Теория и практика

Dependency Injection

Зачем нужен DI?

Слабосвязанный код

Связанность и связность
Зацепление и прочность
Coupling и cohesion

Мифы про DI

- Это сложно и непонятно
- Это нужно только для unit-тестов
- Это требует фреймворков и контейнеров
- * Service Locator (Абстрактная фабрика) решает задачи DI

Программируйте на абстракциях

Пример 1

Стабильные и нестабильные зависимости

Виды внедрения зависимостей

- Через конструктор
- Через публичное свойство класса
- Через метод при каждом вызове
- Через окружающий контекст

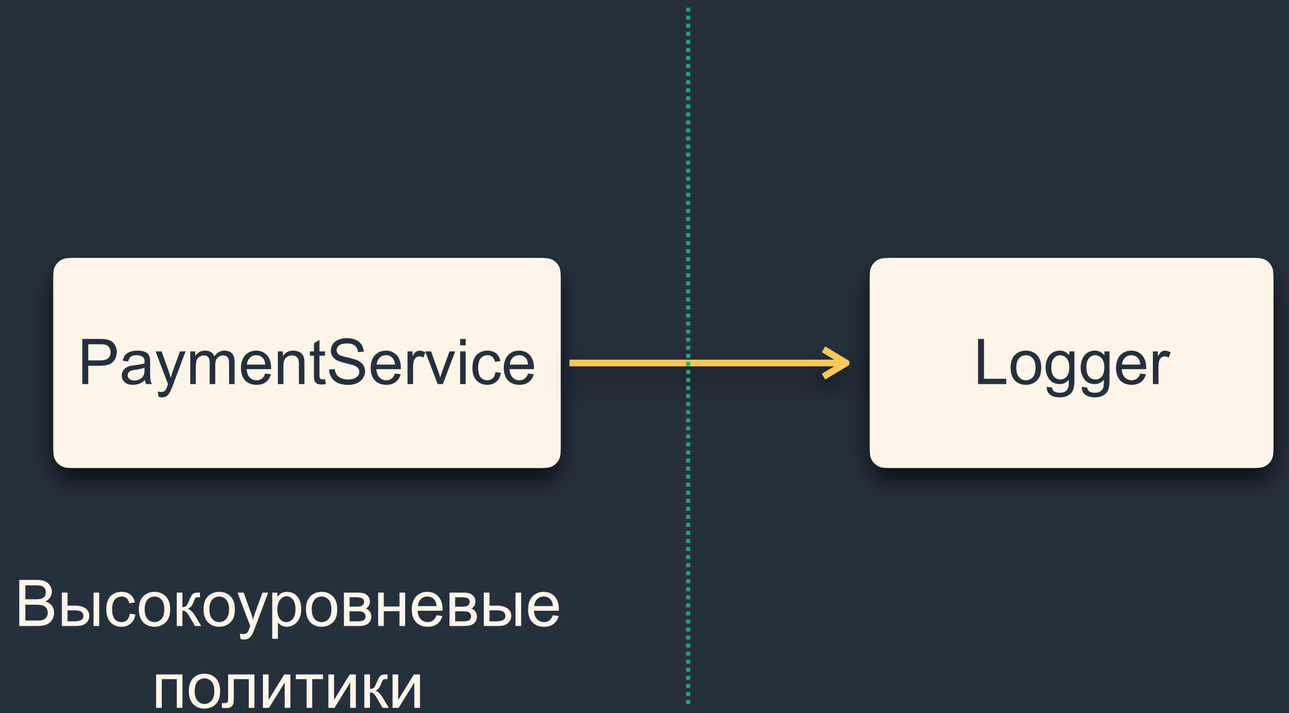
Пример 2,3,4

Антипаттер ServiceLocator

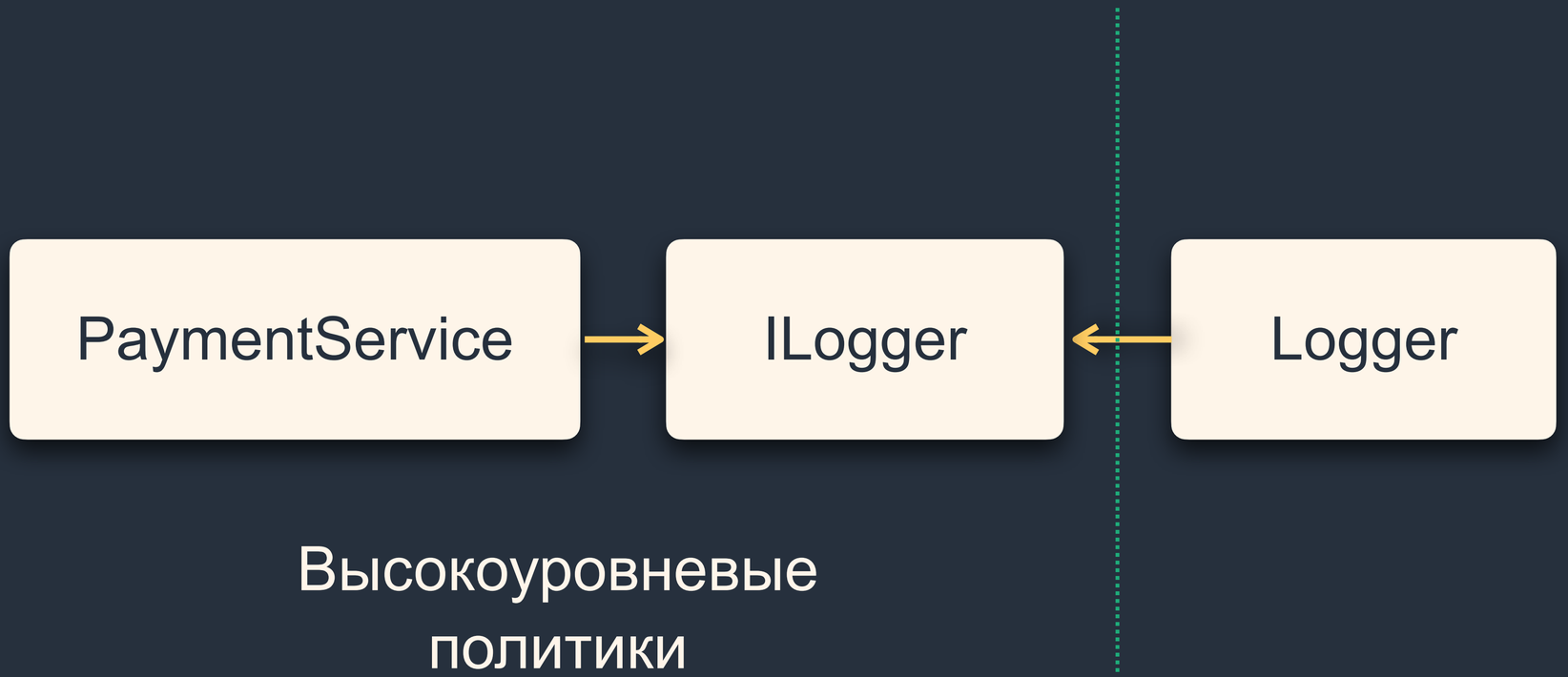
Пример 5

Inversion of Control (IoC) через DI

Развёрнутые зависимости



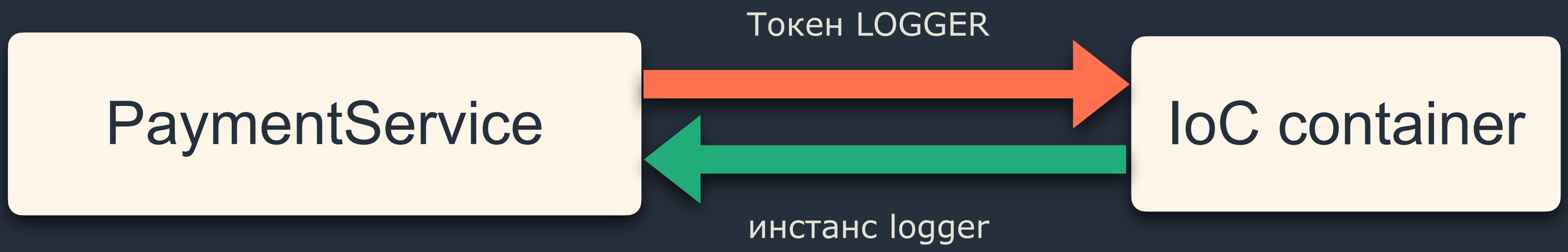
Развёрнутые зависимости



IoC-контейнеры

DI на декораторах, да с контейнером

```
export class PaymentService {
  constructor(@Inject(LOGGER) private readonly _logger: ILogger){ }
}
```



