AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
Wydział Fizyki i Informatyki Stosowanej

Praca Magisterska

Sylwester Macura

kierunek studiów: **Informatyka Stosowana**

Wykorzystanie komponentów projektu Spring w Systemie Zarządzania Treścią

Opiekun: **dr inż. Barbara Kawecka-Magiera**

Kraków, Lipiec 2016

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomowa wykonałem osobiście i samodzielnie i nie korzystałem ze źródeł innych niż wymienione w pracy.

.....

(czytelny podpis)

Merytoryczna ocena pracy przez opiekuna

Końcowa ocena pracy przez opiekuna: . . . . . .

Data: . . . . . Podpis: . . . . . .

Skala ocen: (6.0 – celująca), 5.0 – bardzo dobra, 4.5 – plus dobra, 4.0 – dobra, 3.5 – plus dostateczna, 3.0 – dostateczna, 2.0 - niedostateczna

Merytoryczna ocena pracy przez recenzenta

Końcowa ocena pracy przez recenzenta: . . . . . .

Data: . . . . . Podpis: . . . . . .

Skala ocen: (6.0 – celująca), 5.0 – bardzo dobra, 4.5 – plus dobra, 4.0 – dobra, 3.5 – plus dostateczna, 3.0 – dostateczna, 2.0 – niedostateczna

Spis treści

1 Wstęp

2 Cel pracy

3 Narzędzia użyte w pracy

3.1 Java

Java jest obiektowym językiem programowania stworzonym w Sun Microsystems [1] (obecnie część Oracle). Programy napisane w Javie są kompilowane do kodu pośredniego (bytecode), a następnie uruchamiane na wirtualnej maszynie (ang. Java Virtual Machine, JVM) . Dzięki takiemu rozwiązaniu skompilowany program jest niezależny od platformy, wystarczy tylko że będziemy mogli zainstalować JVM. W JVM jest dodatkowo wbudowany mechanizm automatycznie zwalniający pamięć (ang. Garbage Collector) przez co nie musimy zajmować się zarządzaniem pamięcią. Składnia języka jest silnie wzorowana na C++. Oprócz tego Java posiada aktywną i rozbudowaną społeczność oraz wiele dostępnych narzędzi pomocniczych. Wszystko to sprawia że jest jednym z najpopularniejszych języków programowania.

3.2 Gradle

Gradle jest otwarto źródłowym narzędziem budującym pozwala na definiowanie skryptów budujących w języku Groovy [2]. Dzięki niemu możemy budować aplikacje na różne platformy napisane w rożnych językach. Posiada bogatą kolekcje rozszerzeń która pozwala rozbudowywać oraz upraszcza skrypty budujące.

