Spring Cloud

Spring Cloud

- Wprowadzenie
- Spring Cloud
  - Eureka
  - Config
  - Ribbon
  - Feign
  - Hystrix
  - Bus
  - API Gateway

Spring Cloud

- **Wprowadzenie**
- Spring Cloud
  - Eureka
  - Config
  - Ribbon
  - Feign
  - Hystrix
  - Bus
  - API Gateway

Wprowadzenie

- Spring / Spring IO / Spring Cloud
- Spring Cloud Netflix
- Spring Cloud & Spring Boot
- Serwer vs Klient

Wprowadzenie

- **Spring / Spring IO / Spring Cloud**
- Spring Cloud Netflix
- Spring Cloud & Spring Boot
- Serwer vs Klient

Wprowadzenie

- **Początki** - Spring Framework (**2004**)
- **od 2006** powstało wiele pod-projektów - zebrane wszystkie jako **Spring IO**
- **Spring Cloud** to **zbiór projektów**
- to **biblioteki** odpowiadające na wyzwania rozproszonych aplikacji (**Centralized Config, Service Registration / Discovery, Load Balancer etc.**)

Spring Cloud

Wprowadzenie

- Spring / Spring IO / Spring Cloud
- **Spring Cloud Netflix**
- Spring Cloud & Spring Boot
- Serwer vs Klient

Wprowadzenie

- w 2007, z biznesu związanego z DVD, **NETFLIX** ‘wymyślił się’ na nowo - z kluczowego klienta UPS, został największym ‘generatorem’ ruchu internetowego (wieczory) w Ameryce Północnej
- **pionierskie rozwiązania w Chmurze**
- **podzielił się ze światem** pomysłami / implementacjami (stream’owanie nadal tajemnicą)

Wprowadzenie

- **Netflix OSS** to **dojrzałe** i **sprawdzone** w boju rozwiązanie
- dobre biblioteki, ale niełatwo się w nie wgryźć
- **Spring Cloud bardzo ułatwia życie** - możemy korzystać z bibliotek tylko za pomocą adnotacji i dodania do projektu zależności

Wprowadzenie

- Spring / Spring IO / Spring Cloud
- Spring Cloud Netflix
- **Spring Cloud & Spring Boot**
- **Serwer vs Klient**

Wprowadzenie

- **Projekty** Spring **Cloud bazują** na Spring **Boot**
- **bez** Spring **Boot**, **trudno** używać
  - SB **ułatwia** zarządzanie zależnościami
  - ma zmodyfikowany start ApplicationContext

Spring Cloud

Wprowadzenie

- Klient i Serwer to **niejednoznaczne** pojęcia
- mikroserwis jest **często jednym i drugim**
- to czym właściwie jest, **zależy od roli w relacji** z innymi

Spring Cloud

Wprowadzenie

- **Spring Cloud** to **część** Spring **IO**
- sam jest **zbiorem projektów**
- **jest odpowiedzią na wyzwania** aplikacji **rozproszonych**
- **ułatwia** użycie bibliotek **Netflix'a**
- **bazuje** na **Spring Boot**

Spring
Cloud

Spring Cloud

- Wprowadzenie
- **Spring Cloud**
  - **Eureka**
  - Config
  - Ribbon
  - Feign
  - Hystrix
  - Bus
  - API Gateway

Spring Cloud Eureka

- Passive Service Discovery
- konfiguracja serwera
- konfiguracja klienta

Spring Cloud

Spring Cloud Eureka

- **Passive Service Discovery**
- konfiguracja serwera
- konfiguracja klienta

Spring Cloud

Passive Service Discovery

- service discovery - **analogia**
- **czat**
- zalogowany użytkownik ‘rejestruje się’ na serwerze
- serwer dostarcza listę innych użytkowników
- użytkownik dowiedział się o innych, inni dowiedzieli się o nim

Spring
Cloud

Passive Service Discovery

- **Eureka**
  - **część** blibliotek **Netflix**
    - sprawdzone w boju
  - **dostarcza serwer**
    - klienci rejestrują się na serwerze
      - przekazują metadane (host, port, etc.)
    - klienci ‘pulsują’ (heartbeats) do serwera
      - serwer usuwa serwery bez pulsu

Spring Cloud

Spring Cloud Eureka

- Passive Service Discovery
- **konfiguracja serwera**
- konfiguracja klienta

Spring Cloud

Konfiguracja serwera

- **dodanie zależności** do projektu
- **@EnableEurekaServer** w Main
- na produkcji: wiele instancji
  - replikowane instancji zapewnia dostępność usług

