Spring автозвязывание. АОП

№ урока: 10 **Курс:** Основы Java EE

Средства обучения: InteliJ Idea

Обзор, цель и назначение урока

Авто связывание в Spring. Аннотации конфигурации. Аспектно-ориентированное программирование (АОП).

Изучив материал данного занятия, учащийся сможет:

- Изменить конфигурацию бина, используя аннотации
- Сделать автоматическую подстановку бина
- Разделить основной функционал бизнес логики и «сквозной» функционал с помощью АОП

Содержание урока

1. Повторения Basic Bean Cofiguration
2. Понятия Autowiring.
3. Конфигурация с помощью аннотаций.
4. Пример Swing проекта на Spring.
5. Знакомство с АОП.
6. Основные определения АОП
7. Типы советов в АОП
8. Анализ скорости работы методов с помощью АОП
9. АОП с помощью аннотаций
10. Срез интерфейсов с помощью АОП

Резюме

- **Аспектно-ориентированное программирование, АОП** (англ. Aspect-oriented programming, AOP) - парадигма программирования, которая позволяет выделить перекрестную (сквозную) функциональность (cross-cutting concern).
  Современные программные системы часто решают огромное количество сложнейших задач, требующих хороших инженерных навыков от их разработчиков и надежности инструментальных средств разработки. При росте сложности таких систем растет и программный код, разработчику становится все труднее охватить все детали реализации системы. При поддержке крупных программных средств возрастает время нахождения и исправления ошибки, усложняется добавления новых характеристик, поскольку становится все труднее определить насколько изменения повлияют на систему, не внесут дополнительных ошибок и дефектов. Для решения таких задач применяют различные инженерные средства, такие как многофункциональные среды разработки, шаблоны проектирования, готовые программные каркасы и тому подобное.
  Часто упоминаемым недостатком объектно-ориентированного подхода является невозможность локализации сквозной функциональности в одном классе. В качестве примера такой функциональности часто называют необходимость ведения журналов событий, управления исключительными ситуациями, проверку прав доступа. Код, отвечающий за данную функциональность, часто разбросан по разным классам. Это, с одной стороны, не позволяет сконцентрировать внимание на основной бизнес-логике класса и затрудняет чтение кода. С другой стороны, усложняется внесении изменений в методы работы сквозной функциональности, не всегда можно исправить правильным использованием интерфейсов или шаблонов проектирования. Сейчас аспектно-ориентированный подход часто используют для реализации вышеприведенных примеров, однако, как отмечают некоторые авторы, в этом

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

сфера применения аспектно-ориентированного подхода не ограничиваются, поскольку он может быть использован для проектирования любых систем, содержащих сквозную функциональность.

В результате наличия лишней перекрестной функциональности разрабатываемый модуль содержит запутанный код, удовлетворяет различные программные требования. Отрицательные свойства такого кода:

Разбросан код. Поскольку сквозная функциональность затрагивает много модулей системы, то и вызовы этой функциональности будут разбросаны по всей системе. Например, если программа содержит средства мониторинга производительности работы с базой данных, то и вызовы этой функциональности будут размещены везде, где нужно работать с базой данных. Наличие разбросанных кода имеет негативное влияние на проектирование и реализацию системы;

Тяжесть отслеживания назначения модуля, поскольку он содержит одновременно функциональность для удовлетворения различных требований;

Сложность или невозможность повторного использования модуля в программах с другой сквозной функциональностью;

Велика вероятность ошибок. Наличие кода для реализации функциональности разного рода в одном модуле может привести к тому, что ни одно из заданий не получит достаточного внимания разработчика;

Тяжесть сопровождения. Если появляется необходимость в изменении сквозной функциональности, такое изменение затрагивает много модулей системы, в конечном счете может привести к проблемам совместимости.

Для решения задачи локализации сквозной функциональности была разработана методология аспектно-ориентированного программирования (АОП). Основные идеи АОП были сформулированы идеологом методологии Г. Кинжалесом. Он также разработал самую популярную надстройку языка программирования Java для работы с аспектами - AspectJ.

- К основным понятиям аспектно-ориенованого программирования относятся:

  Аспект (англ. Aspect) - модуль или класс, реализующий сквозную функциональность. Аспект изменяет поведение другого кода, применяя советы в точках соединения, определенных некоторым срезом;

  Совет (англ. Advice) - дополнительная логика, код, который должен быть вызван из точки соединения. Совет может быть выполнена до, после или вместо точки соединения;

  Цель (англ. Target) - объект, к которому будут применяться советы;

  Точка соединения (англ. Join point) - точка в выполняемой программе (вызов метода, создание объекта, обращение к переменной), где следует применить совет;

  Срез (англ. Pointcut) - набор точек соединения. Срез определяет, подходит ли данная точка соединения к заданной советы;

  Внедрение (англ. Introduction) - изменение структуры класса и / или изменение иерархии наследования для добавления функциональности аспекта в инородный код;

  Переплетение (англ. Weaving) - связывание объектов с соответствующими аспектами (возможно на этапе компиляции, загрузки или выполнения программы).

- **Достоинства** аспектно-ориентированного подхода рассматривает программную систему как набор модулей, каждый из которых выражает особенность функционирования системы. При проектировании системы разработчик выбирает модули так, чтобы каждый из них реализовывал определенную функциональную требованию. Зато в рамках объектно-ориентированного подхода реализация некоторых требований к программе часто не может быть локализована в отдельном модуле, в результате чего код, отражающий такие функциональные задачи, будет находиться в разных модулях (например, код ведения журнала событий).

