О СЕБЕ

Меня зовут Илья, я занимаюсь разработкой с использованием .Net в компании Luxoft.

efimilya@gmail.com

+7 910 900 4727

Сложность разрабатываемых систем и количество унаследованного кода заставляют использовать инструменты и подходы.

IoC/DI в этом очень помогает.

IoC/DI на примере Autofac

ВВЕДЕНИЕ (ЗАЧЕМ НУЖЕН IOC)

Объектно-ориентированный дизайн включает

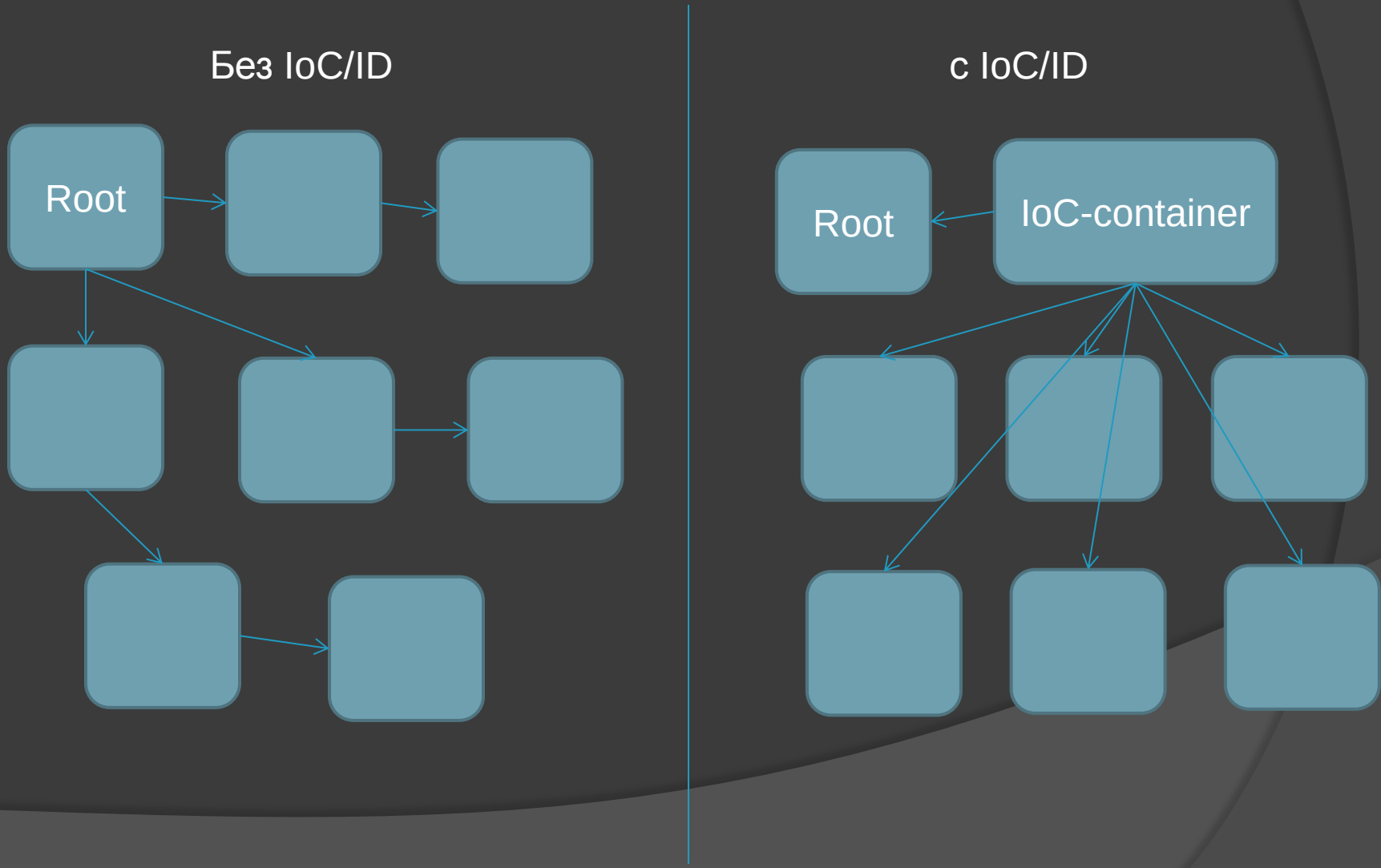
1. Создание экземпляра
2. Передача его потребителю
3. Управление временем жизни
4. Бизнес-логика
5. Очистка ненужного экземпляра