Spring Cloud

Spring Cloud Eureka

- Passive Service Discovery
- konfiguracja serwera
- **konfiguracja klienta**

Spring Cloud

Konfiguracja klienta

- **dodanie zależności** do projektu
- **@EnableDiscoveryClient** w Main
- w application.properties
  - **eureka.client.serviceUrl.defaultZone**: http:// …

Spring Cloud

Konfiguracja klienta

- @EnableDiscoveryClient
  - **automatycznie rejestruje klienta** na serwerze
  - rejestruje nazwę aplikacji, host, port
    - **spring.application.name**

Spring Cloud

Podsumowanie

- **Passive Service Discovery**
  - serwisy **automatycznie rejestrują się** i ‘odnajdują’
- Spring Cloud Eureka **Server**
  - ‘**wrapper**’ dla Netflix Eureka
  - odpowiada za **rejestr klientów**
  - **współdzieli** informację **z innymi** instancjami
- Spring Cloud Eureka **Client**
  - **rejestruje się** na serwerze, ‘ujawnia’ i **uzyskuje info** o innych

Spring Cloud

Spring Cloud

- Wprowadzenie
- **Spring Cloud**
  - Eureka
  - **Config**
  - Ribbon
  - Feign
  - Hystrix
  - Bus
  - API Gateway

Spring Cloud Config

- zarządzanie konfiguracją
  - czym jest
  - wyzwania
  - oczekiwany stan
- konfiguracja serwera
- konfigruacja klienta

Spring Cloud

Spring Cloud Config

- zarządzanie konfiguracją
- konfiguracja serwera
- konfiguracja klienta
- jak to działa

Spring Cloud

Spring Cloud Config

- **zarządzanie konfiguracją**
- konfiguracja serwera
- konfiguracja klienta
- jak to działa

Spring Cloud

Zarządzanie konfiguracją

- **aplikcja** to **więcej niż** sam **kod**
- zazwyczaj nie działa w próżni
- łączy się zasobami i innymi aplikacjami

Spring Cloud

Zarządzanie konfiguracją

- **sposoby konfiguracji** aplikacji
  - pliki konfiguracyjne **spakowane z aplikacją**
    - wymaga restartu
  - pliki konfiguracyjne **w filesystem**
    - niedostępne w chmurze
- użycie **zmiennych systemowych**
  - różnice między platformami
  - duża #
- **użycie rozwiązań** zapewnianych przez dostawcę **Chmury**
  - wiąże z danym vendorem

Zarządzanie konfiguracją

- dodatkowe **wyzwania**
  - mikroserwisy to zazwyczaj **duża #** niezależnych serwisów
  - potrzeba **dynamicznych zmian**
  - potrzeba **kontroli wersji**

Spring Cloud

Zarządzanie konfiguracją

- porządane rozwiązanie powinno zapewnić:
  - **niezależność** od platformy
  - **centralizację**
  - możliwość **update w trakcie działania**
  - **zarządzanie wersjami**
  - **pasywność** - aplikacje same powinny się rejestrować / robić update zmian

Spring Cloud

Zarządzanie konfiguracją

- Spring Cloud **Config**
  - **scentralizowane zarządzanie** konfiguracją
- Spring Cloud **Bus**
  - sposób **notyfikacji klientów**
- Spring Cloud Netflix **Eureka**
  - aplikacje mogą się **rejestrować** i nawzajem **odnajdywać**

Spring Cloud

Spring Cloud Config

- **centralny serwer** zawiera informacje dla klientów
  - konfiguracja w Git, systemie plików itp.
- **klienci** łączą się po HTTP i **pobierają konfigurację**
  - dodatek / nadpisanie ich wewnętrznej konfiguracji

Spring Cloud

Spring Cloud Config

- zarządzanie konfiguracją
- **konfiguracja serwera**
- konfiguracja klienta
- jak to działa ?

Spring Cloud

Konfiguracja serwera

- **dodanie zależności** do projektu Spring Boot
- **konfiguracje** w plikach application.yml / prop.
- **@EnableConfigServer** w klasie głównej

Spring Cloud

Spring Cloud Config

- zarządzanie konfiguracją
- konfiguracja serwera
- **konfiguracja klienta**
- jak to działa ?

Spring Cloud

Konfiguracja serwera

- **dodanie zależności** do projektu Spring Boot
- bootstrap.yml
  - **spring.cloud.config.uri**: http://…

Spring Cloud

Spring Cloud Config

- zarządzanie konfiguracją
- konfiguracja serwera
- konfiguracja klienta
- **jak to działa ?**

Spring Cloud

Jak to działa?

