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

Факультет информационных технологий и программирования

Образовательная программа программирование и интернет-технологии

Направление подготовки (специальность) 09.03.02 информационные системы и технологии

О Т Ч Е Т

о производственной практике

Тема задания: **Создание консольного приложения на OSGi**

Обучающийся: Сладков Михаил Михайлович, группа М3303

Дата: **02.07.2020**

Санкт-Петербург
2020

Этап 1. Подготовительный.

Мною были прочитаны главы 1-4 книги «OSGi in Action». В первой главе приводится общее описание модульной платформы OSGi. В последующих главах приводится более комплексный разбор «фич» данной платформы, а именно модульности, «жизненного цикла» бандла и сервисов.

Бандл (bundle) – OSGi-компонент, представляющий собой JAR-файл и дополнительную информацию (файл META-INF/MANIFEST.MF), которая используется средой OSGi.

В данной практике использовался Apache Felix – имплементация OSGi фреймворка и платформы сервисов. В него можно устанавливать различные бандлы – как компоненты Felix, так и пользовательские. Полный список по этапу 1 можно посмотреть ниже:

Id	Name	Version
0	System Bundle *(org.apache.felix.framework)*	6.0.3
1	jansi *(org.fusesource.jansi)*	1.17.1
2	JLine Bundle *(org.jline)*	3.7.0
3	Apache Felix Bundle Repository *(org.apache.felix.bundlerepository)*	2.0.10
4	Apache Felix Gogo Command *(org.apache.felix.gogo.command)*	1.0.2
5	Apache Felix Gogo JLine Shell *(org.apache.felix.gogo.jline)*	1.1.0
6	Apache Felix Gogo Runtime *(org.apache.felix.gogo.runtime)*	1.1.0
8	Apache Felix Web Management Console (All In One) *(org.apache.felix.webconsole)*	4.5.2.all
9	Apache Felix Servlet API *(org.apache.felix.http.servlet-api)*	1.1.2
10	Apache Felix EventAdmin *(org.apache.felix.eventadmin)*	1.5.0
11	Apache Felix Configuration Admin Service *(org.apache.felix.configadmin)*	1.9.16
12	Apache Felix Http Jetty *(org.apache.felix.http.jetty)*	4.0.18
13	Apache Felix Http API *(org.apache.felix.http.api)*	3.0.0
19	Java Portlet API V3.0 *(portlet-api)*	3.0.1

Рисунок 1. Список бандлов, установленных по этапу 1.

Этап 2. Реализация OSGi-сервиса.

По данному этапу я создал 2 бандла:

1. OSGI Hello Service – бандл, регистрирующий новый сервис по приветствию пользователя. Имеет следующие классы:
    - Activator.java – активатор бандла, в котором происходит регистрация сервиса;

- HelloService.java – интерфейс для взаимодействий, ответственный за вывод приветствия;
- HelloServiceImpl.java – реализация интерфейса HelloService;

Созданы следующие пакеты:

- ru.sladkov.task2service – экспортный пакет, содержащий активатор и интерфейс;
- ru.sladkov.task2service.implementation – приватный пакет, содержащий реализацию интерфейса;

Импортируется только зависимость org.osgi.framework, необходимая для работы бандла в среде OSGi.

2. OSGI Hello Client – бандл, получающий зарегистрированный предыдущим бандлом сервис, и вызывающий у объекта метод по приветствию.

   Имеет класс Activator.java, который получает необходимый сервис и вызывает приветствие. Он лежит в пакете ru.sladkov.task2client.
   Импортируются зависимости org.osgi.framework, а также ru.sladkov.task2service для запуска метода интерфейса.

Пример работы можно видеть ниже:

```
g! start 29
Hello-service started
g! start 30
Hello-client started
Hello OSGi World!
```

Ссылка на git репозиторий: https://github.com/Mixxx-a/ITMO-OSGi-SummerPractice2020/tree/master/SMM_Task2

Этап 3. Apache Felix Service Component Runtime

В первую очередь, были установлены компоненты SCR и необходимые дополнительные бандлы. Список можно видеть ниже:

22	Apache Felix Declarative Services *(org.apache.felix.scr)*	2.1.20
23	https://apache-mirror.rbc.ru/pub/apache//felix/org.apache.felix.scr.annotations-1.12.0.jar *(undefined)*	
24	https://apache-mirror.rbc.ru/pub/apache//felix/maven-scr-plugin-1.26.4.jar *(undefined)*	
25	org.osgi:org.osgi.util.promise *(org.osgi.util.promise)*	1.1.1.201810101357
26	org.osgi:org.osgi.util.function *(org.osgi.util.function)*	1.1.0.201802012106

Аналогично этапу 2, было создано 2 бандла:

1. OSGI Hello Annotation Service - бандл, регистрирующий с помощью аннотаций новый сервис по приветствию пользователя. Имеет следующие классы:
    - Activator.java – активатор бандла, выводящий информацию о старте программы;
    - Greeting.java – интерфейс для взаимодействий, ответственный за вывод приветствия;
    - GreetingImpl.java – реализация интерфейса Greeting, а также регистрация декларативного сервиса с помощью аннотации @Component;

    Созданы следующие пакеты:
    - ru.sladkov.task3service – экспортный пакет, содержащий активатор и интерфейс;
    - ru.sladkov.task3service.implementation – приватный пакет, содержащий реализацию интерфейса;

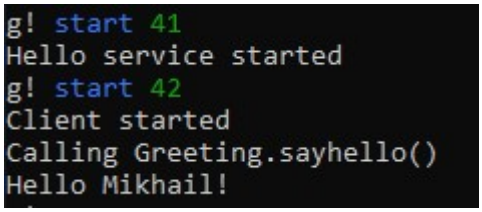
    Импортируется только зависимость org.osgi.framework.

2. OSGI Hello Annotation Clien – бандл, получающий зарегистрированный предыдущим бандлом сервис, и вызывающий у объекта метод по приветствию. Имеет следующие классы:
    - Activator.java – активатор бандла, выводящий информацию о старте программы;
    - Client.java – класс, получающий необходимый сервис с помощью аннотации @Reference и вызывающий метод по вывод приветствия;

    Эти классы лежат в пакете ru.sladkov.task3client.

    Импортируется зависимости org.osgi.framework и ru.sladkov.task3service

Пример работы можно видеть ниже:



Ссылка на git репозиторий: https://github.com/Mixxx-a/ITMO-OSGi-SummerPractice2020/tree/master/SMM_Task3

Этап 4. Создание собственной команды для Apache Felix Gogo

По данному этапу был создан бандл OSGI Hello command creation, который регистрирует новую команду в Gogo. Имеет следующие классы:

- Activator.java – активатор бандла, в котором происходит регистрация команды;
- Hello.java – интерфейс для взаимодействий, ответственный за вывод приветствия;
- HelloImpl.java – реализация интерфейса Hello;

Созданы следующие пакеты:

- ru.sladkov.task4command – экспортный пакет, содержащий активатор и интерфейс;
- ru.sladkov.task4command.implementation – приватный пакет, содержащий реализацию интерфейса;

Импортируется только зависимость org.osgi.framework.

Пример работы можно видеть ниже:

```
g! start 40
Command "practice:hello" started
g! practice:hello Михаил
Hello, Михаил
```

Ссылка на git репозиторий: https://github.com/Mixxx-a/ITMO-OSGi-SummerPractice2020/tree/master/SMM_Task4

Этап 5. Создание приложения

Данный проект состоит из 5-ти бандлов, которые связаны между собой с помощью аннотации:

1. Utility for title handler services – бандл, содержащий интерфейс для обработки заголовков новостей, а также вынесенные отдельно утилитарные методы. Содержит классы:
    - Activator.java – активатор бандла, выводящий информацию о старте программы;
    - HandlersUtility.java – утилитарные статические методы, необходимые уже в реализации;
    - TitleHandler.java – интерфейс по обработке заголовков;

    Эти классы лежат в пакете ru.sladkov.task5.utility и экспортируются.
    Импортируется зависимость org.osgi.framework.
2. Lenta API title handler service – бандл, отвечающий за обработку заголовков с новостного портала Lenta.ru. Имеет следующие классы:
    - Activator.java – активатор бандла, выводящий информацию о старте программы;

- LentaAPIHandler.java – обработчик заголовков, реализация интерфейса TitleHandler. Подключается с помощью аннотации @Component;

Примечание: В данной реализации используется сторонняя библиотека JSON.simple, поэтому бандл с нею необходимо предварительно установить в Felix.

