Урок 12.
IoC фреймворки.

Spring

Гарбузов Николай
Donriver Inc.

DonRiver

Agenda

- Введение в IoC фреймворки
- Spring фреймворк, способы его конфигурации
- Тестирование в Spring

Введение в IoC фреймворки

Класс состоит из полей и методов

DonRiver

Введение в IoC фреймворки

Класс состоит из полей и методов

в реальной жизни от класса мало пользы,
если он не использует(ся) другие классы

Введение в IoC фреймворки

Класс состоит из полей и методов

в реальной жизни от класса мало пользы,
если он не использует(ся) другие классы

**но если все классы знают друг о друге,
то такую систему очень (ОЧЕНЬ) тяжело
модифицировать**

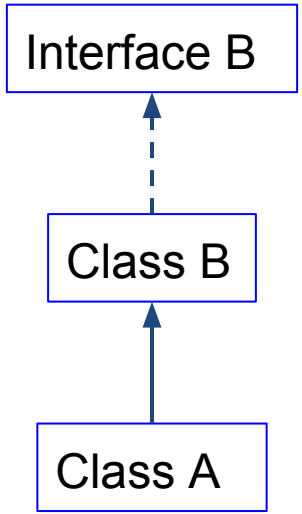
Введение в IoC фреймворки

Q:
Но если классы используют друг друга,
то они должны знать друг о друге, правда?

A:
Нет, не правда

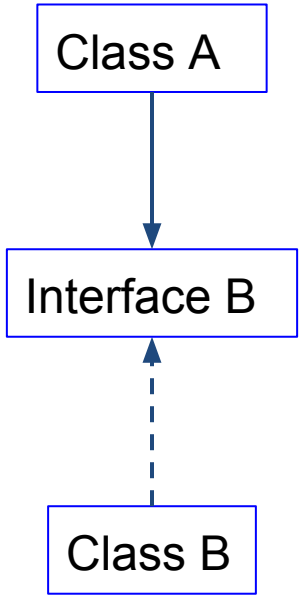
1. Они могут знать об интерфейсах друг друга
2. Конкретные реализации могут создаваться ВНЕ этих классов!

Введение в IoC фреймворки

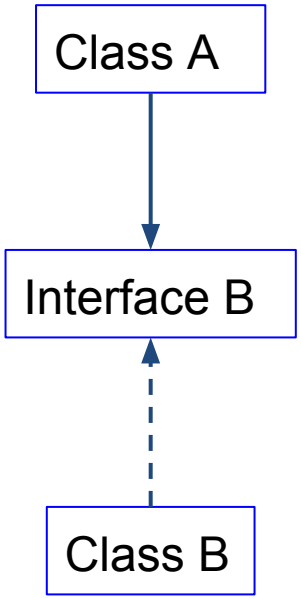


DonRiver

Введение в IoC фреймворки



Введение в IoC фреймворки



Профит:

Класс A **не зависит** от деталей реализации интерфейса B (т.е. от класса B)

Это DI - Dependency Inversion

DonRiver

Введение в IoC фреймворки

Q:
Но если классы используют друг друга,
то они должны знать друг о друге, правда?