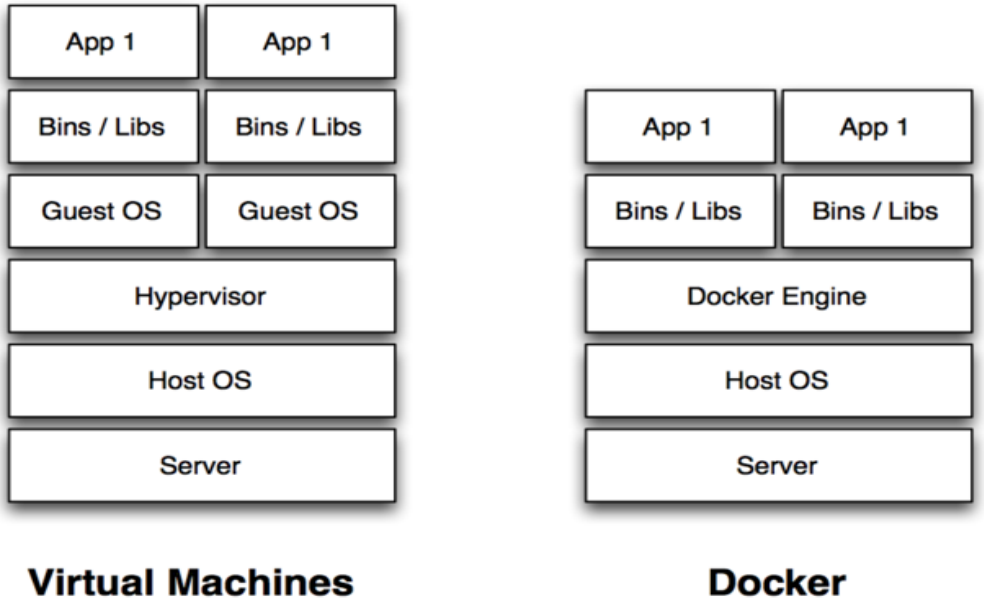
3.2.1 Użycie w pracy

W aplikacji gradle został wykorzystany do następujących rzeczy:

- Budowy części serwerowej
- Budowy aplikacji Android
- Generowanie plików Dockerfile
- Uruchamiania poszczególnych części aplikacji dla celów developerskich

3.3 Docker

Docker jest to otwarto źródłowe narzędzie do uruchamiania aplikacji wewnątrz kontenerów. Kontener jest to lekka i przenośna maszyna wirtualna, która może zostać uruchomiona na dowolnym serwerze z systemem Linux [3].



Ilustracja 1: Docker

http://core0.staticworld.net/images/article/2016/07/dockerswarm-fig01-100671308-large.idge.png

Plik Dockerfile to przepis jak stworzyć obraz Docker, zawiera on wszystkie rzeczy potrzebne do uruchomianie aplikacji oraz zależności. Następnie z tego obrazu możemy stworzyć kontener, czyli działającą maszynę wirtualną z naszą aplikacją. Dzięki wsparciu różnych platform PaaS (ang. Platform as a Service) dla Docker jest on idealnym rozwiązaniem do wgrywania różnych usług chmurowych zachowując przy tym niezależność od platformy na którą jest wgrywany.

```
FROM java:8
MAINTAINER Sylwester Macura <sylwestermacura@gmail.com>
EXPOSE 8080
COPY libs/user-micro-service-0.0.1-SNAPSHOT.jar user-micro-service-0.0.1-
SNAPSHOT.jar
ENTRYPOINT ["java", "-Dspring.profiles.active=docker", "-jar", "user-micro-
service-0.0.1-SNAPSHOT.jar"]
```

Kod 1: Przykładowy Dockerfile

Każdy obraz Docker dziedziczy po innym, w tym przypadku dziedziczymy po obrazie java:8 zawiera on już zainstalowaną Jave w wersji 8. Kolejna linijka wskazuje na autora obrazu. Polecenie expose udostępnia porty kontenera, które będą widoczne z zewnątrz. Kolejna instrukcja wskazuje

jak zbudować obraz, w przykładzie kopiujemy skompilowaną aplikację do obrazu. Entrypoint definiuje polecenie jakie ma się wykonać przy starcie kontenera.

3.3.1 Docker-compose

Docker-compose jest narzędziem pomocniczym dla Docker, które pozwala na uruchomienie oraz połączenie wielu kontenerów. Narzędzie opiera się o plik docker-compose.yml, w którym konfigurujemy jakie obrazy Dockera mają być ściągnięte oraz jak mają być połączone, dodatkowo tworzy wewnętrzną sieć dla naszych kontenerów. Możemy upublicznić niektóre porty z konkretnych kontenerów aby uzyskać dostęp do aplikacji.

```
discovery-server:
  image: magisterka-cms/image-discovery-server
  ports:
    - "8770:8080"
  external_links:
    - image-config-server:config-server
edge-server:
  image: magisterka-cms/image-edge-server
  ports:
    - "8769:8080"
  links:
    - discovery-server
  external_links:
    - image-config-server:config-server
```

