Аспектно-ориентированное решение классических проблем

Гарбузов Николай
Donriver Inc.

DonRiver

Корень всего зла в программировании

Корень всего зла в программировании

Дублирование Кода

“Duplication may be the *root of all evil in software” - Robert C. Martin*

DonRiver

Пример из жизни

```
public long doCredit (Payment payment) {
   return bankService.transferMoney(payment);
}
```

Пример из жизни

```
public long doCredit (Payment payment) {
   MoneyTransfer moneyTransfer = objectConverter.toMoneyTransfer(payment);
   return bankService.transferMoney(moneyTransfer);
}
```

DonRiver

Пример из жизни

```
public long doCredit (Session session, Payment payment) {
   logger.logRequest(session, payment);

   MoneyTransfer moneyTransfer = objectConverter.toMoneyTransfer(payment);
   long transferredMoney = 0;
   try {
      transferredMoney = bankService.transferMoney(moneyTransfer);
   } catch (RuntimeException | Error ex) {
        logger.logFault(session, transferredMoney, ex);
        throw t;
   }
   logger.logReply(session, transferredMoney);
   return transferredMoney;
}
```

DonRiver

Пример из жизни

```
public long doCredit (Session session, Payment payment) {
   logger.logRequest(session, payment);
   businessObjectValidator.validate(payment);
   sessionService.validate(session);
   sessionThresholdService.process(session);

   MoneyTransfer moneyTransfer = objectConverter.toMoneyTransfer(payment);
   long transferredMoney = 0;
   try {
      transferredMoney = bankService.transferMoney(moneyTransfer);
   } catch (RuntimeException | Error ex) {
      logger.logFault(session, transferredMoney, t);
      throw t;
   }
   logger.logReply(session, transferredMoney);
   return transferredMoney;
}
```

DonRiver

Пример из жизни

```
public long doCredit (Session session, Payment payment) {
   logger.logRequest(session, payment);
   businessObjectValidator.validate(payment);
   sessionService.validate(session);
   sessionThresholdService.process(session);

   healthMonitoring.start(session, this);
   MoneyTransfer moneyTransfer = objectConverter.toMoneyTransfer(payment);
   long transferredMoney = 0;
   try {
      transferredMoney = bankService.transferMoney(moneyTransfer);
   } catch (RuntimeException | Error ex) {
      healChecking.stop(session, this, t); logger.logFault(session, transferredMoney, t);
      throw t;
   }
   healthMonitoring.stop(session, this); logger.logReply(session, transferredMoney);
   return transferredMoney;
}
```

DonRiver

Пример из жизни

```
public long doCredit (Session session, Payment payment) {
   logger.logRequest(session, payment);
   businessObjectValidator.validate(payment);
   sessionService.validate(session);
   sessionThresholdService.process(session);

   healthMonitoring.start(session, this);
   MoneyTransfer moneyTransfer = objectConverter.toMoneyTransfer(payment);
   long transferredMoney = 0;
   try {
      transferredMoney = bankService.transferMoney(moneyTransfer);
   } catch (RuntimeException | Error ex) {
      healChecking.stop(session, this, t); logger.logFault(session, transferredMoney, t);
      if (t instanceof ImportantException) {
         notificationService.notify(session, t);
      }
      throw t;
   }
   healthMonitoring.stop(session, this); logger.logReply(session, transferredMoney);
   return transferredMoney;
}
```

DonRiver

Пример из жизни

```
public long doCredit (Session session, Payment payment) {
   logger.logRequest(session, payment);
   businessObjectValidator.validate(payment);
   sessionService.validate(session);
   sessionThresholdService.process(session);

   healthMonitoring.start(session, this);
   MoneyTransfer moneyTransfer = objectConverter.toMoneyTransfer(payment);
   long transferredMoney = 0;
   try {
      transferredMoney = bankService.transferMoney(moneyTransfer);
   } catch (RuntimeException | Error ex) {
      healChecking.stop(session, this, t); logger.logFault(session, transferredMoney, t);
      if (t instanceof ImportantException) {
         notificationService.notify(session, t);
      }
      throw t;
   }
   healthMonitoring.stop(session, this); logger.logReply(session, transferredMoney);
   return transferredMoney;
}
```

DonRiver

Что же нам поможет?

АОП

* аспектно-ориентированное программирование

DonRiver

Сквозная функциональность

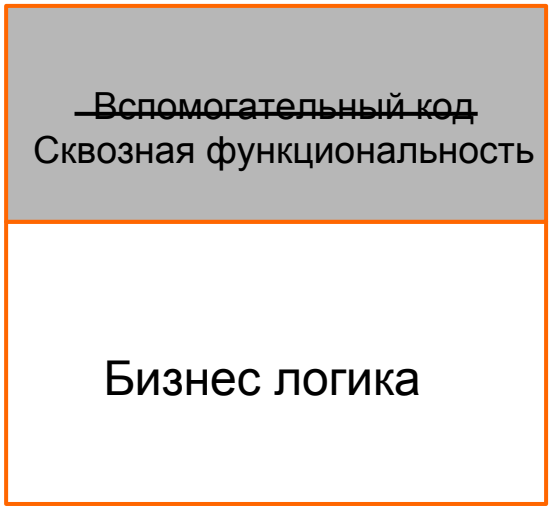
- Динамическая

изменяет поведение системы

- Статическая

изменяет структуру классов

Сквозная функциональность



- Проверка безопасности
- Преобразования данных
- Валидация данных
- Обработка исключений
- Транзакционность
- Логирование
- Мониторинг
- Синхронизация
- Кэширование
- Проверки уровня компиляции
- Изменение структуры классов