  Как подтверждают исследования, аспектно-ориентированный подход уменьшает сложность разрабатываемого кода. Традиционной характеристикой размера программ является количество строк исходного кода. Например, одной из таких метрик являются оценки Холстеда. Основу этой метрики составляют четыре измеряемые характеристики программы:

  - Число уникальных операторов программы;
  - Число уникальных операндов программы;
  - Общее число операторов в программе;
  - Общее число операндов в программе.

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Вторая наиболее информативна группа оценок сложности программ - метрики сложности потока управления программ. Как правило, с помощью этих оценок оперируют или плотностью руководящих переходов внутри программ или взаимосвязями этих переходов.

В результате проведенных исследований на основе разработки системы авторизации с использованием аспектно-ориентированного подхода установлено, что метрики кода в целом на 10-40% ниже в ООП реализации, что положительным образом влияет на систему, поскольку на каждую метрику (ресурс) будет потрачено меньше времени.

- Есть несколько причин, сдерживающих разработчиков от активного применения технологии аспектно-ориентированного программирования (хотя эти недостатки опровергаются в некоторых работах):
    - Недостаточная развитость языковых средств и конструкций, позволяющих запрограммировать аспекты;
    - Необходимость разработки лучших компиляторов для оптимизации аспектного кода
    - Отсутствие достаточно развитых средств подтверждения эффективности использования данного подхода в каждом конкретном случае;
    - Определенные сложности отладки и тестирования разработанных программ.

  Однако важнейшим сдерживающим фактором является необходимость формирования своеобразного мышления для проектирования систем в терминах сквозной функциональности. Поэтому аспектно-ориентированный подход не получил широкого распространения, однако есть перспективной технологией для изучения.
- **Использование** аспектно-ориентированного подхода не требует полного отказа от объектно-ориентированной реализации, поскольку его можно внедрять только частично. Более того, такой подход эффективно дополняет объектно-ориентированный код. Как показывают исследования аспектно-ориентированных программ, около 2% их кода связана со специфическими механизмами языка аспектно-ориентированного программирования (например, AspectJ) 12% - с базовыми механизмами; 86% является объектно-ориентированным.

  Именно поэтому существуют работы по совершенствованию программных каркасов (англ. Frameworks) с помощью технологии аспектов. Объектно-ориентированный фреймворк содержит компоненты, составляющие ядро функционирования, и компоненты, содержащие дополнительную функциональность. При использовании фреймворка стандартная функциональность расширяется с помощью подражания. Применение аспектно-ориентированного подхода позволяет, с одной стороны, разделить на отдельные модули сквозную функциональность ядра, с другой - легко добавлять функциональность используя аспекты ядра.

  Эффективно можно применять аспектно-ориентированное программирование для оптимизации шаблонов проектирования. Первой значительным преимуществом является способность локализовать код шаблона проектирования в одном аспекте или паре тесно связанных аспектов (в отличие от языка Java, где код шаблона может быть разбросанным по многим классам). Возможность видеть весь код в одном месте имеет ряд существенных преимуществ:

  Читатели кода могут легче понять шаблон, если он все находится в одном месте;

  Разработчики могут использовать понятные названия для описания аспекта, позволит другим разработчикам легче понять шаблон;

  Если возникла необходимость изменить реализацию шаблона, это можно сделать в одном месте, вместо того, чтобы искать реализацию по всей системе.

  Исследование использования аспектно-ориентированного подхода проводились в различных отраслях. Например, при разработке мобильных Java-игр можно оптимизировать управление игровым экраном, создание игровых персонажей, загрузки и отображения рисунков и тому подобное. Кроме того, с помощью точек соединения можно внедрять необязательную функциональность, например, отображение фоновых изображений только на устройствах определенного типа.

  Предложенные методы применения аспектно-ориентированного подхода для разработки много агентных систем, где агент - автономная программная единица, при выполнении задания реагирует на окружающую среду и общается с другими программами-агентами (например, программы покупка товаров в Интернете). В данном случае аспекты можно применить для проектирования нескольких платформ коммуникации агентов, дополнительных возможностей обучения, различных ролей и протоколов взаимодействия.

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Закрепление материала

- Что такое Autowiring?
- Как сконфигурировать Bean с помощью аннотаций?
- Что такое AOP?
- Какие основные понятия в AOP?
- Какие существуют типы советов в AOP?

Дополнительное задание

Задание
Начать читать раздел 7
Реализовать аспект для вывода Exception.
Создать метод деления чисел, в случае ошибки: выводит ошибку на экран.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные понятия, рассмотренные на уроке.
Прочитать документацию pdf про Annotation-based container configuration, главы 3.9, 3.10, 3.11

Задание 2
В проекте из заданий заменить реализацию на использование аннотаций (вместо XML).

Задание 3
Выучить определения AOP (на уровне ассоциаций)
К примеру 006_AOP добавить вывод названия папки. Выводить статистику по расширениям.

Задание 4
Из примера 007_AOP_Annotation выводить скорость выполнения только тех методов, которые помечены @ShowTime, выводить результаты в консоль только для тех методов, которые помечены @ShowResult. @ShowTime и @ShowResult создаем самостоятельно.

Задание 5
Прочитать главу 7.6
Из примера 008_AOP_Interface изменить код: выводить время работы только тех методов, которые возвращают тип Map
Разбить метод печати на 2 вида: первый печатает только Set, второй – только Map.

Рекомендуемые ресурсы

Spring docs pdf
https://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/pdf/spring-framework-reference.pdf

Annotations
https://docs.oracle.com/javase/tutorial/java/annotations/

Standart Annotations
https://www.sourceallies.com/2011/08/spring-injection-with-resource-and-autowired/

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com