Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет *Информационных технологий и программирования*

Образовательная программа *Компьютерные технологии*

Направление подготовки (специальность) *Прикладная математика и информатика 01.03.02*

О Т Ч Е Т

о **производственной практике**

Тема задания: *Создание консольных приложений на OSGi*

Обучающийся *Романенко Демьян Александрович, группа М33381*

Дата 16.04.2021

Санкт-Петербург
2021

Этап 1. Подготовительный

Я прочитал главы 1-4 книги “OSGi in Action”. В первой главе приводится общее описание модульной платформы OSGi, а также приводятся примеры кода. В следующих главах более подробно разобраны темы модульности, жизненного цикла бандла, сервисов и компонентов.

Платформа OSGi оперирует модулями, называемыми бандлами (bundle). Каждый bundle в OSGi – это логически завершённый программный компонент, реализующий определенный функционал.

OSGi – это стандарт OSGi alliance, а Apache Felix его имплементация. В него можно устанавливать различные бандлы – как компоненты Felix, так и пользовательские.

Ссылка на git-репозиторий:

https://github.com/DRomanenko/sw-practice/tree/master/stage1

Этап 2. Реализация OSGi-сервиса

В данном этапе я создал 2 бандла при помощи манифестов:

1. OSGi Hello Service – бандл, регистрирующий новый сервис приветствия пользователя. Содержит следующие классы:
   - Hello.java – интерфейс с методом для вывода приветствия;
   - HelloImpl.java – реализация интерфейса Hello;
   - HelloService.java – активатор бандла, в котором регистрируется сервис.

   И следующие пакеты:
   - com.github.dromanenko.stage2.service – экспортный пакет, содержащий интерфейс и его реализацию, а также активатор бандла.

   А также содержит следующие зависимости:
   - org.osgi.framework – необходим для работы бандла в среде OSGi.
2. OSGI Hello Client – бандл, получающий зарегистрированный предыдущим бандлом сервис, и вызывающий у объекта метод приветствия. Содержит следующие классы:
   - HelloClient.java – активатор бандла, получающий необходимый сервис и вызывающий метод-приветствие.

   И следующие пакеты:
   - com.github.dromanenko.stage2.client – пакет, содержащий активатор бандла.

А также содержит следующие зависимости:

- org.osgi.framework – необходим для работы бандла в среде OSGi.

Ссылка на git-репозиторий:

https://github.com/DRomanenko/sw-practice/tree/master/stage2

Этап 3. Apache Felix Service Component Runtime

Для корректной работы бандлов необходимо обновить следующие компоненты и бандлы:

- org.apache.felix.scr
- org.osgi.util.function
- org.osgi.util.promise

В данном этапе я создал 2 бандла:

1. OSGi Hello Service – бандл, регистрирующий при помощи аннотаций новый сервис приветствия пользователя. Содержит следующие классы:
   - Hello.java – интерфейс с методом для вывода приветствия;
   - Service.java – реализация интерфейса Hello и активатор бандла, в котором регистрируется декларативный сервис при помощи аннотации @Component.

   И следующие пакеты:
   - com.github.dromanenko.stage3.swpractice.service – экспортный пакет, содержащий интерфейс и его реализацию, а также активатор бандла;
   - com.github.dromanenko.stage3.swpractice.service.impl – пакет, содержащий реализацию интерфейса.

   А также содержит следующие зависимости:
   - org.osgi.framework – необходим для работы бандла в среде OSGi.
2. OSGI Hello Client – бандл, получающий зарегистрированный предыдущим бандлом сервис, и вызывающий у объекта метод приветствия. Содержит следующие классы:
   - Client.java – активатор бандла, получающий необходимый сервис при помощи аннотации @Reference и вызывающий метод-приветствие.

   И следующие пакеты:
   - com.github.dromanenko.stage3.swpractice.client – пакет, содержащий активатор бандла.

А также содержит следующие зависимости:

- com.github.dromanenko.stage3.swpractice.service
- org.osgi.framework – необходим для работы бандла в среде OSGi.

Ссылка на git-репозиторий:

https://github.com/DRomanenko/sw-practice/tree/master/stage3

Этап 4. Создание собственной команды для Apache Felix Gogo

В данном этапе я создал бандл OSGi Hello command, который регистрирует новую команду в Apache Felix Gogo. Содержит следующие классы:

- Hello.java – интерфейс с методом для вывода приветствия;
- HelloCommand.java – реализация интерфейса Hello;
- HelloActivator.java – активатор бандла, в котором регистрируется сервис.

И следующие пакеты:

- com.github.dromanenko.swpractice.stage4.client – экспортный пакет, содержащий интерфейс и его реализацию, а также активатор бандла.

А также содержит следующие зависимости:

- org.osgi.framework – необходим для работы бандла в среде OSGi.

Команда "practice:hello" принимает любое количество аргументов – выводит “Hello, <List args>”. Способ вызова можно посмотреть через вызов команды без аргументов.