IoC-контейнер берет на себя 1, 2, 3, 5

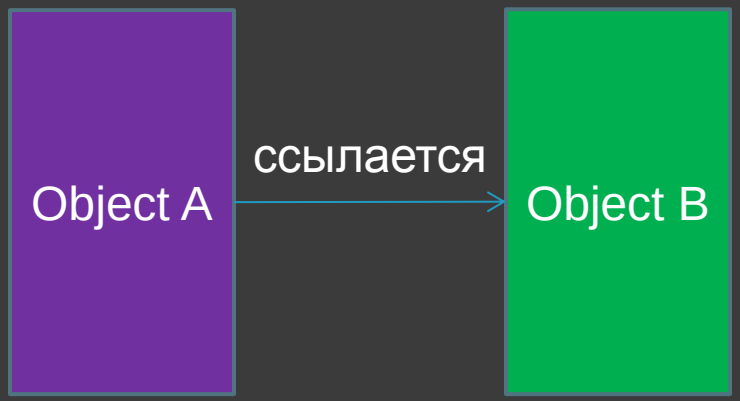
Вопросы для обсуждения.

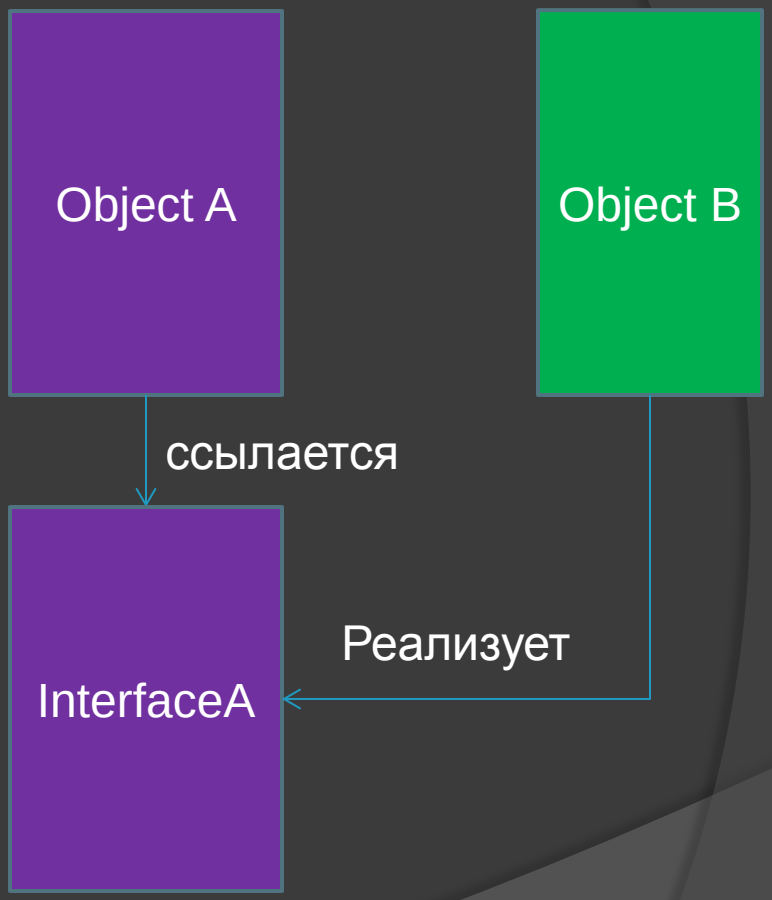
1. Что такое IoC? В чем инверсия?
2. В чем преимущества IoC/DI?
3. Как это связано с буквой D из SOLID?
4. Что такое вложенные контейнеры?
5. IDisposable и IoC-контейнеры.
6. Автоматические фабрики.

1. Что такое IoC? В чем именно здесь заключается инверсия?
2. В чем преимущества.



3. КАК ЭТО СВЯЗАНО С ПОСЛЕДНЕЙ БУКВОЙ D ИЗ SOLID?



