**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

об учебной практике

Тема задания: Создание консольного приложения на OSGi

Обучающийся *Куц Никита Денисович., № группы* М33071

Руководитель практики от профильной организации: *Сиротин Георгий Борисович, Системный архитектор, ООО «Софтверке»*

Руководитель практики от университета: Зубок Д.А., доцент ФИТИП, Университет ИТМО

Практика пройдена с оценкой \_\_\_\_

Дата 17.06.2022

Санкт-Петербург

2022

# Создание консольного приложения на OSGi

## Технологии

* Java SE 1.8+
* OSGi
* Apache Felix
* Apache Maven

## Цель задания

Познакомить студента с базовыми концепциями OSGi, попробовать эти концепции на примерах и с их использованием создать небольшое консольное приложение (этап 5).

## Описание

Практика состоит из 5 этапов. На первом этапе требуется познакомиться с OSGi, Apache Felix. На 2-4 этапах требуется создать несколько простых приложений, чтобы на практике познакомиться с основными концепциями. На 5-м этапе требуется создать небольшое, но полноценное приложение, применив знания и умения, полученные на этапах 1-4.

## Этапы выполнения задания

### Этап 1. Подготовительный

1. Прочитайте главы 1-4 «OSGi in Action». Постарайтесь понять основные концепции OSGi: modularity, lifecycle, bundles, services, components и т.п.
2. Скачайте и установите [Apache Felix](http://felix.apache.org/downloads.cgi). Установите Felix Web Console.
3. Запустите Apache Felix в режиме debug:  
   java -Xdebug -Xrunjdwp:transport=dt\_socket,server=y,suspend=n,address=<port-number> -jar bin/felix.jar

### Этап 2. Реализация OSGi-сервиса

1. Создайте бандл, содержащий сервис (интерфейс с одним методом и его реализацию). Зарегистрируйте сервис, используя активатор бандла. Метод должен выводить на консоль сообщение «Hello OSGi World!».
2. Установите бандл в Felix.
3. Создайте новый бандл. Бандл должен потреблять созданный ранее сервис. Метод сервиса, созданного на 1 шаге, должен вызываться при активации нового бандла.
4. Установите бандл в Felix. Убедитесь, что метод сервиса был вызван.

### Этап 3. Apache Felix Service Component Runtime

Аналогично этапу 2, создайте два бандла: который содержит реализацию сервиса и который использует этот сервис. Используйте [Apache Felix Service Component Runtime](http://felix.apache.org/documentation/subprojects/apache-felix-service-component-runtime.html), [SCR Annotations](http://felix.apache.org/documentation/subprojects/apache-felix-maven-scr-plugin/scr-annotations.html) и [Apache Felix Maven SCR Plugin](https://felix.apache.org/documentation/subprojects/apache-felix-maven-scr-plugin/apache-felix-maven-scr-plugin-use.html) для сборки.

### Этап 4. Создание собственной команды для Apache Felix Gogo

Требуется создать собственную команду «practice:hello» с одним параметром, которая при вызове печатает на консоль «Hello, <param>», где <param> - введенный пользователем параметр.

См. <http://coders-kitchen.com/2012/04/06/tutorial-an-own-gogo-shell-command/>

### Этап 5. Создание приложения

Требуется создать приложение, которое получает через API новостных порталов список актуальных новостей и выводит на консоль 10 самых часто встречающихся слов из заголовков новостей. Данные берутся из следующих источников:

1. Lenta.ru API (<https://api.lenta.ru/rss>).
2. RSS АиФ (<http://www.aif.ru/rss/news.php>).
3. Прочие источники на усмотрение студента.

Процесс подсчета инициируется пользователем приложения с помощью консольной команды «news:stats». Пользователь может передать источник в качестве параметра. Если команда вводится без параметров, пользователю предлагается выбрать источник данных (один из доступных в системе, или все сразу). Список источников выводится на консоль.

Если пользователь выбирает незарегистрированный (недоступный) источник данных, в системе не зарегистрировано ни одного источника или сеть недоступна, пользователю выводится соответствующее сообщение.

