Technical Session

Java

Topic

Introduction to Spring Boot

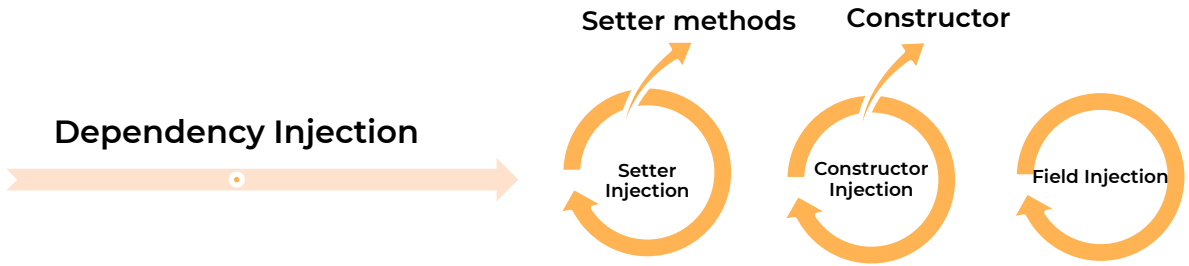
Kwabena Aboagye Frank **(K.A.F)**
Engineer **(B.E at this point)**

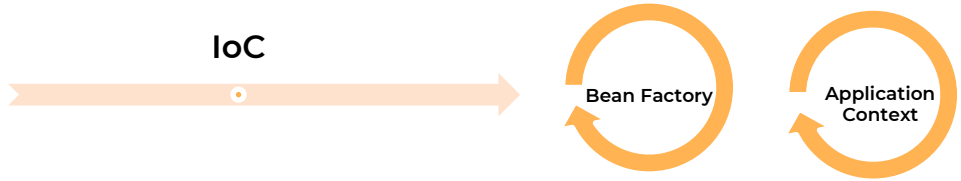
AGENDA

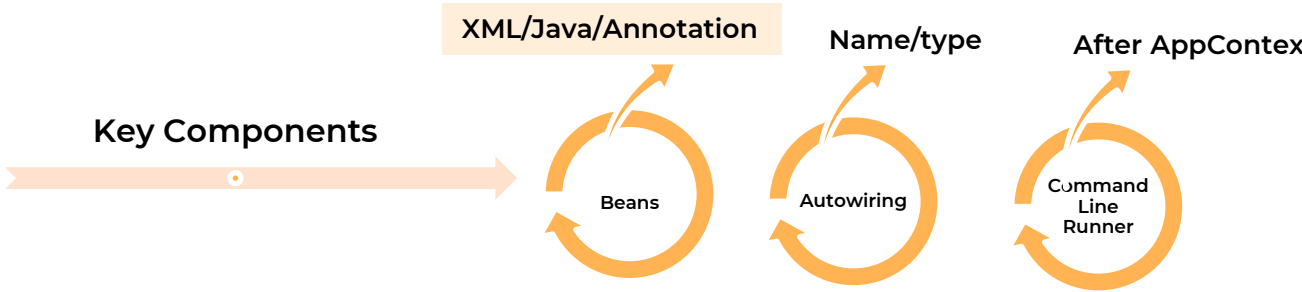
- ➔ Intro (+ review)
- ➔ Spring Boot Basic Intro
- ➔ Annotations
- ➔ Spring Profiles + Configuring External Properties
- ➔ (All + Demo)

General Into + review

Last week


Dependency Injection
Setter methods
Constructor
Setter Injection
Constructor Injection
Field Injection


IoC
Bean Factory
Application Context


XML/Java/Annotation
Name/type
After AppContex
Key Components
Beans
Autowiring
Command Line Runner

Spring Boot Intro

History Of Spring

Spring Boot?
- Opinionated view of spring
- Handles predictable setup

Features?
- Dependency Management
- Auto-Configuration
- Packaging and Runtime
- Integration Testing

Starters to the rescue?
- Parent POM
- Starter Dependencies

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.7.5</version>
</parent>
```

Defines properties for dependencies, for example:
`${spring-framework.version}= 5.3.23`

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter</artifactId>
```

About 18 Dependencies?
- spring-boot-*.jar
- spring-context-*.jar
- spring-core-*.jar
- spring-beans-*.jar
- ....

Explore with start.spring.io

Annotations

Typical Spring App?
- Manual Setup

Some Spring Core Annotations?
- @Component
- @Controller
- @Autowired
- @Qualifer
- @Primary
- @Aspects
- @Bean
- @PropertySource
- ...

Typical Application?
- Combine them

Some Springboot Core Annotations?
- @SpringBootApplication
- @EnableAutoConfiguraton
- @RestController
- @RequestMapping
- @ResponseBody
- @PathVariable
- @Entity
- @SpringBootTest
- ...

Annotation	Spring Boot	Spring Framework
`@SpringBootApplication`	`@Configuration`, `@EnableAutoConfiguration`, `@ComponentScan`	manual setup using multiple annotations
`@EnableAutoConfiguration`	Automatically configures Spring application based on dependencies in the classpath.	Not available. All configurations are manual.
`@RestController`	`@Controller`, `@ResponseBody`	Requires both `@Controller` and `@ResponseBody`.
`@ComponentScan`	Scans for components, configurations, and services automatically.	Same, but lack of auto-configuration
`@Value`	Injects properties	Same functionality - explicit setup
`@ConditionalOnProperty`	Loads a bean conditionally	programmatic conditions.

(Demo- Bootstrap) Spring Profiles + Configuring External Properties

Selecting Properties?

1. Devtools settings
2. @TestPropertySource and @SpringBootTest properties
3. Command line arguments
4. SPRING_APPLICATION_JSON (inline JSON properties).
5. ServletConfig / ServletContext parameters.
6. JNDI attributes from java:comp/env
7. Java System properties
8. OS environment variables
9. Profile-specific application properties
10. Application properties / YAML
11. @PropertySource files
12. SpringApplication.setDefaultProperties.

Location of application.properties
@SpringBootApplication
- /config sub-directory
- Working Directory
- Classpath root

Most common place
- Classpath root: src/main/resources

Profiles
- application-{profile}.properties

Note
- application.properties is always loaded

```
db.driver=org.postgresql.Driver
db.url=jdbc:postgresql://localhost/transfer
db.user=transfer-app
db.password=secret45
```
application-local.properties

```
db.driver=org.postgresql.Driver
db.url=jdbc:postgresql://prod/transfer
db.user=transfer-app
db.password=secret99
```
application-cloud.properties

YAML is supported

Thank You!
Q&A