

**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 ( <i>org.apache.felix.framework</i> )	6.0.3
1	jansi ( <i>org.fusesource.jansi</i> )	1.17.1
2	JLine Bundle ( <i>org.jline</i> )	3.7.0
3	Apache Felix Bundle Repository ( <i>org.apache.felix.bundlerepository</i> )	2.0.10
4	Apache Felix Gogo Command ( <i>org.apache.felix.gogo.command</i> )	1.0.2
5	Apache Felix Gogo JLine Shell ( <i>org.apache.felix.gogo.jline</i> )	1.1.0
6	Apache Felix Gogo Runtime ( <i>org.apache.felix.gogo.runtime</i> )	1.1.0
8	Apache Felix Web Management Console (All In One) ( <i>org.apache.felix.webconsole</i> )	4.5.2.all
9	Apache Felix Servlet API ( <i>org.apache.felix.http.servlet-api</i> )	1.1.2
10	Apache Felix EventAdmin ( <i>org.apache.felix.eventadmin</i> )	1.5.0
11	Apache Felix Configuration Admin Service ( <i>org.apache.felix.configadmin</i> )	1.9.16
12	Apache Felix Http Jetty ( <i>org.apache.felix.http.jetty</i> )	4.0.18
13	Apache Felix Http API ( <i>org.apache.felix.http.api</i> )	3.0.0
19	Java Portlet API V3.0 ( <i>portlet-api</i> )	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](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 ( <i>org.apache.felix.scr</i> )	2.1.20
23	▶ <a href="https://apache-mirror.rbc.ru/pub/apache//felix/org.apache.felix.scr.annotations-1.12.0.jar">https://apache-mirror.rbc.ru/pub/apache//felix/org.apache.felix.scr.annotations-1.12.0.jar</a> ( <i>undefined</i> )	
24	▶ <a href="https://apache-mirror.rbc.ru/pub/apache//felix/maven-scr-plugin-1.26.4.jar">https://apache-mirror.rbc.ru/pub/apache//felix/maven-scr-plugin-1.26.4.jar</a> ( <i>undefined</i> )	
25	▶ org.osgi:org.osgi.util.promise ( <i>org.osgi.util.promise</i> )	1.1.1.201810101357
26	▶ org.osgi:org.osgi.util.function ( <i>org.osgi.util.function</i> )	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 Client – 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.

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

Пример работы можно видеть ниже:

```
g! start 41
Hello service started
g! start 42
Client started
Calling Greeting.sayhello()
Hello Mikhail!
```

Ссылка на git репозиторий: [https://github.com/Mixxx-a/ITMO-OSGi-SummerPractice2020/tree/master/SMM\\_Task3](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](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 ней 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, который может быть "1", "2", "3" или "4". Соответственно, выводятся результаты по новостным порталам Lenta.ru, АиФ, Gazeta.ru или по всем сразу. Пример работы можно видеть ниже:

```
g! news:stats 1_
10 самых встречающихся слов в заголовках на портале Lenta.ru
Слово "в" встретилось 50 раз
Слово "на" встретилось 21 раз
Слово "по" встретилось 21 раз
Слово "с" встретилось 17 раз
Слово "россии" встретилось 16 раз
Слово "и" встретилось 10 раз
Слово "поправкам" встретилось 8 раз
Слово "коронавирусом" встретилось 8 раз
Слово "о" встретилось 7 раз
Слово "голосовании" встретилось 7 раз
g! news:stats 4
10 самых встречающихся слов в заголовках на порталах Lenta.ru, АиФ и Gazeta.ru
Слово "в" встретилось 290 раз
Слово "на" встретилось 133 раз
Слово "по" встретилось 82 раз
Слово "с" встретилось 71 раз
Слово "и" встретилось 62 раз
Слово "о" встретилось 57 раз
Слово "сша" встретилось 52 раз
Слово "россии" встретилось 51 раз
Слово "как" встретилось 42 раз
Слово "к" встретилось 31 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.