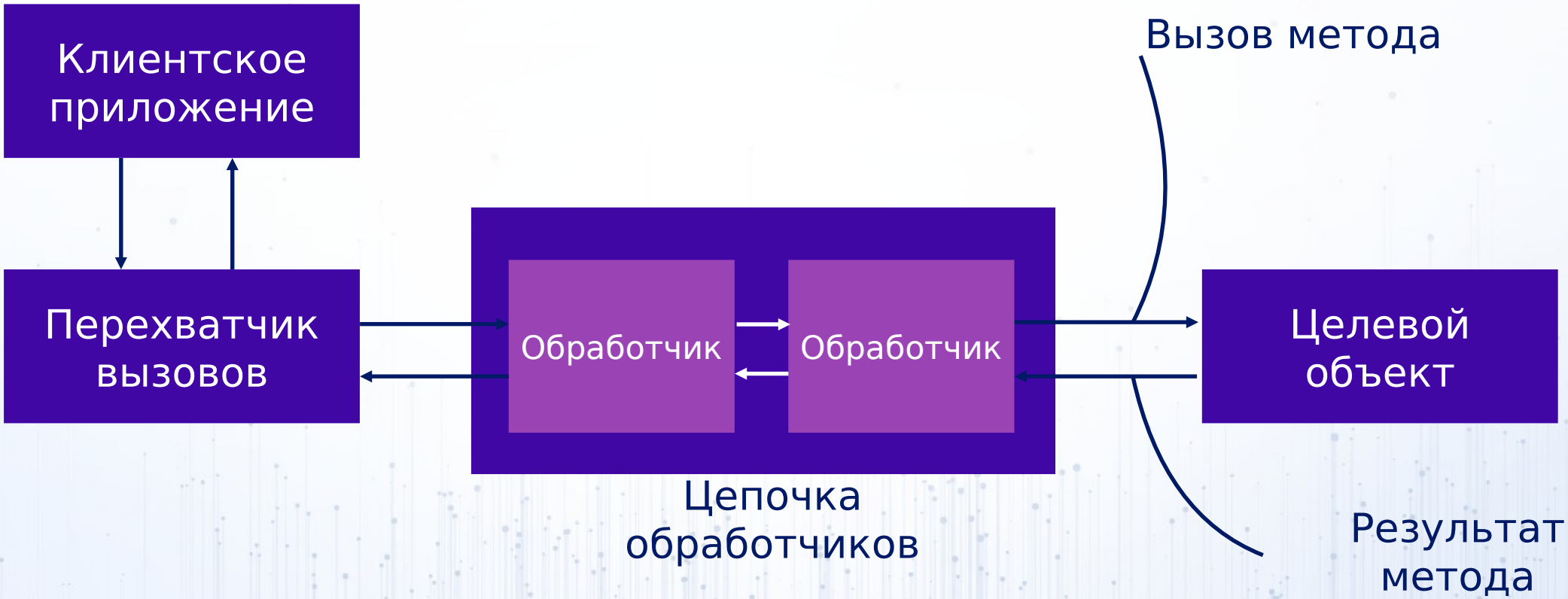
Динамическое связывание – v2

Castle Dynamic Proxy – позволяет создавать динамические прокси-обертки во время выполнения

RealProxy – механизм для создания динамических оберток в .Net Framework

DispatchProxy – механизм для создания динамических оберток в .Net Core

Общий принцип



```
public class BeforeAfterAdvice : IInterceptor
{
    public void Intercept(IInvocation invocation)
    {
        Logger.Log( message: "Before");
        invocation.Proceed();
        Logger.Log( message: "After");
    }
}
```

```
public static class BeforeAfterAspect
{
    public static TService Apply<TService>()
        where TService : class
    {
        var generator = new ProxyGenerator();
        var proxy = generator.CreateClassProxy<TService>(
            new BeforeAfterAdvice());

        return proxy;
    }
}
```

```
public class Program
{
    public static void Main()
    {
        var service = BeforeAfterAspect.Apply<CarRentService>();
        service.RentCar(carId: 1);
    }
}
```

Нет нормальных точек прикрепления
Неудобно применять и переиспользовать

```
[AttributeUsage(AttributeTargets.Method)]
public class BoundaryLoggingAspect : MethodBoundaryAspect
{
    [InjectDependency]
    public ILogger Logger { get; set; }

    public override void OnEntry(IInvocationPipeline pipeline)
    {
        Logger.Log($"{DateTime.UtcNow:u} | Начато выполнение метода " +
                   $"'{pipeline.Context.Method.Name}'");
    }

    public override void OnExit(IInvocationPipeline pipeline)
    {
        Logger.Log($"{DateTime.UtcNow:u} | Завершено выполнение метода " +
                   $"'{pipeline.Context.Method.Name}'");
    }
}
```

- метод

```
[AttributeUsage(AttributeTargets.Method)]
public class BoundaryLoggingAspect : MethodBoundaryAspect
{
    [InjectDependency]
    public ILogger Logger { get; set; }

    public override void OnEntry(IInvocationPipeline pipeline)
    {
        Logger.Log($"{DateTime.UtcNow:u} | Начато выполнение метода " +
                   $"'{pipeline.Context.Method.Name}'");
    }

    public override void OnExit(IInvocationPipeline pipeline)
    {
        Logger.Log($"{DateTime.UtcNow:u} | Завершено выполнение метода " +
                   $"'{pipeline.Context.Method.Name}'");
    }
}
```

Точки применения совета

```
public override void OnEntry(IInvocationPipeline pipeline)
```

```
public override void OnExit(IInvocationPipeline pipeline)
```

```
public class CarRentService : ICarRentService
{
    private readonly IRepository<Car> _carRepository;

    public CarRentService(IRepository<Car> carRepository)
    {
        _carRepository = carRepository;
    }

    [BoundaryLoggingAspect]
    [CarRentRequestValidation]
    [CarRentExceptionHandling]
    public void RentCar(CarRentRequest request)
    {
        var car = _carRepository.Get(request.CarId);

        car.RentEndDate = DateTime.UtcNow.Add(request.RentDuration);
        car.RenterId = request.RenterId;

        _carRepository.Update(car);
    }
}
```

```
1 usage   bitshift *
private static WindsorContainer CreateContainer()
{
    var container = new WindsorContainer();

    AspectsConfigurator
        .UseContainer(new WindsorAspectsContainer(container))
        .Initialize();

    container.Register(
        Component.For<ICarRentService>()
            .ImplementedBy<CarRentService>());

    return container;
}
```

Динамическое связывание

Достоинства

1. Поддержка внедрения зависимостей

2. Код сборки не модифицируется

3. Не требуются дополнительные утилиты для внедрения аспектов

Недостатки

1. Некоторые ограничения для внедрения аспектов

2. Влияет на производительность. Иногда – драматично

3. Код иногда необходимо подстраивать под аспект

Заключение

Аспектно-ориентированное программирование позволяет:

- Улучшить читаемость и поддерживаемость кода
- Увеличить скорость добавления новой функциональности
- Произвести декомпозицию модулей

Ссылки

- https://msdn.microsoft.com/en-us/magazine/dn574804.aspx – Aspect-Oriented Programming with the RealProxy Class
- http://www.castleproject.org/projects/dynamicproxy/ – Castle DynamicProxy
- https://github.com/SolarLabRU/IvorySharp – IvorySharp
- https://2017.dotnext-piter.ru/en/talks/aop-via-net/ – Igor Yakovlev – AOP via .NET
- https://www.manning.com/books/aop-in-net – AOP in .NET