- jak ładowane są wartości z konfiguracji?
  - Spring ma obiekt **Environment**
  - Environment ma w sobie PropertySources
    - ładowane ze zmiennych systemowych, JNDI, pliki etc.
  - Spring Cloud Config **dodaje** kolejny **PropertySource**
    - łącząc się po HTTP
    - http://<serwer>:<port>/<spring.app.nazwa>/<profil>

Spring Cloud

Spring Cloud Config

- co z klientami **nie napisanymi** w **Javie**?
- serwer wystawia dane w **prostym interfejsie** HTTP
- **łatwe do wywołania** z innych technologi, nie tak zautomatyzowane jak w Spring'u

Spring Cloud

Spring Cloud Config

- co, **gdy serwer nie odpowiada**?
  - na produkcji będziemy mieli **kilka instancji**
- **klienci mogą kontrolować** zachowanie
  - **spring.cloud.config.failFast**=true
  - domyślny: false
  - serwer nadpisuje lokalne: można zapewnić domyślne

Spring Cloud

Cloud Config - podsumowanie

- trudno o ręczne zarządzanie setkami instancji
- Spring Cloud Config - sprawdzone narzędzie

Spring Cloud

Spring Cloud

- Wprowadzenie
- Spring Cloud
  - Eureka
  - Config
  - **Ribbon**
  - Feign
  - Hystrix
  - Bus
  - API Gateway

Spring Cloud Netflix Ribbon

- Client-Side Load Balancing
- Spring Cloud Netflix Ribbon

Spring Cloud

Spring Cloud Netflix Ribbon

- **Client-Side Load Balancing**
- Spring Cloud Netflix Ribbon

Spring Cloud

Client-Side Load Balancing

- Load Balancer
  - (**tradycyjnie**) server-side
    - **przekierowuje ruch** pomiędzy **serwerami**
      - software, lub sprzęt
  - client-side
    - **wybiera**, z którym serwerem się łączyć
    - w oparciu o pewne **kryteria**
    - jest częścią **soft**'u **klienta**
    - nie wyklucza użycia server-side

Client-Side Load Balancing

- dlaczego?
  - serwer może być
    - **niedostępny** (błędy, awarie)
    - **wolniejszy** niż inne (wydajność)
    - **bardziej oddalony** (regiony)

Spring Cloud

Spring Cloud Netflix Ribbon

- Client-Side Load Balancing
- **Spring Cloud Netflix Ribbon**

Spring Cloud

Spring Cloud Netflix Ribbon

- **Ribbon** - kolejny członek rodziny **Netflix OSS**
  - client-side load balancer
  - automatycznie zintegrowany z **Eureka**
  - wbudowana ’odporność’ na błędy (**Hystrix**)
  - cache
  - wiele protokołów (HTTP, TCP, UDP)
- Spring Cloud dostarcza kolejny, wygodny ’wrapper’

Spring Cloud Netflix Ribbon

- Ribbon - kluczowe pojęcia
  - lista serwerów
  - odfiltrowana lista serwerów
  - Ping
  - Load Balancer

Spring Cloud

Spring Cloud Netflix Ribbon

- Ribbon - kluczowe pojęcia
  - **lista serwerów**
  - odfiltrowana lista serwerów
  - Ping
  - Load Balancer

Spring Cloud

Spring Cloud Netflix Ribbon

- lista serwerów
- wskazuje **dostępne serwery** dla klienta
- **statyczna** (ustawiana z konfiguracji w pliku)
- **dynamiczna** (ustawiana dzięki Eureka)

Spring Cloud

Spring Cloud Netflix Ribbon

- Ribbon - kluczowe pojęcia
  - lista serwerów
  - **odfiltrowana lista serwerów**
  - Ping
  - Load Balancer

Spring Cloud

Spring Cloud Netflix Ribbon

- **odfiltrowana** lista serwerów
- na podstawie określonych kryteriów
- **ograniczenie dostępnych** serwerów
- domyślnie - zone

Spring Cloud

Spring Cloud Netflix Ribbon

- Ribbon - kluczowe pojęcia
  - lista serwerów
  - odfiltrowana lista serwerów
  - **Ping**
  - Load Balancer

Spring Cloud

Spring Cloud Netflix Ribbon

- Ping
  - sprawdzenie, czy serwer odpowiada
  - domyślnie - delegowane do Eureki

Spring Cloud

Spring Cloud Netflix Ribbon

- Ribbon - kluczowe pojęcia
  - lista serwerów
  - odfiltrowana lista serwerów
  - Ping
  - **Load Balancer**