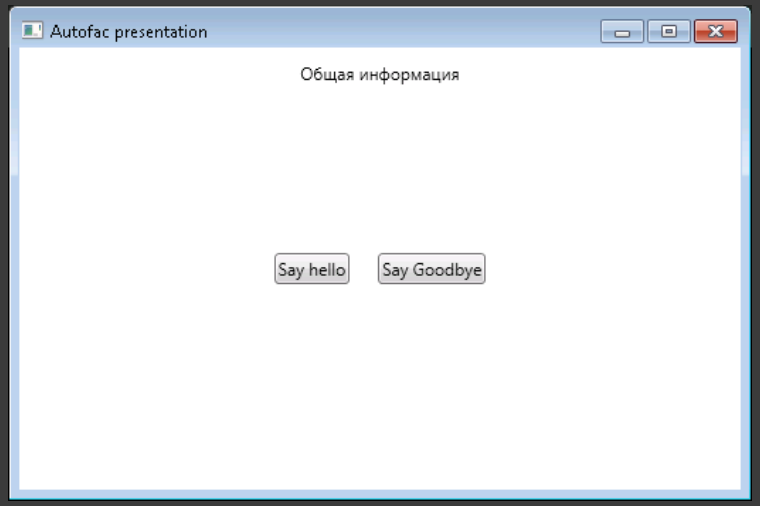


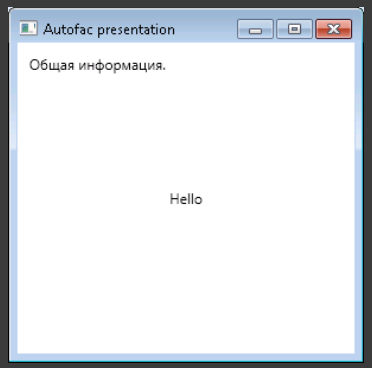
4. Что такое механизм вложенных (child) контейнеров?

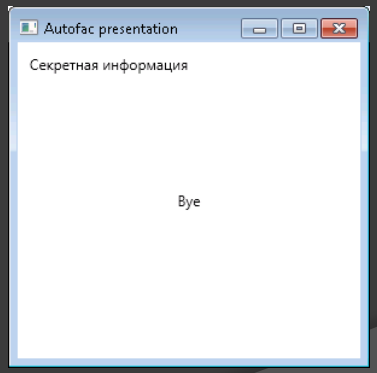
- Регистрация зависимостей в отдельных контейнерах.
- Которые могут брать зависимости из родительских контейнеров.
- Регистритрация зависимостей, параметризованных данными времени выполнения.
- Поэтому не придется обходиться с этими данными вручную (передавать в качестве параметров).
- Компоненты не могут иметь доступ друг к другу во время выполнения.
- Можно освободить ресурсы, вызвав Dispose вложенного контейнера.

4. ЧТО ТАКОЕ МЕХАНИЗМ ВЛОЖЕННЫХ (CHILD) КОНТЕЙНЕРОВ?

Задача для фронэнда:







4. Что такое механизм вложенных (child) контейнеров?

Первое решение.

- Вся регистрация в одном контейнере.
- Создание экземпляр дочернего окна явно через конструктор.
- Передача ему нужных зависимостей (исходя из введенных пользователем данных).

5. Что такое механизм вложенных (child) контейнеров?

Плюсы:

- Мы решили задачу
- Зависимости (за исключение одной) мы разрешили через контейнер.

4. Что такое механизм вложенных (child) контейнеров?

Минусы:

- Одну из зависимостей создана через new.
- Противоречит IoC/DI.
- Не сможем использовать очистку.
- Если увеличится количество компонентов, увеличится сложность решения.

4. ЧТО ТАКОЕ МЕХАНИЗМ ВЛОЖЕННЫХ (CHILD) КОНТЕЙНЕРОВ?

Второе решение

- Использование вложенных скоупов.
- Регистрация зависимостей в отдельных контейнерах.
- Вложенный контейнер конфигурируются на момент выполнения.

4. ЧТО ТАКОЕ МЕХАНИЗМ ВЛОЖЕННЫХ (CHILD) КОНТЕЙНЕРОВ?

Минусы:

- На первый взгляд это может показаться довольно сложным и запутанным приемом (издержки подхода)

4. Что такое механизм вложенных (child) контейнеров?

Плюсы:

- Нет ни одного оператора new.
- Простая регистрация и разрешение зависимостей.

4. ЧТО ТАКОЕ МЕХАНИЗМ ВЛОЖЕННЫХ (CHILD) КОНТЕЙНЕРОВ?

Задача усложнилась.

- Добавились новые требования.
- Стало больше компонентов.
- Контекстно-зависимые компоненты располагаются глубоко относительно ChildWindowViewModel
- Поэтому создание ее экземпляра становится сложным.

4. Что такое механизм вложенных (child) контейнеров?

