**Microservices with Spring Cloud**

(na podstawie tutka z udemy -> <https://www.udemy.com/microservices-with-spring-boot-and-spring-cloud/>)

Ogólnie tutek spoko i w sumie tyle. Aaa no warto wspomnieć o tym, że jak coś było o niczym to zwyczajnie pomijałem w opracowaniu. Warto również wspomnieć, że każdy obrazek, który się pojawi w opracowaniu jest oczywiście nie mój 😊 Wszelkie uwagi po przeczytaniu wskazane ++

To jak było ogólnie to teraz skrobnę coś w miarę szczegółowo. Znaczy nie będzie to turbo szczegółowe bo wiem jak (nie)dobrze piszę, ale postaram się przybliżyć wam co fajnego oferuje nam Spring Cloud. No to zaczynając, jest nam to po to aby ułatwić pracę w architekturze mikroserwisów.

# Spring Cloud Config Server

Port Domyślny: 8888

Jest to serwer, który ma za zadanie wspierać zarządzanie konfiguracją mikroserwisów. Bez znajomości Spring Cloud Config zazwyczaj trzyma się nasze ustawienia razem z kodem w plikach typu properties. No i tu jest problem bo może zdarzyć się tak, że ktoś przez nieuwagę np. ja 😊 zacommituje do publicznego użytku coś co nie powinno być współdzielone np. hasła. Ponadto gdy nie wersjonujemy naszych properties to tracimy możliwość przeglądania historii zmian w tych plikach i może dojść do sytuacji gdzie załadujemy do naszej apki inne ustawienia (apka działa, po prostu ustawienia są inne niż ostatnio) i będziemy zachodzić w głowę czy i dlaczego, ktoś je zmienił, może faktycznie była taka potrzeba, ale po co itp. To może zająć sporo czasu. Kolejny minus properties per mikroserwis to brak możliwości ich współdzielenia. Bez serwera z konfiguracjami musimy kopiować zawartość plików, które można uwspólniać. Do tego dochodzi niebezpieczeństwo przechowywania haseł plain textem. Trochę to minusów ma ale z pomocą przychodzi nasz Config Serwer.

Obraz zawierający tekst

Opis wygenerowany przy bardzo wysokim poziomie pewności

No to tutaj wyżej widzimy jak to pięknie wszystko działa jeśli korzystamy z Config Serwera. To teraz opisze dlaczego to tak ładnie działa, przy okazji jak to zaimplementować i jeszcze pewnie kilka innych rzeczy na raz, żeby za prosto nie dało się tego wszystkiego skumać.

W dużych apkach spotkamy się bowiem z setkami mikroserwisów z których każdy z nich posiada wiele środowisk. Co więcej część z tych środowisk może mieć po kilka instancji tego samego mikroserwisu. Ręczna konfiguracja indywidualnych przypadków jest wtedy bardzo bardzo trudna. Noi tu nasz serwerek mówi – OK, ty mi włóż gdzieś (np. do gita) całą konfigurację twoich mikroserwisów a ja już zadbam o zarządzanie nimi. I tak np. jeden mikroserwis powie: „Hej, chcę być konfigurowany za pomocą ustawień produkcyjnych”, a inny „Hola, hola! Jeszcze coś zepsuję, yyy… mi wystarczy konfiguracja testowa”. W każdym z tych przypadków nasz Config Serwer zadba o to by każdy mikroserwis został właściwie skonfigurowany.

Według konwencji portem domyślnym dla tego serwera jest port nr 8888. Tworzymy sobie prosty mikroserwis: „limits-service”. Po ustawieniu będzie on zawierał swoje propertisy, które następnie zostaną nadpisane konfiguracją ze SpringCloudConfigServer-a. Plan jest taki:

Obraz zawierający obiekt

Opis wygenerowany przy bardzo wysokim poziomie pewności

Co to za CurrencyCalculation i CurrencyExchange będzie potem.

LimitsService propertiesFile na początku zawiera:

spring.application.name=limits-service

i defaultowy port = 8080

Tworzymy klasę, która będzie zawierała pola minimum i maximum reprezentujące przykładowe wartości obiektu LimitConfiguration:

Obraz zawierający zrzut ekranu

Opis wygenerowany przy bardzo wysokim poziomie pewności

Oraz przykładowy kontroler zwracający nam obiekt LimitConfiguration.

Obraz zawierający zrzut ekranu

Opis wygenerowany przy bardzo wysokim poziomie pewności

Na razie bez żadnych Rocket Science 😊

Feature: SpringBoot pozwala nam na odczyt danych konfiguracyjnych bezpośrednio z pliku properties i do tego właśnie dążymy, aby móc konfigurować mikroserwisy na podstawie danych z Config Serwera, które znajdują się właśnie w **plikach properties.**

**Dodajmy sobie dwie dane konfiguracyjne do pliku properties w mikroserwisie limits-service:**

limits-service.minimum=99

limits-service.maximum=9999

Stwórzmy nową klasę, gdzie za pomocą prostej adnotacji zadzieje się magia 😉

Obraz zawierający zrzut ekranu

Opis wygenerowany przy bardzo wysokim poziomie pewności

Adnotacja @ConfigurationProperties pozwala nam na wskazanie, który plik konfiguracyjny powinien być użyty. W nawiasie podajemy prefix z pomocą którego tworzyliśmy dane konfiguracyjne w pliku properties.