Spring Cloud

Spring Cloud Netflix Ribbon

- Load Balancer
- komponent odpowiadający za przekierowania
- dostępne różne strategie

Spring Cloud

Spring Cloud Netflix Ribbon

- użycie Ribbon API
- programistycznie
- lub deklaratywnie (preferowane)
    - adnotacje

Spring Cloud

Ribbon - podsumowanie

- Client-Side Load Balancing
  - daje klientowi możliwość wyboru serwera
  - rozszerza możliwości tradycyjnego lb
- Spring Cloud Ribbon to wygodna w użyciu implementacja

Spring Cloud

Spring Cloud

- Wprowadzenie
- Spring Cloud
  - Eureka
  - Config
  - Ribbon
  - **Feign**
  - Hystrix
  - Bus
  - API Gateway

Spring Cloud Feign

- Feign - wprowadzenie
- Feign - użycie

Spring Cloud

Spring Cloud Feign

- **Feign - wprowadzenie**
- Feign - użycie

Spring Cloud

Feign - czym jest?

- **deklaratywny klient** REST od Netflix'a
- pozwala na tworzenie **REST bez implementacji**
- łatwiejsza alternatywa dla RestTemplate
- Spring Cloud zapewnia kolejny 'wrapper'

Spring Cloud

Spring Cloud Feign

- Feign - wprowadzenie
- **Feign - użycie**

Spring Cloud

Feign - użycie

- **zdefiniuj interfejs** z adnotacją @FeignClient
- dodaj **adnotacje** Spring MVC **do metod**
- **Spring Cloud** utworzy **w runtime** implementacje
- @EnableFeignClients przeskanuje interfejsy
- **zaimplementuje kod**

Feign - użycie

- zależność Feign
- wymagana w runtime
- niewymagana w compile

Spring Cloud

Feign, Ribbon i Eureka

- hardcode url w adnotacji
  - **@FeignClient(url=”localhost:8080/hello”)**
- użycie ID klienta z Eureki
  - **@FeignClient(”HELLO”)**

Spring Cloud

Feign - podsumowanie

- Feign ułatwia wywoływanie serwisów
- automatycznie integruje się z Eureka i Ribbon

Spring Cloud

Spring Cloud

- Wprowadzenie
- Spring Cloud
  - Eureka
  - Config
  - Ribbon
  - Feign
  - **Hystrix**
  - Bus
  - API Gateway

Spring Cloud

Spring Cloud Hystrix

- cascade failure & circuit breakers
- Spring Cloud Netflix Hystrix

Spring Cloud

Spring Cloud Hystrix

- **cascade failure & circuit breakers**
- Spring Cloud Netflix Hystrix

Spring Cloud

Cascade failure & circuit breakers

- **duża # serwisów**, będących zależnościami
  - to **idealne warunki do** kaskadowych **błędów**
- systemy rozproszone - dużo czynników sprzyjającym porażce
- pamiętamy The Fallacies of Distributed Computing?

Spring Cloud

Cascade failure & circuit breakers

- The Circuit Breaker Pattern
  - podobnie jak bezpiecznik w domu
    - **wyłącza przepływ**
    - gdy problem rozwiązany, można go włączyć
    - zapobiega większym problemom
      - jak zwykły bezpiecznik przed pożarems domu
- błąd w jednym serwisie, może wywrócić kilkanaście innych

Spring Cloud Hystrix

- cascade failure & circuit breakers
- **Spring Cloud Netflix Hystrix**

Spring Cloud

Spring Cloud Netflix Hystrix

- **Hystrix** - część **Netflix OSS**
- Spring Cloud Hystrix - ’wrapper’ dostarczony przez Spring Cloud
  - w przypadku spełnienia warkunku błędu ‘otwiera się’
    - metoda fallback
      - co robić w przypadku błędu
      - coś jak catch, ale bardziej wysublimowany
- automatycznie ’zamyka się’
- ‘otwieranie’ / ’zamykanie’
  - domyślne zachowanie / custom

Spring Cloud

Spring Cloud Hystrix

- konfiguracja
- dodaj zależność
- @EnableHystrix w Main

Spring Cloud

Spring Cloud Hystrix

```
@HystrixCommand(
    fallbackMethod = "myFallback",
    commandProperties = {
      @HystrixProperty(name="property.hystrix.dla.zmiany.default", value="20")
}
public Object methodWithPotentialProblems() {
}


private Object myFallback(){

}
```


