Pertemuan 11 – Konfigurasi Lingkungan Spring

Konfigurasi Otomatis Spring

- Spring Boot melakukan konfigurasi secara otomatis. Konfigurasi otomatis menyederhanakan pengembangan aplikasi Spring.
- Untungnya, Spring Boot menyediakan sebuah cara dengan properti konfigurasinya. Properti konfigurasi tidak lebih dari properti dari bean di Spring Fine-Tuning Autoconfiguration

Fine-tuning autoconfiguration

- Ada dua cara yang berbeda dari jenis konfigurasi di Spring
  - Bean wiring—Konfigurasi yang mendeklarasikan komponen aplikasi yang akan dibuat
  - Property injection—Konfigurasi yang mengeset nilai bean di dalam konteks aplikasi Spring

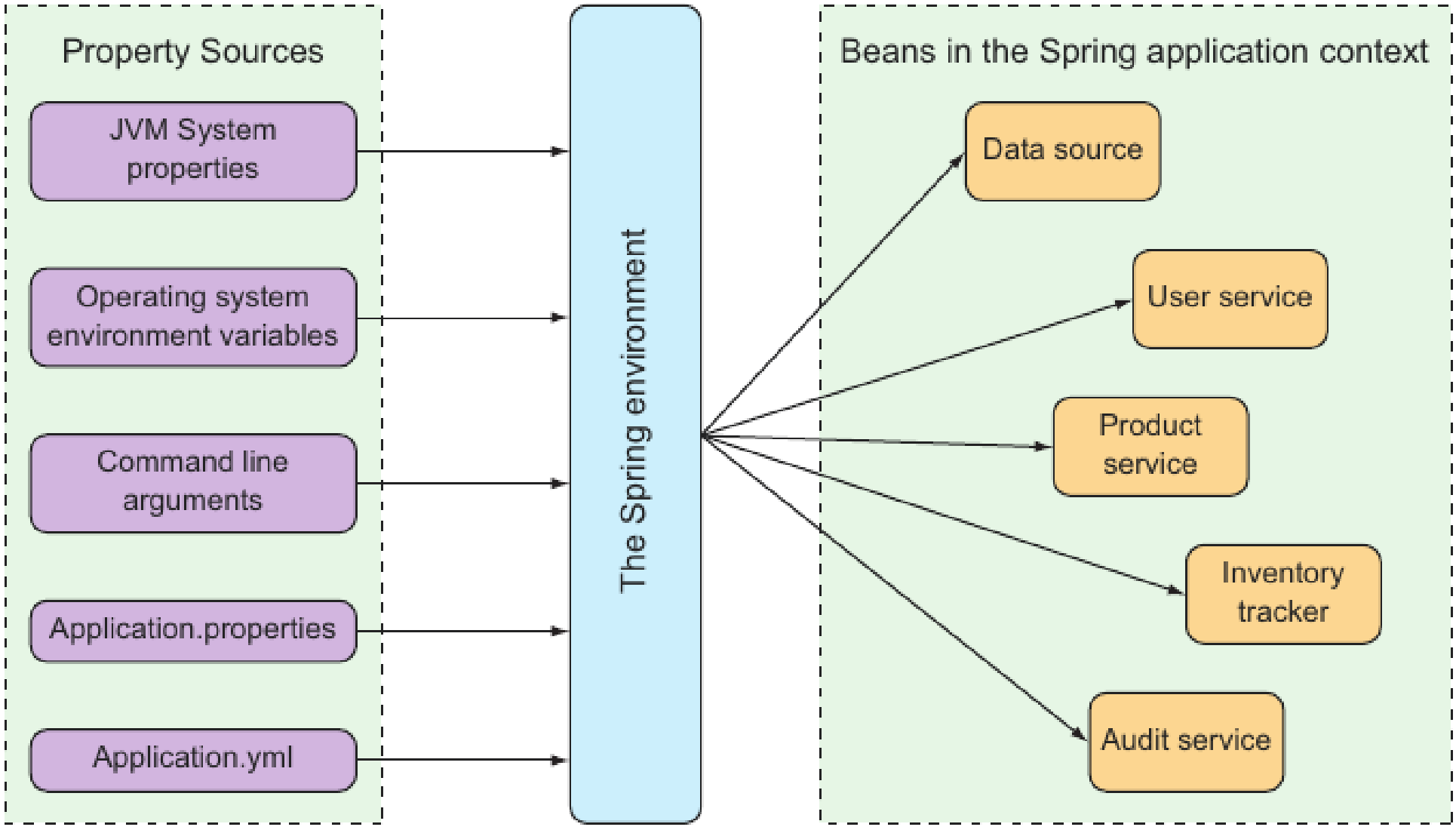
Memahami Lingkungan Abstraksi

- Lingkungan Abstraksi Spring adalah seperti toko untuk semua properti konfigurasi
- Dia melakukan abstraksi asal dari tiap properti jadi bean yang memerlukan properti itu dapat mengaksesnya dari Spring itu sendiri