A:
Нет, не правда

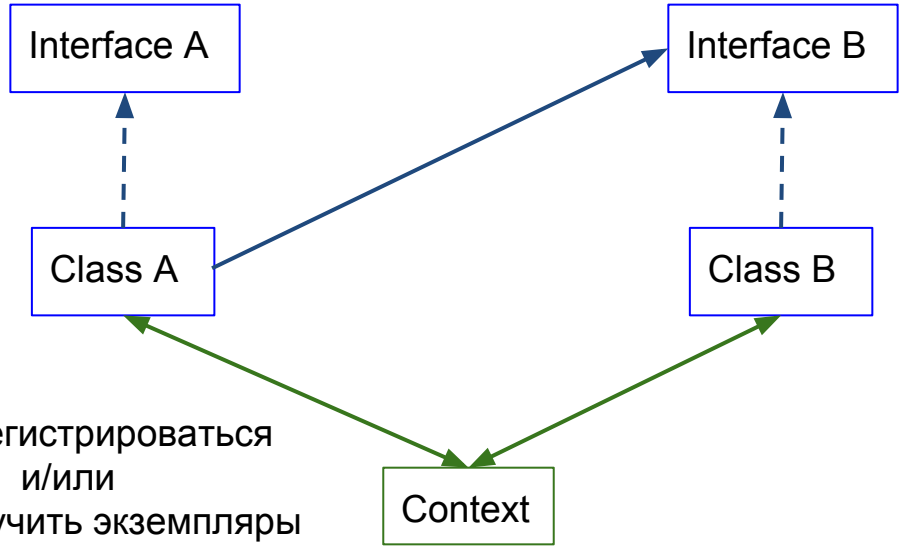
1. Они могут знать об интерфейсах друг друга
2. Конкретные реализации могут создаваться ВНЕ этих классов!

Введение в IoC фреймворки

Существует 2 метода, которые позволяют создавать реализации классов вне классов, которые используют их:

1. **Dependency Lookup** - реализации создаются некоторым механизмом и тогда, когда классы хотят их использовать, то обращаются САМИ к этому механизму

Введение в IoC фреймворки



- зарегистрироваться
  и/или
- получить экземпляры
  других классов

Введение в IoC фреймворки

Существует 2 метода, которые позволяют создавать реализации классов вне классов, которые используют их:

1. **Dependency Lookup** - реализации создаются некоторым механизмом и тогда, когда классы хотят их использовать, то обращаются САМИ к этому механизму

Паттерн **Фабрика** это пример Dependency Lookup

Введение в IoC фреймворки

Существует 2 метода, которые позволяют создавать реализации классов вне классов, которые используют их:

1. **Dependency Lookup** - реализации создаются некоторым механизмом и тогда, когда классы хотят их использовать, то обращаются САМИ к этому механизму

2. **Dependency Injection** - некоторый механизм создает реализации классов и САМ вставляет их в необходимые классы

Введение в IoC фреймворки

Существует 2 метода, которые позволяют создавать реализации классов вне классов, которые используют их:

1. **Dependency Lookup** - реализации создаются некоторым механизмом и тогда, когда классы хотят их использовать, то обращаются САМИ к этому механизму

2. **Dependency Injection** - некоторый механизм создает реализации классов и САМ вставляет их в необходимые классы

DonRiver

Введение в IoC фреймворки

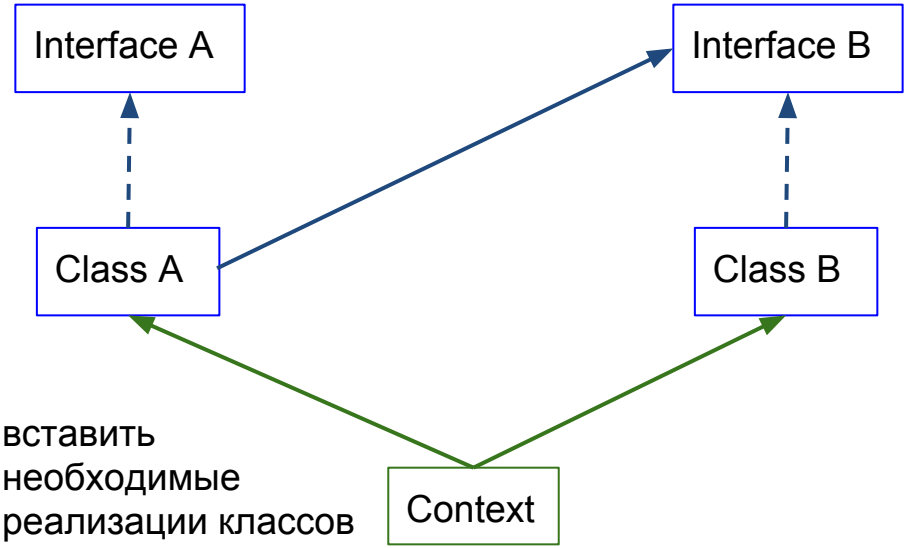
Dependency Injection (DI) - некоторый механизм создает реализации классов и сам вставляет их в необходимые классы