DonRiver

Элементы АОП

- Joinpoint (точка соединения) - выполнение/вызов метода, конструктора, доступ к полям и т.д.
- Pointcut (срез) - набор точек соединения с дополнительными инструкциями
- Advice (совет) - код, который выполнится в точках соединения, подпадающих под срез

- Aspect (аспект) - комбинация совета, среза и точек соединения

Элементы АОП

```
public aspect Example {
      before(String name) :
            call(public com.example.Service..* (String,..) && args(name,..)) {
            // do  smth
      }
}
```

Срез

Совет

```
@Aspect
public class Example {
  @Before("call(public com.example.Service..* (String,..))  && args(name,..)")
  public void process(String name) {
    //do smth
  }
}
```

* AspectJ синтаксис

Реализации АОП

- Spring AOP
- AspectJ

Реализации АОП

- Spring AOP
- AspectJ
- Nanning aspects
- Java Aspect Components (JAC)
- DemeterJ
- jBoss AOP
- AspectWerkz

Spring AOP

- Использует стандартный AspectJ синтаксис
- Динамическое связывание (Jdk Dynamic или CgLib прокси)

Spring AOP

- Использует стандартный AspectJ синтаксис
- Динамическое связывание (Jdk Dynamic или CgLib прокси)

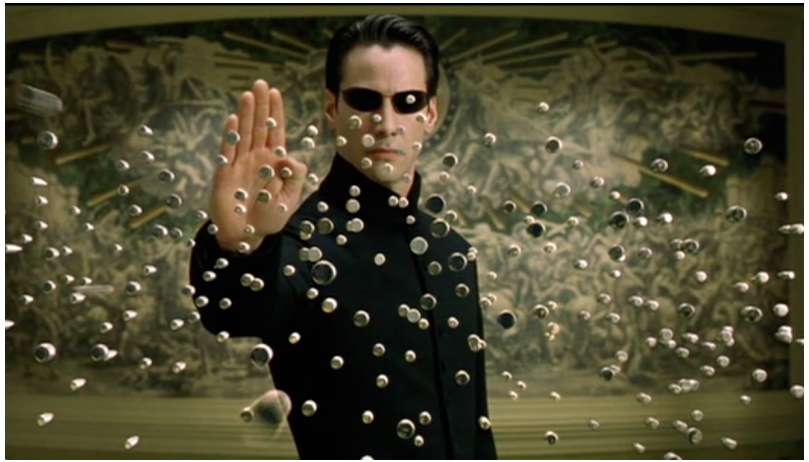
\+ Включается <aop:aspectj-autoproxy/>
\+ Не требует дополнительных библиотек
\+ Модули не обязаны зависеть от аспектов

\- Только динамический crosscutting
\- Только public методы
\- CgLib не поддерживает final методы
\- Нет полной поддержки всех типов срезов
\- Немного уменьшает производительность

AspectJ

LTW (load-time weaving)

CTW (compile-time weaving)



DonRiver

AspectJ LTW

- Динамическое связывание при загрузке классов (LoadTimeWeaver)

AspectJ LTW

- Динамическое связывание при загрузке классов (LoadTimeWeaver)

\+ Модули не обязаны зависеть от аспектов
\+ Внедрение аспектов в любые методы
\+ Поддержка всех динамических типов срезов

\- Требует aop.xml с описание всех аспектов
\- Требуется реализация LoadTimeWeaver (или java агент)
\- Увеличивает время загрузки классов

AspectJ CTW

- Статическое связывание при компиляции классов
- Использует ajc компилятор

AspectJ CTW

- Статическое связывание при компиляции классов
- Использует ajc компилятор

\+ Поддержка динамической и статической сквозной функциональности
\+ Внедрение аспектов в любые методы
\+ Поддерживает изменения при пост-компиляции
\+ Поддержка всех типов срезов

\- Требует aspectj-maven-plugin
\- Увеличивает время компиляции
\- Модули обязаны зависеть от аспектов

В итоге

	Spring AOP	AspectJ LTW	AspectJ CTW
Связывание	Динамическое	Динамическое	Статическое
Сквозная функциональность	Динамическая	Динамическая	Динамическая и Статическая
Увеличивает время	Выполнения кода	Загрузки классов	Компиляции
Сложность внедрения в проект	Низкая	Высокая	Средняя
Объем решаемых задач	Небольшой	Средний	Большой

Ресурсы

- AspectJ in Action

http://www.manning.com/laddad2/

- Spring AOP

http://docs.spring.io/spring/docs/current/spring-framework-reference/html/aop.html

- AOP example project

https://github.com/m91snik/AOP

Вопросы ?

DonRiver