Липецкий государственный технический университет

Кафедра АСУ

Индивидуальное домашнее задание

по организации графических систем и систем мультимедиа

Студент Бровкин А.Н.

Группа М-АС-19

Руководитель Кургасов В.В.

Липецк

2020

**Задание кафедры**

Все проекты одного студента должны быть упакованы в один контейнер Docker, после запуска которого станет возможным запуск всех выполненных им проектов.

# Теоретические сведения

## Spring Boot

Приложение было разработано с использованием Фреймворка Spring Boot. Среда Spring Boot представляет способ создания готовых к использованию Spring-приложений с минимальным временем настройки. Основные цели создания проекта Spring Boot были сконцентрированы на идее, согласно которой пользователи должны иметь возможность быстро освоиться в среде Spring.

В проекте Spring Boot также прослеживается особый взгляд на платформу Spring и библиотеки сторонних разработчиков. Особый взгляд означает, что в Spring Boot излагается набор знакомых абстракций, общий для всех Spring проектов. Этот особый взгляд предоставляет подводки к тому, что необходимо всем проектам, притос ничем не мешая создателю. Таким образом, Spring Boot упрощает замену компонентов при изменении проектных требований. Он основывается на экосистеме Spring и библиотеках сторонних разработчиков[1].

В общем используя Spring Boot разработчику необходимо описывать только бизнес логику приложения, не вдаваясь в подробности как это реализовано внутри.

## Kotlin

В качестве языка разработки бизнес логики приложения был выбран язык Kotlin.

Язык Kotlin предлагает выразительный синтаксис, мощную и понятную систему типов, великолепную поддержку и бесшовную совместимость с существующим кодом на Java, богатый выбор библиотек и фреймворков. Kotlin может компилироваться в байт-код Java, поэтому его можно использовать везде, где используется Java, включая Android. А благодаря эффективному компилятору и маленькой стандартной библиотеке Kotlin практически не привносит накладных расходов.

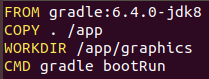
Основная цель языка Kotlin - предложить более компактную, производительную и безопасную альтернативу языку Java, пригодную для использования везде, где сегодня применяется Java. Java - чрезвычайно популярный язык, который используется в самых разных окружениях, начиная от смарт-карт (технология Java Card) до крупнейших вычислительных центров таких компаний, как Google, Twitter и Linkedln. В большинстве таких окружений применение Kotlin способно помочь разработчикам достигать своих целей меньшим объемом кода и избегая многих неприятностей.

Наиболее типичные области применения Kotlin [2]:

* разработка кода, работающего на стороне сервера (как правило, серверной части веб-приложений);
* создание приложений, работающих на устройствах Android.

# Ход выполнения работы

После написания приложения, которое будет упаковываться в Docker образ, создадим Dockerfile



1. Содержимое Dockerfile-а

Опишем каждую строчку данного документа.

FROM gradle:6.4.0-jdk8 – указываем на основе какого образа будет строится образ приложения

COPY . /app – копируем все файлы из текущего местоположения в папку app контейнера (заведомо положим туда папку с проектом Spring Boot)