Kod 2: docker-compose.yml Przykład konfiguracji

Kod 1 zawiera przykład pliku konfiguracyjnego docker-compose. W przypadku wykonania polecenia:

docker-compose up

Zostaną stworzone i uruchomione dwa kontenery. Pierwszy zostanie stworzony z obrazu o nazwie magisterka-cms/image-discovery-server, port kontenera 8080 zostanie zmapowany na port 8770 hosta. Dodatkowo kontener będzie posiadał wpis DNS (ang. Domain Name System) „config-server” który będzie prowadził do zewnętrznego kontenera o nazwie image-config-server. Następnie zostanie stworzony kolejny kontener z obrazu magisterka-cms/image-edge-server którego port 8080 zostanie zmapowany na port 8769 hosta. Ten kontener będzie posiadał dwa wpisy DSN jeden prowadzący do kontenera discovery-server i drugi prowadzący do zewnętrznego kontenera image-config-server.

Teraz możemy bardzo łatwo skalować w szerz nasze kontenery, wywołując polecenie:

docker-compose scale discovery-server=3

W efekcie zostaną stworzone dwie dodatkowe instancje obrazu magisterka-cms/image-discovery-

server.

3.3.2 Docker Hub

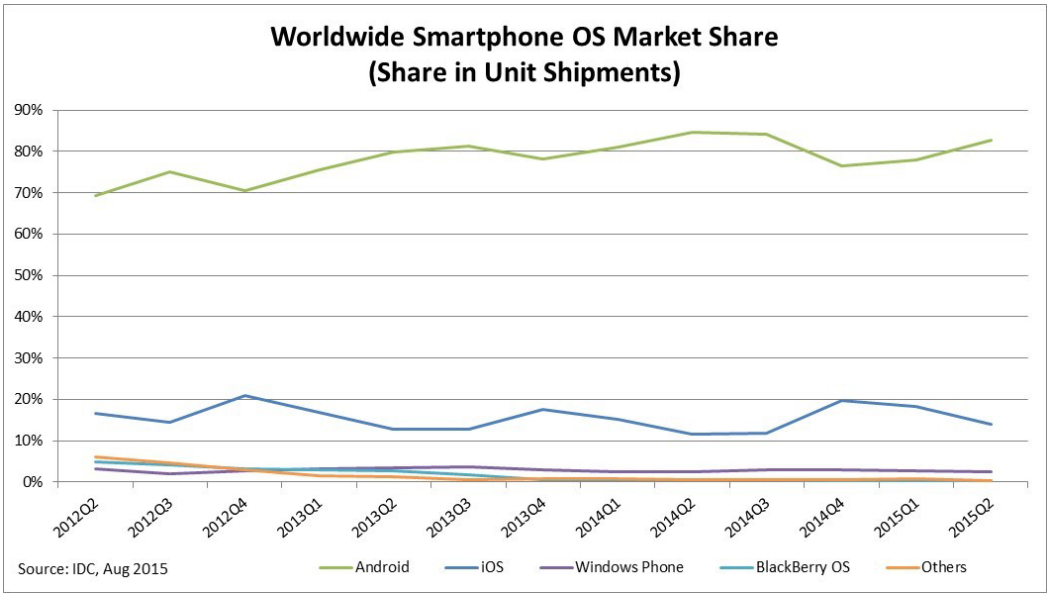
Docker Hub jest centralnym repozytorium dla obrazów Docker. Z tego miejsca są ściągane obrazy gdy nie można ich znaleźć na lokalnej maszynie. Do zalet należą darmowa rejestracja oraz bogata kolekcja już gotowych obrazów.

3.3.3 Użycie w pracy

Docker został użyty aby wgrywać aplikację na serwer zewnętrzny Google Compute Engine. Dzięki użyciu narzędzia pomocniczego docker-compose wgranie całej aplikacji sprowadza się do wykonania jednego polecenia.

3.4 Android

Android jest to system operacyjny z jądrem Linux dla urządzeń mobilnych takich jak telefony komórkowe, smartfony, tablety i netbooki [4]. Obecnie jest najpopularniejszym systemem operacyjny na urządzenia mobilne.