Wstrzyknijmy teraz naszą magiczną klasę do kontrolera obsługującego naszą konfigurację. Uzyskamy to za pomocą @Autowired, a następnie odwołując się do pól z tej klasy.

Obraz zawierający zrzut ekranu

Opis wygenerowany przy bardzo wysokim poziomie pewności

Hurra! Nasz serwis jest w pełni gotowy do czytania danych konfiguracyjnych bezpośrednio ze swojego pliku properties.

Teraz możemy postawić nasz Config Serwer na nogi 😊

Więc, tworzymy nowy mikroserwis, który potem dostanie skromny upgrade. Na razie zainicjujmy nowy projekt o nazwie spring-cloud-config-server.

PS. Dodajmy do mavena zależność, która potem nam się przyda:

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-config-server</artifactId>

</dependency>

I ustawmy mu nazwę oraz port:

spring.application.name=spring-cloud-config-server

server.port=8888

**TERAZ INSTALUJEMY GITA**

<https://git-scm.com/>

- Tworzymy lokalne repozytorium

- Tworzymy w nim pliki tekstowe, które posłużą jako konfiguracja (dajemy im rozszerzenie properties)

- Wycinamy z .properties serwisu limits-service dane konfiguracyjne (limits-service.minimum oraz limits-service.maximum) i przeklejamy je do nowo utworzonego w repozytorium pliku limits-service.properties (nazwa pliku – limits-service oznaczać będzie defaultowy plik konfiguracyjny). Ustawiamy je np. na minimum=8 oraz maximum=888 – tak było w tutku 😊

- Commitujemy zmiany w naszym lokalnym repozytorium

Teraz wypada wskazać naszemu serwerowi miejsce, w którym znajduje się cała nasza konfiguracja czyt. Ścieżkę do naszego lokalnego repo.

Możemy to uzyskać dodając do naszego pliku konfiguracyjnego

spring.cloud.config.server.git.uri=**file://**/ścieżka do lokalnego repo z plikami konfiguracyjnymi

U mnie:

**spring.cloud.config.server.git.uri=** <file:///C:/Users/181395/Desktop/microservices/spring-cloud-config-server/git-localconfig-repo>

* **file://** <- to składniowo jest ważne celem wskazania dobrej ścieżki!!!

Następnie, żeby zadziała się magia Spring Clouda należy uruchomić nasz mikroserwis w trybie Config Serwera. Możemy to uzyskać dodając do naszej klasy bootstrapującej serwer adnotację @EnableConfigServer.

Żeby sprawdzić poprawność tej części należy strzelić do localhost:8888/limits-service/default co oznacza localhost:{port na którym odpalony jest serwer}/{nazwa usługi}/default

**Teraz to na co wszyscy czekali czyli multikonfiguracja.**

Więc, stwórzmy sobie dwa dodatkowe pliki konfiguracyjne – niech będzie jak w tutku –

limits-service-dev.properties oraz limits-service-qa.properties i nadajmy im odpowiednio minimum=1 lub 2 oraz maximum = 111 lub 222

**ZACOMMITUJMY NOWE PLIKI .PROPERTIES**

Dla testu możemy się odwołać: **localhost:8888/limits-service/qa** , **localhost:8888/limits-service/dev**

Cel jest taki, aby mikroserwis limits-service był konfigurowany przez Spring Cloud Config Server z poziomu gita. Należy więc teraz zmienić nazwę pliku .properties dla limits-service na bootstrap.properties (bo tak 😊 ) oraz dodać do niego adres serwera poprzez dodanie:

**spring.cloud.config.uri=http://localhost:8888** (u nas jest to 8888)

Żeby ładnie móc konfigurować serwisy w zależności od np. środowiska należy w jakiś cudowny sposób wskazać im czym mają być konfigurowane. Uzyskać to możemy poprzez wyspecyfikowanie w propertiesach profilu dla jakiego mają zostać zaczytane ustawienia.

Otrzymamy to poprzez dodanie:

**spring.profiles.active=dev** lub **spring.profiles.active=qa**

Na tej podstawie określamy z którego pliku dane konfiguracyjne Spring Cloud Config Server powinien pobrać dla naszego limits-service.

**\*Żeby wprowadzać zmiany w plikach konfiguracyjnych znajdujących się na repo i widzieć je należy te zmiany commitować 😊**

JUPI!!! W TAKI OTO CUDOWNY SPOSÓB UDAŁO SIĘ NAM WSPÓLNIE STWORZYĆ MIKROSERWIS KONFIGUROWANY DZIĘKI SPRING CLOUD CONFIG SERVEROWI Z POZIOMU GITA. ALE PO CO NAM TO, TAK WŁAŚCIWIE?

Obraz zawierający obiekt

Opis wygenerowany przy bardzo wysokim poziomie pewności

W tym momencie wystarczy, że stworzymy sobie np. CurrencyExchangeService i wskażemy mu tak jak w limitsService:

Obraz zawierający zrzut ekranu

Opis wygenerowany przy wysokim poziomie pewności

Nazwę, uri config serwera oraz profil z jakiego ma zczytywać sobie dane konfiguracyjne et voila – podłączyliśmy sobie kolejny mikroserwis do naszej zewnętrznej konfiguracji.

# Load Balancing

## Ribbon

## Feign

# Naming Server – Eureka

# API Gateway – Zuul

# Rozproszone śledzenie

## Zipkin

## Sleuth

# Odporność na uszkodzenia – Hystrix