class PaymentService

class Logger

В контейнер можно положить всё, что нужно

Контейнер управляет жизненным циклом зависимости

Пример 6

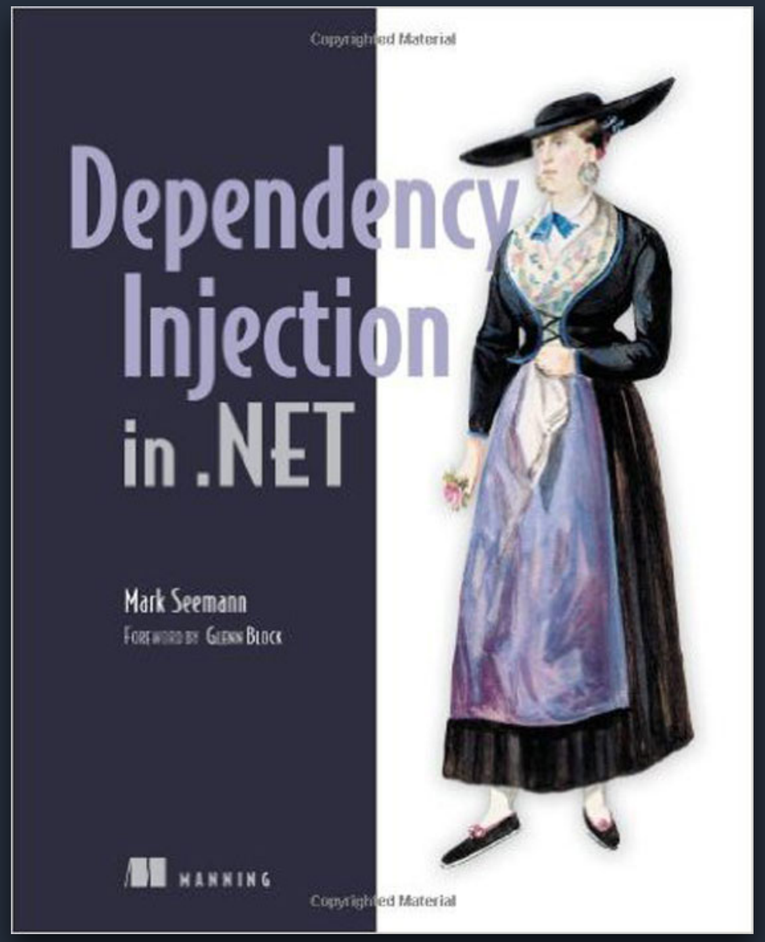
Composition Root

Диапазон жизненного цикла

- SINGLETON — одна копия на приложение
- REQUEST — новый инстанс на каждый запрос
- TRANSIENT — инстанцируются при каждой инъекции

Пример 7

Литература


Copyrighted Material
Dependency Injection in .NET
Mark Seemann
Foreword by Glenn Block
MANNING
Copyrighted Material