Процесс установки классов называется инъекция (injection)

- **Method injection** - инъекция через метод
- **Constructor injection** - инъекция через конструктор

DonRiver

Введение в IoC фреймворки



- вставить необходимые реализации классов

Введение в IoC фреймворки

Q:
Но если классы используют друг друга,
то они должны знать друг о друге, правда?

A:
Нет, не правда

1. Они могут знать об интерфейсах друг друга
2. Конкретные реализации могут создаваться ВНЕ этих классов!

IoC (Inversion of Control)
по сути и есть комбинация этих
2 утверждений!

Введение в IoC фреймворки

Самые распространенные IoC

- Pico
- Guice
- Spring
- Tapestry

Введение в IoC фреймворки

Самые распространенные IoC

- Pico
- Guice
- Spring
- Tapestry

Введение в IoC фреймворки

Spring это прежде всего IoC контейнер

Есть возможность описывать его конфигурацию с помощью

- XML
- Аннотаций
- Java классов
- Groovy скриптов (начиная с 4 версии)

Введение в IoC фреймворки

Spring это прежде всего IoC контейнер

Есть возможность описывать его конфигурацию с помощью

- XML
- Аннотаций
- Java классов
- Groovy скриптов (начиная с 4 версии)

Введение в IoC фреймворки

Spring XML-based конфигурация основывается на xml и может быть интегрирована с Annotation-based и с Java-based

Плюсы

- гибкая
- пассивная
  (классы не знают про DI)
- внешняя к классам
- не нужно компилировать

Минусы

- сложно поддерживать в больших проектах
- нельзя писать код (есть костыль в виде Spring SpEL)
- неявная

Введение в IoC фреймворки

Spring Annotation-based конфигурация основывается на аннотациях и используется только вместе с Xml-based и/или с Java-based

Плюсы

- более простая
- явная
- легче поддерживать

Минусы

- активная
  (классы знают о DI)
- нужно компилировать

Введение в IoC фреймворки

Spring Java-based конфигурация основывается на Java классах и может быть использована вместе с Xml-based и с Annotation-based

Плюсы

- более простая и явная
- легче поддерживать
- пассивная
- можно писать код
- внешняя по отношению к классам

Минусы

- нужно компилировать

Введение в IoC фреймворки

Экземпляры классов, находящиеся под управлением Spring называются **бинами**

- Бины различаются по области действия
  - Singleton - один на весь контейнер
  - Prototype - создается каждый раз при необходимости его вставки
  - Request - создается для запроса
  - Session - создается для сессии
  - Global-session - создается для глобальной сессии

Введение в IoC фреймворки

Экземпляры классов, находящиеся под управлением Spring называются **бинами**

- Бины различаются по области действия
  - Singleton - один на весь контейнер
  - Prototype - создается каждый раз при необходимости его вставки
  - Request - создается для запроса
  - Session - создается для сессии
  - Global-session - создается для глобальной сессии

Введение в IoC фреймворки

Spring это больше чем просто IoC, это целый фремворк для создания приложений и он включает в себя

Модули для работы

- с базой данных
- с веб-сервисами
- с безопасностью
- с тестированием
- и тп

Введение в IoC фреймворки

Spring это больше чем просто IoC, это целый фремворк для создания приложений и он включает в себя

Модули для работы

- с базой данных
- с веб-сервисами
- с безопасностью
- с тестированием
- и тп

Введение в IoC фреймворки

Spring test модуль позволяет использовать контекст Spring в юнит тестах и инжектить бины

Магия происходит благодаря

- @ContextConfiguration - указываем на конфигурацию Spring
- @RunWith - говорим jUnit запускаться с помощью Spring

Home Work

1. Добавить поддержку Spring в сетевой чат