Ilustracja 2: Popularność systemów mobilnych

http://www.idc.com/prodserv/smartphone-ms-img/chart-ww-smartphone-os-market-share.png

Oprócz dużej społeczności posiada bardzo rozwinięte narzędzia developerskie. Aplikacje natywne są tworzone w języku Java, a warstwa widokowa za pomocą plików XML, oprócz tego posiada zaawansowane narzędzia do dostosowywania widoków, kolorów w zależności od urządzenia.

3.4.1 Android Annotations

Android annotations jest otwartoźródłowym narzędziem wspomagającym tworzenie aplikacji na platformę Android [5]. Dzięki użyciu adnotacji możemy bardzo uprościć nasz kod oraz zwiększyć jego przejrzystość.

```
@EActivity(R.layout.activity_create_template)
@OptionsMenu(R.menu.create_template_menu)
public class ApplicationTemplateDetailsActivity extends AppCompatActivity {
    public static final String TEMPLATE_INTENT = "DOCUMENT_INTENT";
    ApplicationTemplateDTO template;
    @RestService
    ApplicationTemplateClient templateClient;
    @Bean
    ApplicationTemplateDetailsAdapter adapter;
    @Bean
    ActionsAdapter actionsAdapter;

}
```

Kod 3: Przykład użycia android Annotations

Na zaprezentowanym przykładzie widzimy activity androidowe do którego zostaje przypisany plik widoku activity_create_template.xml (annotacja @Eactivity), oraz menu z pliku create_template_menu.xml (adnotacja @OptionsMenu). Po stworzeniu activity zostają wstrzyknięte instancje klas ApplicationTemplateDetailsAsapter oraz ActionAdapter, oraz na podstawie interfejsu ApplicationTemplateClient zostaje wygenerowany klient do usługi REST (ang. Representational State Transfer).

3.5 Git

Git jest rozproszonym systemem kontroli wersji pierwotnie stworzonym na potrzeby rozwoju jądra Linuxa [6]. Obecnie jest jednym z najpopularniejszych systemów kontroli wersji (ang. Version Control System z skrócie VCS) używanych w tworzeniu oprogramowania. Jego rozproszona natura umożliwia łatwą współpracę w czasie tworzenia aplikacji. Dodatkowo istnieje wiele usług hostujących repozytoria Git np. Bitbucket czy GitHub.

3.5.1 GitHub

GitHub jest platformą hostującą repozytoria Git. Mamy do wyboru bezpłatne repozytoria publiczne oraz płatne repozytoria prywatne. Oprócz tego mamy do dyspozycji issue tracker który pozwala na monitorowanie naszych postępów w pisaniu kodu, zgłaszanie błędów, podpinanie commitów pod poszczególne zgłoszenia oraz wyznaczanie kamieni milowych naszej aplikacji. Dodatkowo mamy do dyspozycji bogatą kolekcję webhooks, która pozwala na integrację z zewnętrznymi narzędziami np. TravisCI.

3.6 TravisCI

TravisCI jest narzędziem do Ciągłej Integracji (ang. Continuous Integration, CI). Ciągła integracja polega na ciągłym budowaniu i testowaniu aplikacji aby jak najszybciej wyłapywać błędy powstałe podczas tworzenia systemu. Travis posiada pakiet darmowy, (który choć z ograniczonym funkcjonalnościami) stanowi i tak bardzo przydatne narzędzie do CI. Nasze środowisko konfigurujemy za pomocą pliku .travis.yml, pozwala to na łatwe dodawanie usług do naszego środowiska testowego np. Javy, Dockera.

```
sudo: required
language: java
services:
  - docker
script:
  - ./gradlew build -Pbuild-travis="" -x test
after_success:
  - docker login -e="$DOCKER_EMAIL" -u="$DOCKER_USERNAME" -p="$DOCKER_PASSWORD";
    docker push wemstar/magisterka-cms-image-edge-server;
```

Kod 4: .travis.yml Konfiguracja TravisCI

Plik .travis.yml dzieli się na następujące sekcje:

- sudo – określa czy potrzebujemy uprawnień administratora aby zbudować aplikację, w tym przypadku potrzebujemy dlatego umieszczamy wartość requires
- language – określa język naszego projektu dla nas jest to java, dołącza JDK (ang. Java Development Kit) do środowiska oraz mavena i gradle
- script – jest to polecenie którym zbudujemy projekt, domyślnie Travis po wykryciu build.gradle zbuduje naszą aplikację narzędziem gradle, ale potrzebujemy dodatkowe zmienne aby zbudować obrazy dockera.
- after_sucess – co mamy zrobić po pomyślnym zbudowaniu aplikacji, tutaj zaloguje się do Docker Hub oraz przeniesie obraz wemstar/magisterka-cms-image-edge-server

3.7 Google Compute Engine

Google Compute Engine jest narzędziem IAAS i jest częścią Google Cloud Platform. Infrastructure as a Service (IAAS, z ang. „infrastruktura jako usługa”) to jeden z modeli chmury obliczeniowej. Jest to usługa polegająca na dostarczeniu całej infrastruktury informatyczne, takie jak wirtualizowany sprzęt, skalowany w zależności od potrzeb użytkownika[7]. W aplikacji został wykorzystany do hostowania Dockera na którym jest uruchomiona aplikacja.

3.8 IntelliJ IDEA

IntalliJ IDEA jest zintegrowanym środowiskiem programistycznym (ang. Integrated Development Environment, IDE) stworzonym przez firmę JetBrains. Jest to jedno z najpopularniejszych narzędzi tego typu. Zapewnia świetna integracją z wieloma frameworkami oraz narzędziami wspomagającymi developerów np. Spring, SpringBoot, Docker, Git i wiele innych. Dodatkowo możemy rozszerzyć jego funkcjonalność dzięki bogatej bazie pluginów. Oprócz tego jest to wydajne środowisko oraz posiada intuicyjny interfejs.

3.8.1 Android Studio

Android Studio jest to wersja IntelliJ IDEA przeznaczona specjalnie do tworzenia aplikacji androidowych. Jest to oficjalne środowisko zalecane przez Google, posiada dodatkową integrację z usługami Google.

4 Projekt Spring

Projekt z Spring powstał jako narzędzie wspomagające tworzenie projektów J2EE (ang. Java Platform, Enterprise Edition). Jego pierwsza wersja została stworzona przez Roda Johnsona jako część książki po tytułem „Expert One-on-One J2EE Design and Development” i została wydana w 2002 [8]. Projekt wprowadzał wiele ułatwień i nowych rozwiązań które stawały się później standardami. Z czasem projekt się rozrósł, zarówno na inne języki programowania jak i na inne zagadnienia związane z tworzeniem aplikacji. Sam projekt jak i jego składowe są udostępnione na licencji Apache License 2.0. Aktualnie stabilną dostępną wersją jest 4.3.0. Ogromną zaletą Spring jest to że kod nie jest stale związany z frameworkiem. Pozwala to zmienić narzędzie na inne bez większych trudności.

4.1 Spring Core

Spring Core jest najważniejsza częścią frameworku Spring to na nim opierają się pozostałe składowe, najczęściej żeby użyć jednej z części projektu Spring musimy dołączyć.

4.1.1 Contener Benów

Ważną częścią Spring jest jego kontener Beanów. To tu są tworzone klasy które zdefiniujemy w kodzie oraz te dostarczane przez narzędzie Spring. Oprócz samego tworzenie klas to tutaj są wsztrzykiwane zależność (ang. Dependency Injection). To również w tym miejscu nasza klasa jest opakowywana np. w aspekty czy klasy tworzące logi. Beany możymy stworzyć na dwa sposoby, pierwszy przez konfigurację w pliku XML, drugi przy użyciu adnotacji na naszej klasie, trzeci to stworzenie metody w Javie która wyprodukuje instację naszej klasy.