WORKDIR /app/graphics – устанавливаем рабочую директорию папку проекта

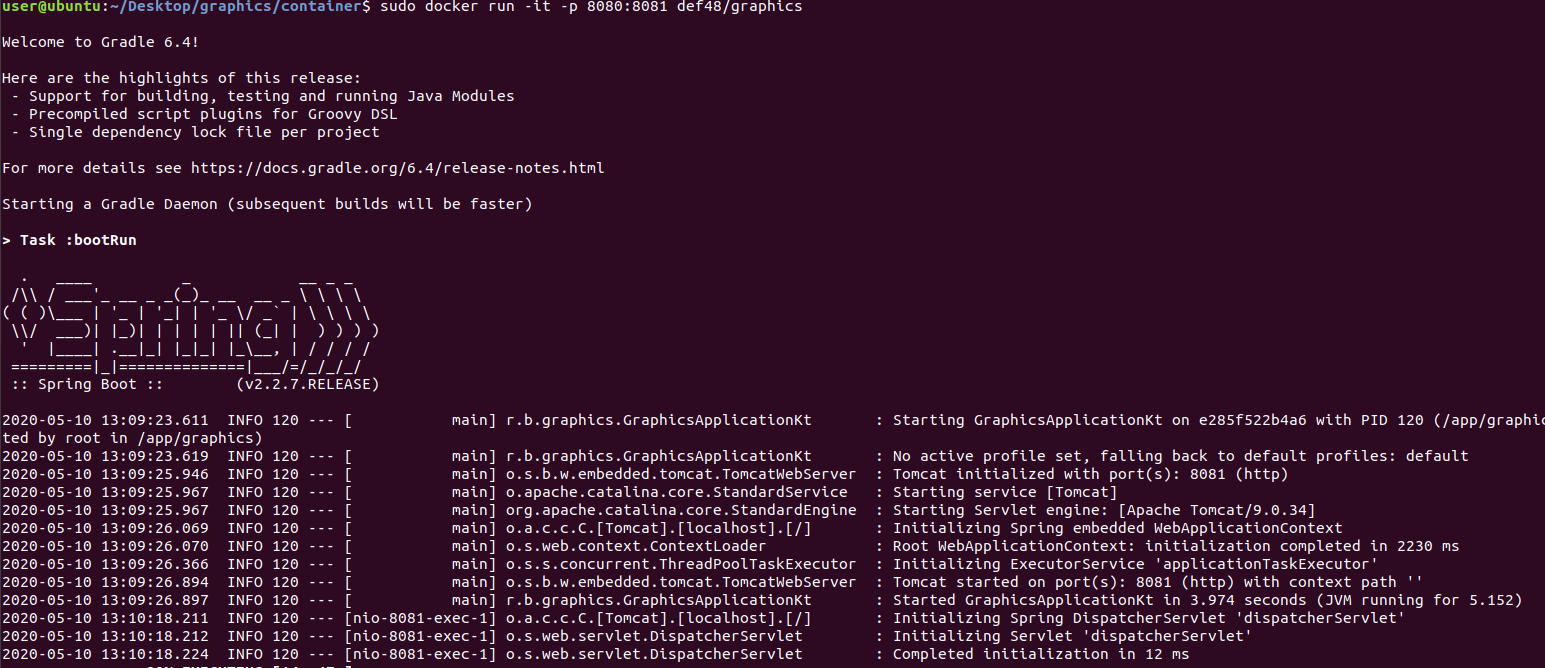
CMD gradle bootRun – показываем какая команда должна вызываться при запуске образа (данная команда соответствует запуску Spring boot приложения).

Далее собираем наш образ (рисунок 2).



1. Сборка образа

Производим запуск контейнера и пробрасываем порт 8081 на порт 8080 хоста.



1. Запуск контейнера.

# Листинг кода

## GraphicApplication.kt

package ru.brovkin.graphics  
  
import org.springframework.boot.autoconfigure.EnableAutoConfiguration  
import org.springframework.boot.autoconfigure.SpringBootApplication  
import org.springframework.boot.runApplication  
import org.springframework.context.annotation.ComponentScan  
import org.springframework.context.annotation.Import  
  
@SpringBootApplication //указывем что этот класса является классом приложения Spring Boot  
@EnableAutoConfiguration //разрешаем автоконфигурацию приложения  
@ComponentScan(basePackages = ["ru.brovkin.graphics"]) //указываем базовы пакет прилоэения для обнаружения вспомоагтельных элементов  
class GraphicsApplication  
  
//точка входа в приложение  
fun main(args: Array<String>) {  
 *runApplication*<GraphicsApplication>(\*args)  
}

## LabController.kt

package ru.brovkin.graphics.controllers  
  
import org.springframework.beans.factory.annotation.Autowired  
import org.springframework.stereotype.Controller  
import org.springframework.web.bind.annotation.GetMapping  
import org.springframework.web.bind.annotation.RequestMapping  
import org.springframework.web.bind.annotation.RequestMethod  
import org.springframework.web.bind.annotation.ResponseBody  
import org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer  
  