Spring Cloud

Podsumowanie

- programowe ’bezpieczniki’ chronią aplikację rozproszoną
- Spring Cloud Netflix Hystrix zapewnia wygodny ’wrapper

Spring Cloud

Spring Cloud

- Wprowadzenie
- Spring Cloud
  - Eureka
  - Config
  - Ribbon
  - Feign
  - Hystrix
  - **Bus**
  - API Gateway

Spring Cloud

Spring Cloud Bus

- Problem z odświeżaniem konfiguracji
- Spring Cloud Bus
- Konfiguracja
- Jak działa dynamiczne odświeżanie

Spring Cloud

Spring Cloud Bus

- **Problem z odświeżaniem konfiguracji**
- Spring Cloud Bus
- Konfiguracja
- Jak działa odświeżanie w Spring'u

Spring Cloud

Problem z odświeżaniem konfiguracji

- Spring Cloud Config - przypomnienie
  - **klienci** łączą się z serwerem i **pobierają konfig.**
    - robią to **podczas** swojego **startu**
- co w przypadku **zmiany konfiguracji**?
  - klienci mogą okresowo sprawdzać zmiany
  - lepiej - **broker wiadomości**

Spring Cloud

Spring Cloud Bus

- Problem z odświeżaniem konfiguracji
- **Spring Cloud Bus**
- Konfiguracja
- Jak działa odświeżanie w Spring'u

Spring Cloud

Spring Cloud Bus

- przesyła zmiany konfiguracji do klientów
- klienci zostają subskrybentami zmian
- wykorzystana technologia AMQP (RabbitMQ)

Spring Cloud

Spring Cloud Bus

- Problem z odświeżaniem konfiguracji
- Spring Cloud Bus
- **Konfiguracja**
- Jak działa odświeżanie w Spring'u

Spring Cloud

Konfiguracja

- **dodaj zależność** do projektów
  - Spring Cloud Config Server
  - subskrybentów
- **uruchom serwer** AMQP (**RabbitMQ**)
  - Bus automatycznie konfiguruje się z Rabbit na localhost
- zmień plik(i) konfig.
- **komunikat POST** do jednej instancji klienta odświeża pozostałe

Spring Cloud

Spring Cloud Bus

- Problem z odświeżaniem konfiguracji
- Spring Cloud Bus
- Konfiguracja
- **Jak działa odświeżanie w Spring'u**

Spring Cloud

Jak działa odświeżanie?

- @ConfigurationProperties
- @RefreshScope

Spring Cloud

Jak działa odświeżanie?

- @ConfigurationProperties
  - pojawiło się wraz z Spring Boot
  - alternatywa dla wielu @Value
- POST odświeża prop’sy

Spring Cloud

Jak działa odświeżanie?

- @RefreshScope
  - POST przeładowuje całe ziarno
    - efekt uboczny - lazy bean
- Spring tworzy proxy, wstrzykiwane jako zależność
  - odświeżenie, powoduje tworzenie nowego ziarna
  - po utworzeniu proxy wskazuje na nowe ziarno

Spring Cloud

Spring Cloud

- Wprowadzenie
- Spring Cloud
  - Eureka
  - Config
  - Ribbon
  - Feign
  - Hystrix
  - Bus
  - **API Gateway**

Spring Cloud

API Gateway

- API Gateway - potrzeba
- Spring Cloud Netflix Zuul

Spring Cloud

Spring Cloud Netflix Zuul

- **API Gateway - potrzeba**
- Spring Cloud Netflix Zuul

Spring Cloud

API Gateway - potrzeba

- serwisy wystawione ’na zewnątrz’
  - wystawienie wewnętrznego API - problem z nieodpowiednim użyciem
  - problemy z bezpieczeństwem
- klient
  - większa ilość requestów
  - Cross-Origin Resource Sharing
  - różni klienci (Web, mobile etc.) różne potrzeby

Spring Cloud

Spring Cloud Netflix Zuul

- API Gateway
  - fasada serwisów
  - custom API - dla konkretnego klienta
  - ogranicza # requestów (agregacja)
  - obsługuje
    - bezpieczeństwo, cache, tłumaczenie protokołów etc.

Spring Cloud

Spring Cloud Netflix Zuul

- API Gateway - potrzeba
- **Spring Cloud Netflix Zuul**

Spring Cloud Netflix Zuul

- Zuul - JVM’owy router i LB
  - użycie
    - dodanie zależności
      - spring-cloud-starter-zuul zawiera Ribbon i Hystrix
    - w Main @EnableZuulProxy

Spring Cloud