Созданы следующие пакеты:

- ru.sladkov.task5.lenta_handler – содержит активатор;
- ru.sladkov.task5.lenta_handler.implementation – приватный пакет, содержащий реализацию интерфейса;

Импортируются зависимости:

- org.osgi.framework;
- ru.sladkov.task5.utility;
- org.json.simple;
- org.json.simple.parser;

3. Aif RSS title handler service – бандл, отвечающий за обработку заголовков с новостного портала АиФ. Имеет следующие классы:
   - Activator.java – активатор бандла, выводящий информацию о старте программы;
   - AiFRSSHandler.java – обработчик заголовков, реализация интерфейса; TitleHandler. Подключается с помощью аннотации @Component;

   Созданы следующие пакеты:
   - ru.sladkov.task5.aif_handler– содержит активатор;
   - ru.sladkov.task5.aif_handler.implementation – приватный пакет, содержащий реализацию интерфейса;

   Импортируются зависимости org.osgi.framework и ru.sladkov.task5.utility.
4. Gazeta XML title handler service – бандл, отвечающий за обработку заголовков с новостного портала Gazeta.ru. Имеет следующие классы:
   - Activator.java – активатор бандла, выводящий информацию о старте программы;
   - GazetaXMLHandler.java– обработчик заголовков, реализация интерфейса;
   - TitleHandler. Подключается с помощью аннотации @Component;

   Созданы следующие пакеты:
   - ru.sladkov.task5.gazeta_handler – содержит активатор;

- ru.sladkov.task5. gazeta _handler.implementation – приватный пакет, содержащий реализацию интерфейса;

Импортируются зависимости:

- org.osgi.framework;
- ru.sladkov.task5.utility;
- javax.xml.parsers;
- org.w3c.dom;

5. "news:stats" command creation – бандл, который регистрирует новую команду в Gogo и проводит её исполнение. Имеет следующие классы:
    - Activator.java – активатор бандла, выводящий информацию о старте программы;
    - TitleExplorer.java – интерфейс для команды;
    - TitleExplorerImpl.java – реализация интерфейса TitleExplorer. Необходимые зависимости подключаются с помощью аннотации @Reference;

    Созданы следующие пакеты:

    - ru.sladkov.task5.command– содержит активатор и интерфейс;
    - ru.sladkov.task5.command.implementation – приватный пакет, содержащий реализацию интерфейса;

    Импортируются зависимости:

    - org.osgi.framework;
    - ru.sladkov.task5.utility;
    - ru.sladkov.task5.lenta_handler;
    - ru.sladkov.task5.aif_handler;
    - ru.sladkov.task5.gazeta_handler;

Команда "news:stats" принимает 1 аргумент – название новостного портала для парсинга. Список зарегистрированных новостных порталов можно посмотреть вызовом данной командой без параметров. Пример работы можно видеть ниже:

```
g! news:stats
Доступные сайты для парсинга:
AiF
Gazeta
Lenta
all
g! stats Lenta
10 самых встречающихся слов в заголовках на портале Lenta длинной более 2 символов
Слово "россии" встретилось 10 раз
Слово "коронавируса" встретилось 5 раз
Слово "коронавирусом" встретилось 5 раз
Слово "россиянам" встретилось 4 раз
Слово "случаев" встретилось 4 раз
Слово "россиян" встретилось 4 раз
Слово "новых" встретилось 3 раз
Слово "видео" встретилось 3 раз
Слово "российском" встретилось 3 раз
Слово "число" встретилось 3 раз
```

Выводы.

Компонентная платформа OSGi обеспечивает модульность решения. Каждый модуль реализовывает определенный функционал, а сами модули слабо связаны.

OSGi обладает следующими преимуществами:

- Сниженная сложность разработки;
- Подключение модулей в систему как на этапе сборки приложения, так и во время работы приложения (run-time) (hotplugging);
- Разрешение зависимостей модулей с учётом версий;
- Управление модулями в run-time;
- Динамическое обновление;
- Механизм конфигурирования модулей;
- Улучшение безопасности за счёт дополнительной инкапсуляции public-методов;
- Легкая масштабируемость;
- Упрощенное развертывание;

Таким образом, OSGi определяет сервис-ориентированную платформу построения приложений с поддержкой модульности. На данной технологии целесообразно создавать приложения, которые состоят из отдельных компонентов.