//контроллер для web запросов  
@Controller  
class LabController {  
 //запрос для лабораторной работы №1  
 @RequestMapping(value = ["/lab1"])  
 fun getLabFirstPage1(): String {  
 return "page1"  
 }  
 //запрос для лабораторной работы №2  
 @RequestMapping(value = ["/lab2"])  
 fun getLabFirstPage2(): String {  
 return "page2"  
 }  
 //запрос для лабораторной работы №3  
 @RequestMapping(value = ["/lab3"])  
 fun getLabFirstPage3(): String {  
 return "page3"  
 }  
 //запрос для лабораторной работы №4  
 @RequestMapping(value = ["/lab4"])  
 fun getLabFirstPage4(): String {  
 return "page4"  
 }  
}

## application.properties

Для работы Spring Boot требуется специальный файл настроек. В данном случае были использованы следующие настройки:

server.port – указываем порт на котором будет работать приложение

spring.freemarker.template-loader-path – указываем путь до web страниц приложения

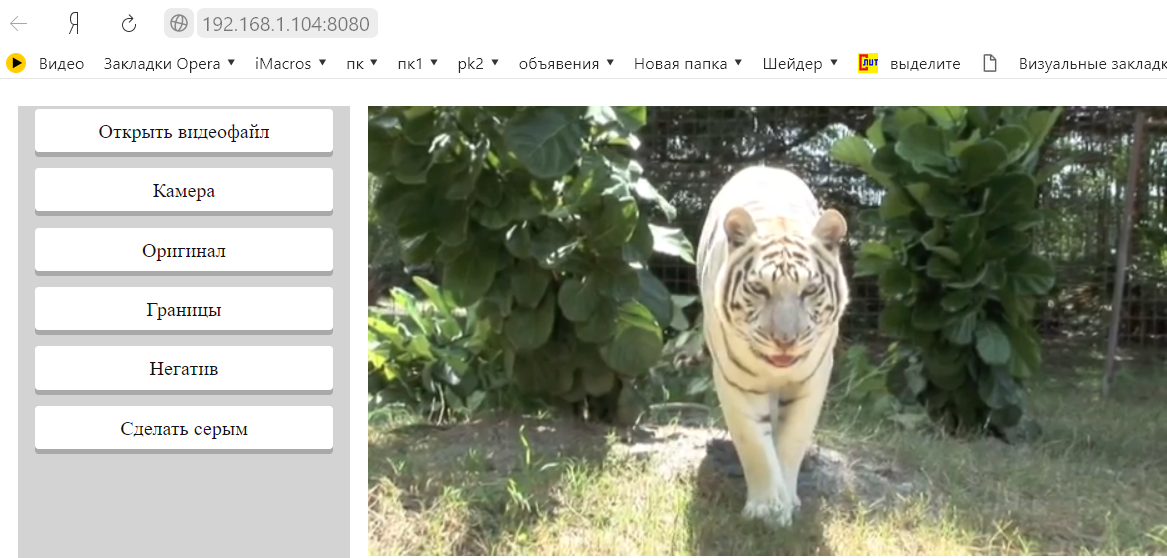
spring.freemarker.suffix – указываем расширения для web страниц

В итоге получился следующий файл

server.port = 8081  
  
spring.freemarker.template-loader-path: classpath:/templates  
spring.freemarker.suffix: .ftl

# Результаты выполнения работы

После запуска контейнера, приложение будет работать на порту 8080 машины хоста, в данном случае для запуска 4-ой лабораторной работы используется ссылка <http://192.168.1.104:8080/lab4>.



1. Ответ на запрос к 4-ой лабораторной работы

**Вывод**

В ходе выполнения индивидуального домашнего задания были изучены основы работы с Docker. Сами по себе контейнеры напоминают маленькие виртуальные машины, так как они тоже используют технологии виртуализации, но запускаются гораздо быстрее. В контейнер можно упаковать все необходимое и тем самым запустить его независимо от платформы, достаточно иметь установленный Docker.

**Список используемой литературы**

1. Лонг, Д. Java в облаке. Spring Boot, Spring Cloud, Cloud Foundry [Текст] / Д. Лонг, К. Бастани. – Санкт-Петербург: Питер, 2019. – 624 c.
2. Жемеров, Д Kotlin в действии [Текст] / Жемеров Д., Исакова С.; пер. с англ. Киселевым А. Н. - Москва: ДМК Пресс, 2018. - 402 с.