Ссылка на git-репозиторий:

https://github.com/DRomanenko/sw-practice/tree/master/stage4

Этап 5. Создание приложения

Для корректной работы бандлов необходимо обновить и установить следующие компоненты и бандлы:

- org.apache.felix.scr
- org.osgi.util.function
- org.osgi.util.promise
- com.google.code.gson

В данном этапе я создал 5 бандлов, связанных при помощи аннотаций:

1. OSGi Base Service – бандл, содержащий интерфейс для обработки заголовков новостей, общие служебные дата-классы для GSON и абстрактный класс с общим кодом. Содержит следующие классы:

- BaseNewsService.java – абстрактный класс с общими методами для получения данных по URL и их парсинга при помощи GSON.
- Data.java – дата-класс верхнего уровня для списка заголовков новостей;
- DataNews.java – дата-класс нижнего уровня для заголовка новости;
- NewsService.java – интерфейс для обработки заголовков.

Данные классы находятся в пакете:

- com.github.dromanenko.swpractice.stage5.baseservice – экспортный пакет.

А также содержит следующие зависимости:

- com.google.code.gson – необходим для работы парсера JSON;
- org.osgi.framework – необходим для работы бандла в среде OSGi.

2. OSGi Lenta Service – бандл, отвечающий за обработку заголовков новостей с сайта https://lenta.ru. Содержит следующие классы:
   - LentaNewsService.java – класс, расширяющий функционал BaseNewsService и реализующий интрефейс NewsService, обрабатывающий новостные заголовки. Будет подключен далее, при помощи аннотации @Component.

   Данный классы находятся в пакете:

   - com.github.dromanenko.swpractice.stage5.lentaservice – экспортный пакет.

   А также содержит следующие зависимости:

   - com.github.dromanenko.swpractice.stage5.baseservice;
   - com.google.code.gson – необходим для работы парсера JSON;
   - org.osgi.framework – необходим для работы бандла в среде OSGi.
3. OSGi Aif Service – бандл, отвечающий за обработку заголовков новостей с сайта https://aif.ru. Содержит следующие классы:
   - AifNewsService.java – класс, расширяющий функционал BaseNewsService и реализующий интрефейс NewsService, обрабатывающий новостные заголовки. Будет подключен далее, при помощи аннотации @Component.

   Данный классы находятся в пакете:

   - com.github.dromanenko.swpractice.stage5.aifservice – экспортный пакет.

   А также содержит следующие зависимости:

   - com.github.dromanenko.swpractice.stage5.baseservice;
   - com.google.code.gson – необходим для работы парсера JSON;

- org.osgi.framework – необходим для работы бандла в среде OSGi.

4. OSGi Meduza Service – бандл, отвечающий за обработку заголовков новостей с сайта https://meduza.io. Содержит следующие классы:
   - DataMeduza.java – дата-класс верхнего уровня для отображения из служебной ссылки в заголовки новостей;
   - MeduzaNewsService.java – класс, расширяющий функционал BaseNewsService и реализующий интрефейс NewsService, обрабатывающий новостные заголовки. Будет подключен далее, при помощи аннотации @Component.

   Данные классы находятся в пакете:
   - com.github.dromanenko.swpractice.stage5.meduzaservice – экспортный пакет.

   А также содержит следующие зависимости:
   - com.github.dromanenko.swpractice.stage5.baseservice;
   - com.google.code.gson – необходим для работы парсера JSON;
   - org.osgi.framework – необходим для работы бандла в среде OSGi.
5. OSGi Command Service – бандл, который регистрирует новую команду в Apache Felix Gogo. Содержит следующие классы:
   - NewsCommand.java – интерфейс с методом для вывода статистики;
   - NewsCommandService.java – реализация интерфейса NewsCommand. Необходимые зависимости подключаются с помощью аннотации @Reference.

   Данные классы находятся в пакете:
   - com.github.dromanenko.swpractice.stage5.commandservice – экспортный пакет.

   А также содержит следующие зависимости:
   - com.github.dromanenko.swpractice.stage5.baseservice;
   - com.github.dromanenko.swpractice.stage5.lentaservice;
   - com.github.dromanenko.swpractice.stage5.aifservice;
   - com.github.dromanenko.swpractice.stage5.meduzaservice;
   - com.google.code.gson – необходим для работы парсера JSON;
   - org.osgi.framework – необходим для работы бандла в среде OSGi.

Команда "news:stats" принимает один аргумент – название новостного источника для вывода статистики. Способ вызова и список зарегистрированных новостных источников можно посмотреть через вызов команды без аргументов.

Ссылка на git-репозиторий:

https://github.com/DRomanenko/sw-practice/tree/master/stage5

Выводы

OSGi - это стандарт, используемый для создания модульных приложений. Он добавляет новый уровень модульности - bundles (также известные как components, modules). Каждый модуль реализовывает определенный функционал, а сами модули слабо связаны.

Преимущества:

+ Модульность – снижение сложности, инкапсуляция, управление зависимостями и легкость развертывания;
+ Динамические обновление;
+ Версионирование;
+ Ленивая активация модулей;

Недостатки:

- модульность с циклами;
- нет защиты от нарушения loading constraints;
- нет защиты деталей реализации от доступа через Reflection;
- активация бандлов зависит от схемы разрешения ссылок в JVM.

OSGi – это не только стандарт построения модульных приложений. Он также определяет среду, в которой пакеты существуют и запускаются. Это то, о чем вы должны знать - при использовании OSGi вы должны запускать свое приложение в специальной среде. В целом, поддержка модульности во время выполнения упрощает разработку модульного программного обеспечения, поскольку контейнер времени выполнения обеспечивает выполнение ваших конструкций во время разработки. Таким образом данную технологию целесообразно использовать для разработки модульных приложений.