/// Przykład z Adnotacjami

///Przykład z XML

/// Przykład z konfiguracja w javie

4.1.2 Dependency Injection

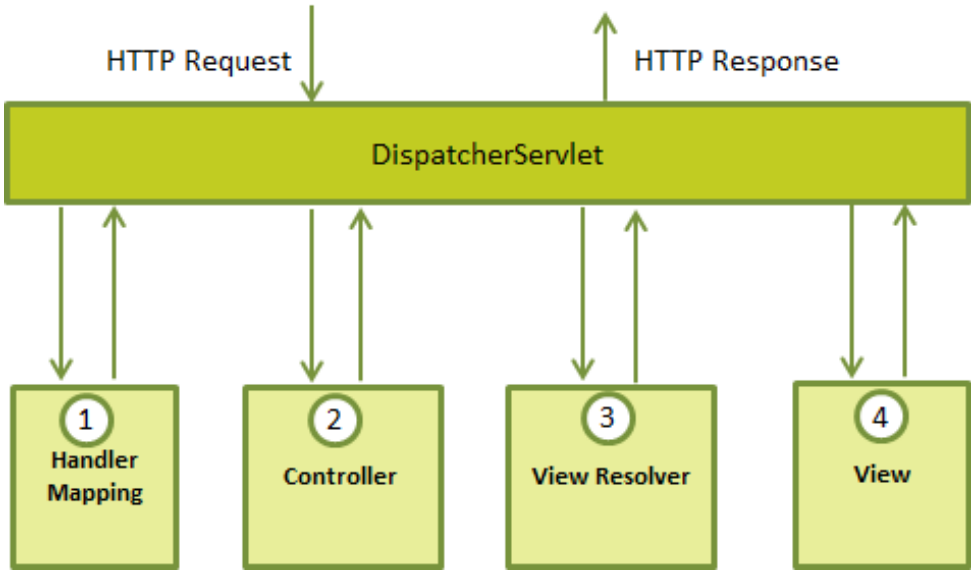
Wstrzykiwanie zależności (ang. Dependency Injection) jest jednym z najczęściej stosowanych technik zarówno w samym projekcie Spring jak i w aplikacja stworzonych przy jego pomocy. Obiekty możemy wstrzykiwać za pomocą konstruktorów lub seterów lub konstruktorów. Od strony użytkownika możemy to osiągnąć odpowiednimi adnotacjami umieszczonymi na konstruktorze lub seterze. Jest też możliwość umieszczenie konfiguracji w xml. Wstrzykiwanie zależności umożliwia pisanie luźno powiązanych ze sobą obiektów (ang. Loose coupling), pozwala to na łatwa wymianę poszczególnych komponentów aplikacji. Dzięki temu nasza plikacja staje się prostsza i łatwiej ją utrzymywać

4.1.3 Spring MVC

Spring MVC jest jednym z komponentów Spring Core. Pozwala on na pisanie aplikacji webowych za pomocą wzorca MVC (ang. Model View Controller). MVC jest jednym z najpopularniejszych wzorców projektowych wykorzystywanych przy tworzeniu aplikacji webowych. Aplikację składają się z 3 części:

- Model – odpowiada za pobieranie oraz przechowywanie danych.
- View – widok odpowiada za wyświetlanie danych z modelu oraz przekazywanie komunikatów do kontrolera.
- Controller – odpowiada za przetwarzanie danych w odpowiedzi na komunikaty przesłane z widoku.

Dzięki takiemu podejściu do pisania aplikacji, możemy łatwo wymieniać poszczególne elementy (modele, widoki i kontrollery). Daje nam to aplikacje którą łatwo się rozwija.