Реализация получения данных из каждого источника должна быть отдельной имплементацией общего для всех источников сервиса (интерфейса).

По желанию можно написать Unit тесты.

***Рекомендация****: сервисы доступа к данным из различных источников реализуйте в разных бандла0445.*

## Общие требования для всех этапов

1. Следуйте [Java Code Conventions](https://www.oracle.com/java/technologies/cc-java-programming-language.html) (обязательное условие для получения зачета).
2. Система сборки – Maven.
3. IDE – Intellij IDEA или Eclipse.

# Создание консольного приложения на OSGi

### Этап 1. Подготовительный

Входе выполнения подготовительного этапа я освоил основные концепции OSGI: modularity, lifecycle, services и т. п.

Кратко об основных концепциях

* Модульный уровень -связан с упаковкой файлов и общим кодом
* Уровень жизненного цикла – связан с обеспечением управления модулями во время выполнения
* Сервисный уровень – касается взаимодействия и связи между модулями, точнее содержащимися в них компонентами

Подробнее о module layer

* Модульность — это форма разделения ответственности, обеспечивающая как логическую, так и физическую инкапсуляцию классов.
* Желательна модульность, поскольку она позволяет логически разбивать приложения на независимые части, которые могут быть независимо изменены и обоснованы.
* Bundle — это имя модуля в OSGi. Это файл JAR, содержащий код,
* ресурсов и метаданных модульности.
* Метаданные модульности детализируют удобочитаемую информацию, идентификацию пакета и видимость кода.
* Видимость кода пакета состоит из внутреннего пути к классу, экспортированных пакетов, и импортированных пакетов, что значительно отличается от предположения о глобальном типе стандартных файлов JAR.
* Платформа OSGi использует метаданные об импортированных и экспортированных пакетах для автоматического разрешения зависимостей пакетов и обеспечения согласованности типов, прежде чем пакет можно будет использовать.
* Импортированные и экспортированные пакеты фиксируют зависимости пакетов между пакетами, но ограничения использования необходимы для захвата зависимостей пакетов внутри пакета, чтобы обеспечить полную согласованность типов.

Подробнее о lifecycle layer:

* Пакет можно использовать, только установив его в работающий экземпляр OSGi фреймворка.
* API уровня жизненного цикла состоит из трех основных интерфейсов: BundleActivator, BundleContext и Bundle.
* BundleActivator — это то, как пакет подключается к уровню жизненного цикла, чтобы стать с учетом жизненного цикла, что позволяет ему получить доступ ко всем средствам инфраструктуры для проверки и изменения состояния фреймворка во время выполнения.
* Платформа связывает состояние жизненного цикла с каждым установленным пакетом, а Интерфейсы жизненного цикла BundleContext и Bundle позволяют переходить пакетам через эти состояния во время выполнения.
* Мониторинг событий жизненного цикла пакетов — это доступная форма динамического расширения.
* В среде OSGi на основе динамически изменяющегося установленного набора пакетов (также известного как шаблон расширения).
* Уровни жизненного цикла и модуля имеют тесную связь, что подтверждается при обновлении и удалении пакетов. Вы используете службу администрирования пакетов для управления этим взаимодействием.

Подробнее о Services:

* Услуга — это «работа, выполненная для другого».
* Контракты на обслуживание определяют обязанности и сопоставляют потребителей с поставщиками.
* Услуги поощряют непринужденный, подключаемый, основанный на интерфейсе подход к программированию.
* Вам не нужно заботиться о том, откуда поступает услуга, если она соответствует контракту.
* Лучше всего использовать сервисы между заменяемыми компонентами.
* Тщательно подумайте, прежде чем использовать службы в тесно связанном или высокопроизводительном коде.
* Службы могут заменить шаблон слушателя гораздо более простым шаблоном белой доски.
* Службы OSGi используют модель публикации-нахождения-привязки.
* Службы OSGi действительно динамичны и могут появиться или исчезнуть в любое время.
* Самый простой и безопасный подход — использовать служебный класс OSGi ServiceTracker.
* Для абстракций службы более высокого уровня рассмотрите модели компонентов в главах 11 и 12.
* Службы OSGi основаны на предыдущих модулях и уровнях жизненного цикла и взаимодействуют с ними.
* OSGi определяет базовые службы инфраструктуры и дополнительные службы компендиума.