Первому решению плюсов не добавилось. Минусы усугубились:

```
builder.Register<ChildWindowViewModelFactory>(
            context =>
            {
                var parentContext = context.Persist();
                return speaker =>
new ChildWindowViewModel(
GetSpeaker(speaker, parentContext), new ChildHeaderViewModel(
new HeaderTextFormatter(
GetFormatHeaderStrategy(speaker, parentContext)),
parentContext.Resolve<HeaderTextProvider>()));
            });
```

4. Что такое механизм вложенных (child) контейнеров?

- Второе решение не усложнилось
- Добавилась всего-лишь регистрация новых компонентов

4. Что такое механизм вложенных (child) контейнеров?

Задача для бекэнда:

- Обслужить запрос пользователя.
- А именно: загрузить исходные данные.
- На их основе произвести вычисления.

4. Что такое механизм вложенных (child) контейнеров?

Первое решение.

- Регистрация всех компонентов в одном контейнере.
- Передача зависимостей через параметры методов.

4. Что такое механизм вложенных (child) контейнеров?

Плюсы:

- Задача решена.
- Выглядит несложно.

4. ЧТО ТАКОЕ МЕХАНИЗМ ВЛОЖЕННЫХ (CHILD) КОНТЕЙНЕРОВ?

Минусы:

- Приходится часто передавать параметры методам.

```
public int Calculate(int customerId, int programId, string operationName)
{
        var customer = _customerRepository.Get(customerId);
        var program = _programRepository.Get(programId);
        var operation = _operationRepository.Get(operationName);

        return _firstStepAggregator.Aggregate(customer, program, operation) +
            _secondStepAggregator.Aggregate(customer, program, operation);
}

 public int Aggregate(Customer customer, Program program, Operation operation)
 {
        return (_customerHandler.Handle(customer) +
            _programHandler.Handle(program) +
            _operationHandler.Handle(operation)) * _globalServiceData.Value;
 }
```

4. Что такое механизм вложенных (child) контейнеров?

Второе решение

- Использовать вложенные контейнеры.
- Контейнер занимается передачей зависимостей.

4. ЧТО ТАКОЕ МЕХАНИЗМ ВЛОЖЕННЫХ (CHILD) КОНТЕЙНЕРОВ?

Минусы

- Регистрация чуть сложнее (издержки подхода)

4. Что такое механизм вложенных (child) контейнеров?

Плюсы:

- Данные, которые мы передавали в качестве параметров, теперь зарегестрированы в контейнере и получать к ним доступ очень просто.
- Методы стали намного проще по сигнатуре.

```
public int Calculate()
{
        return _firstStepAggregator.Aggregate() +
            _secondStepAggregator.Aggregate();
}

public int Aggregate()
{
        return (_customerHandler.Handle() +
            _programHandler.Handle() +
            _operationHandler.Handle()) * _globalServiceData.Value;
}
```

5. КАК КОНТЕЙНЕРЫ ОБРАБАТЫВАЮТ ЗАВИСИМОСТИ, РЕАЛИЗУЮЩИЕ IDISPOSABLE?

- Порой нужна очистка.
- Контейнер умеет вызывать Dispose() зависимостей.
- Удобно использовать CompositeDisposable и Disposable<T>

5. КАК КОНТЕЙНЕРЫ ОБРАБАТЫВАЮТ ЗАВИСИМОСТИ, РЕАЛИЗУЮЩИЕ IDISPOSABLE?

Disposable<T>

принимает два параметра:

- T value
- Action disposeAction

5. КАК КОНТЕЙНЕРЫ ОБРАБАТЫВАЮТ ЗАВИСИМОСТИ, РЕАЛИЗУЮЩИЕ IDISPOSABLE?

CompositeDisposable

- Контейнер для элементов IDisposable.
- Очистка нескольких компонентов за раз.
- Удобно, например, для отписки от событий.

6. АВТОМАТИЧЕСКИЕ ФАБРИКИ

- Func<int, string, … , T>
- Создание экземпляров, которые на вход принимают загерестрированные зависимости, а также произвольные параметры.
- Экземпляр создан через контейнер, а значит мы получаем все преимущества контейнера (передача зависимостей, управление временем жизни и очистка).

ЗАКЛЮЧЕНИЕ

- IoC/DI делает часть работы за нас.
- Помогает упростить объектно-ориентированный дизайн.
- Autofac поддерживает вложенные контейнеры.
- А также автоматические фабрики.

МАТЕРИАЛЫ

- Книга Марка Симана - внедрение зависимостей в .net
- Документация Autofac
- Исходный код презентации
- Статья Удобное создание Composition Root с помощью Autofac
- Статья Самая простая и надежная реализация Статья шаблона проектирования Dispose
- Статья Disposable без границ