Lanjutan

- Lingkungan Spring menarik beberapa properti dari beberapa sumber termasuk:
  - JVM system properties
  - Operating system environment variables
  - Command-line arguments
  - Application property configuration files

Lanjutan


Property Sources
JVM System properties
Operating system environment variables
Command line arguments
Application.properties
Application.yml
The Spring environment
Beans in the Spring application context
Data source
User service
Product service
Inventory tracker
Audit service

Konfigurasi Sumber Data

Pada dasar nya menggunakan database internal seperti H2 merupakan hal yang tepat (untuk sekarang). Namun nantinya ketika memasuki tahap produksi sebaiknya menggunakan DB Eksternal

Lanjutan

Meskipun kita dapat mengkonfigurasi secara eksplisit melalui bean yang biasanya tidak perlu.

Sebagai ganti, ada sebuah cara lebih simpel untuk mengkonfigurasi URL dan keamnanan database via configuration properties.

Lanjutan

Sebagai contoh, jika kita mulai menggunakan MySQL. Maka sebaiknya menambahkan konfigurasi tambahan ke application.yml:

```
spring:
    datasource:
        url: jdbc:mysql://localhost/tacocloud
        Username:tacodb
        password:tacopassword
```

Lanjutan

- Spring Boot menggunakan koneksi data ini ketika melakukan konfigurasi otomatis DataSource bean.
- DataSource bean akan dikumpulkan menggunakan JDBC Tomcat connection pool jika tidak tersedia di kelasnya

Konfigurasi Server Embed

- Meskipun kita secara eksplisit mengkonfigurasikan server.port ke 0, server tidak akan jalan di port 0.
- Sebagai gantinya, dia akan memilih port secara acak. Hal ini berguna ketika menjalankan pengujian integrasi otomatis untuk memastikan tes tidak bertabrakan dengan port khusus

Lanjutan

- Selain server port, ada sebuah hal yang umum yang perlu dilakukan untuk mengatur permintaan HTTPS.
- Untuk melakukannya kita perlu membuat sebuah keystore menggunakan perintah JDK Keytool
- keytool -keystore mykeys.jks -genkey -alias tomcat -keyalg RSA

Lanjutan

- Kita akan ditanyai beberapa pertanyaan seperti nama, dan organisasi yang sebagian besarnya tidak relevan. Tetapi ketika dianyai soal password, ingatlah apa yang telah dipilih.

```
server:
    port: 8443
        ssl:
            key-store: file:///path/to/mykeys.jks
            key-store-password: letmein
            key-password: letmein
```

Konfigurasi Logging

- Sebagian besar aplikasi menyediakan sebuah catatan khusus (logging). Jika tidak berarti aplikasi tersebut memiliki cara khusus untuk mencatat aktivitas yang dilakukan oleh aplikasi tersebut
- Secara default, Spring Boot melakuakn konfigurasi logging via Logback (http://logback.qos.ch) untuk menulis di konsole

Konfigurasi Sendiri

- Properti konfigurasi tidak lebih dari properti bean yang di desain untuk menerima konfigurasi dari Lingkungan Abstraksi Spring
- Untuk mendukung injeksi properti dari properti konfigurasi, Spring Boot menyediakan anotasi @ConfigurationProperties. Ketika diletakkan di Spring Bean manapun, dia akan melakukan spesifikasi dari properti bean

Menggunakan Profil

- Ketika aplikasi dirilis di lingkungan pengembangan yang berbeda (Sistem Operasi, IDE editor, dll) pastinya konfigurasi yang dibutuhkan akan berbeda
- Oleh karena itu profil digunakan untuk membedakan konfigurasi di lingkungan yang berbeda tersebut

Contoh

- Sebagai contoh:
  - Ketika kita mengembangkan aplikasi, kita hanya memerlukan database H2 dan logging level INFO
  - Namun ketika kita ingin merilis aplikasi yang telah dibuat, kita memerlukan database MySQL dan logging dengan level yang lebih dalam (WARN)

Mendefinisikan Profil

- Profil dapat didefinisikan dengan cara mudah yaitu melalui application.yaml:

```
spring:
    profiles:prod
    datasource:
        url: jdbc:mysql://localhost/tacocloud
        username: tacouser
        password: tacopassword
    logging:
        level:
            tacos:WARN
```