Ilustracja 3: Architektura spring MVC

http://www.tutorialspoint.com/spring/images/spring_dispatcherservlet.png

Główną częścią Spring MVC jest DispatcherServlet, jest to servlet których rejestrujemy w pliku web.xml. W wywołanie metody HTTP przebiega następująco:

1. HandlerMapping mapuje zapytanie i wyszukuje odpowiedni kontroler, jeśli go znajdzie zapytanie jest tam przykazywane.
2. Controller jest miejscem gdzie wykonywana jest logika naszej aplikacji
3. ViewResolver wyszukuje widok w którym zostanie umieszczony dane z modelu.
4. View odpowiada za wyświetlenie wyniku zapytania.

```
<web-app id="WebApp_ID" version="2.4"
            xmlns="http://java.sun.com/xml/ns/j2ee"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
            http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
<servlet>
            <servlet-name>mvc-dispatcher</servlet-name>
            <servlet-class>org.springframework.web.servlet.DispatcherServlet
</servlet-class>
            <load-on-startup>1</load-on-startup>
   </servlet>
   <servlet-mapping>
            <servlet-name>mvc-dispatcher</servlet-name>
            <url-pattern>*.htm</url-pattern>
   </servlet-mapping>
</web-app>
```

Kod 5: Przykład pliku web.xml

W tym przykładzie tworzymy servlet DispatcherServlet o nazwie mvc-dispatcher. Kazdy adres URL kończący się na .htm zostanie do niego przekazany i jeśli znajdzie odpowiedni kontroler przekaże go do wywołanie do niego.

Kontrolery możemy tworzyć poprzez rzszerzenie klasy AbstractController i nadpisanie odpowiedniej metody, jest to stary spobób i już nie zalecany. Polecanym sposobem jest użycie adnotacji.

```
@Controller
public class HelloWorldController {


  @RequestMapping("/helloWorld")
  public String helloWorld(Model model) {
    model.addAttribute("message", "Hello World!");
    return "helloWorld.jsp";
  }
}
```

Kod 6: Przykład Kontrolera

W Kod 6 mamy przykład kontrolera, adnotacja @Controller wskazuje na to. Adnotacja @RequestMapping informuje dla jakiej sciezki ma być wywołana metoda. Oprócz samej ścieżki możemy też ustawić dla jakiej metody HTTP będzie wykonywana funkcja. Sama metoda umieszcza w modelu parametr message o wartości „Hello World!” następnie wywołanie jest przykazywane do widoku helloWorld.jsp.

Adnotacja @RestController jest jedną z adnotaci wywodzących się z @Controller, ułatwia pisanie usług REST (ang. Representational State Transfer).

4.1.4 Spring AOP

Programowanie aspektowe (ang. Aspect-Oriented Programming, AOP) to sposób programowania wspomagający jak największa separację części programów nie związanych funkcjonalnie [9]. Gówną przyczyną powstania tego paradygamtu były problemy z wykonywaniem zadań pobocznych (autoryzacji, monitoringu aplikacji) w ramach funkcjonalności. Powodowało to nie tylko zaciemnienie oryginalnego kodu ale również mutltiplikowanie tego samego kodu w różnych miejscach. Programowanie aspektowe stara się rozwiązać te problem poprzez przekierowanie zadań pobocznych do aspektów które opakowują naszą funkcjonalność. Z programowanie aspektowym wiążą się trzy ważne pojęcia:

- Aspekt – zbiór zadań w ramach jednej funkcjonalności
- Joint Point – miejsce w którym zostanie nałożony aspekt
- Advice – funkcjonalność która ma zostać wykonana w ramach aspektu.

Jedną z implementacji dostępnych na platformę Java jest Spring AOP. Od strony implementacyjnej Spring AOP implementuje wzorzec Proxy na klasach wskazanych w Joint Point.