Apache Felix установлен и запущен

Изображение выглядит как текст

Автоматически созданное описание

Web console и другие бандлы установлены

Изображение выглядит как текст

Автоматически созданное описание

### Этап 2. Реализация OSGi-сервиса

Струкра бандла

Изображение выглядит как текст, снимок экрана, монитор, черный

Автоматически созданное описание

Интерфейс содержащий метод печати

Изображение выглядит как текст

Автоматически созданное описание

Реализация интерфейса, содержащего метод печати

Изображение выглядит как текст

Автоматически созданное описание

Активатор бандла печати

Изображение выглядит как текст

Автоматически созданное описание

Активатор бандла клиента

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Использование сервиса печати сервисом клиента

package org.service.client;  
  
import org.service.printer.IPrinter;  
  
public class RunnablePrinter implements Runnable  
{  
 private final int TWO\_SECS = 3000;  
 private static int *counter* = 0;  
 private boolean stop;  
  
 private IPrinter service;  
  
 void setPrinterService(IPrinter service)  
 {  
 this.service = service;  
 }  
  
 void start()  
 {  
 stop = false;  
 new Thread(this).start();  
 }  
 void stop()  
 {  
 stop = true;  
 }  
  
 public void run() {  
 while (!stop) {  
 service.print("hello [" +  
 String.*valueOf*(++*counter*) + "] ...");  
 try {  
 Thread.*sleep*(TWO\_SECS);  
 if (*counter* == 5) {  
 *counter* = 0;  
 stop = true;  
 }  
 } catch (InterruptedException e) {  
 stop = true;  
 }  
 }  
 }  
}

### Этап 3. Apache Felix Service Component Runtime

Структура бандла

Изображение выглядит как текст, снимок экрана, монитор, внешний

Автоматически созданное описание

Интерфейс содержащий метод печати

Изображение выглядит как текст

Автоматически созданное описание

Реализация интерфейса, содержащего метод печати

Изображение выглядит как текст

Автоматически созданное описание

Использование сервиса печати сервисом клиента

Изображение выглядит как текст

Автоматически созданное описание

Бандлы успешно установлены и активированы, что видно через Web Console



### Этап 4. Создание собственной команды для Apache Felix Gogo

Структура бандла

Изображение выглядит как текст, монитор, снимок экрана, экран

Автоматически созданное описание

Реализация команды

Изображение выглядит как текст, снимок экрана, монитор, экран

Автоматически созданное описание

Ссылка на гитхаб репозиторий: <https://github.com/ParzivalP1/Studying-OSGi>

**Выводы:**

Подводя итоги, я выделил для себя несколько важных аспектов:

* Платформа Java отлично подходит для разработки приложений, но ее модульность в значительной степени ограничивается мелкозернистыми объектно-ориентированными механизмами нежели более грубыми функциями модульности, необходимыми для управления проектами.
* Сервисная платформа OSGi через инфраструктуру OSGi устраняет недостатки модульности Java, создавая мощное и гибкое решение.
* Декларативный подход, основанный на метаданных, используемый OSGi, обеспечивает неинвазивный способ использования его сложных модульных возможностей путем изменения способа упаковки проектов с небольшими изменениями в коде.
* Платформа OSGi определяет контролируемый динамический жизненный цикл модуля для упрощения управление.
* Следуя хорошим принципам проектирования, OSGi продвигает подход к программированию на основе интерфейсов, чтобы отделить интерфейсы от реализаций.
* Без нужных зависимостей некоторые модули не будут работать

Работа над заданиями этой практики была довольно интересной, несмотря на возникшие на пути трудности. Пришлось изучить Maven, Apache Felix, OSGi, прочитать много документации на английском. Однако, всё это становится мелочами, когда приходит понимание и осознание всей пользы, что приносят эти технологии для языка программирования Java.