// Przykład Aspektów

4.2 Spring Security

Spring Security jest frameworkiem który pozwala na autoryzacje i uwierzytelnianie programów napisanych w Javie. Jego największa zaletą jest łatwość w rozszerzaniu tak aby mógł sprostać wyzwaniom klienta [10]. Do jego możliwości należą:

- wspomaganie autoryzacji i autentykacji
- zabezpieczenie przed atakami typu: session fixation, clickjacking, cross site request forgery i wiele innych
- integracja ze Spring MVC

W Spring Security użytkownik po poprawnym uwierzytelnianiu użytkownik otrzymuje jedną lub więcej ról. Każda rola składa się z pozwoleń. Na podstawie ról oraz pozwoleń sprawdzane jest czy użytkownik ma dostęp do zasobu. Role reprezentują wysokopoziomowe role w systemie natomiast pozwolenia reprezentują niskopoziomowe role, pozwolenia w systemie. Proces autoryzacji może dotyczyć całych klas lub poszczególnych metod. Preferowanym sposobem konfiguracji jest użycie odpowiednich adnotacji.

```
@PreAuthorize(„hasRole(‘ROLE_USER’)”)
public void create(Contact contact);
```

Kod 7: Przykład autoryzacji przy pomocy roli

Adnotacja @PreAuthorize uruchamia autoryzację przed wywołaniem metody, sprawdzane jest czy użytkownik ma role ‘ROLE_USER’. W przypadku braku tej roli zostanie rzucony wyjątek.

Oprócz konfiguracji za pomocą adnotacji możemy użyć kodu Javy aby ustalić dostęp do zasobów. Taka konfiguracja daje więcej możliwości bo poza samymi dostępami możemy także wyłączyć framework dla niektórych ścieżek, skonfigurować użytkowników, dodać własny system uwierzytelniania i wiele innych.

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
	@Autowired
	private CustomAuthenticationProvider customAuthenticationProvider;
	@Override
	protected void configure(AuthenticationManagerBuilder builder) throws Exception {
	builder.authenticationProvider(customAuthenticationProvider);
	}
	@Override
	protected void configure(HttpSecurity http) throws Exception {
		http.authorizeRequests().anyRequest().authenticated()
		.and().requestCache().requestCache(new NullRequestCache())
		.and().httpBasic();
	}
	@Override
	public void configure(WebSecurity web) throws Exception {
		web.ignoring().antMatchers("/hystrix.stream")
		.and().ignoring().antMatchers("/login");
	}
}
```

Kod 8: Przykład konfiguracji Spring Security

W tym przykładzie mamy do czynienia z konfiguracją (o czym informuje nas adnotacja @Configuration), adnotacja @EnableWebSecurity uruchamia uwierzytelnianie dla całej aplikacji. Klasa rozszerza WebSecurityConfigurerAdapter aby możliwa była konfiguracja Spring Security za pomocą rozszerzenia odpowiednich metod. W metodzie configure(AuthenticationManagerBuilder builder) ustawiamy własny sposób uwierzytelnienia który zaimplementowaliśmy w CustomAuthenticationProvider.

W metodzie configure(HttpSecurity http) ustawiamy autoryzację dla wszystkich zapytań, sposobem uwierzytelnienia będzie Basic Http. Metoda configure(WebSecurity web) wyłącza Spring Security dla dwóch adresów ‘’/login’’ oraz "/hystrix.stream"

4.3 Spring Data

4.3.1 Spring Data Rest

4.4 Spring Boot

4.5 Spring Cloud

4.5.1 Eureka

4.5.2 Zuul

4.5.3 Hystrix

4.5.4 Ribbon

4.5.5 Config Server

4.5.6 Feign

5 Architektura aplikacji

5.1 Mikroserwisy

6 Funkcjonalności systemu

7 Opis aplikacji Android

8 Zawartość dołączonej płyty CD

9 Bibliografia

1: https://pl.wikipedia.org/wiki/Java
2: https://en.wikipedia.org/wiki/Gradle
3: https://en.wikipedia.org/wiki/Docker_(software)
4: https://pl.wikipedia.org/wiki/Android_(system_operacyjny)
5: http://androidannotations.org/
6: https://pl.wikipedia.org/wiki/Git_(oprogramowanie)
7: https://pl.wikipedia.org/wiki/Infrastructure_as_a_Service
8: https://en.wikipedia.org/wiki/Spring_Framework
9: https://pl.wikipedia.org/wiki/Programowanie_aspektowe

10 Indeks ilustracji