

ทบทวน Spring Core ใน Spring Framework

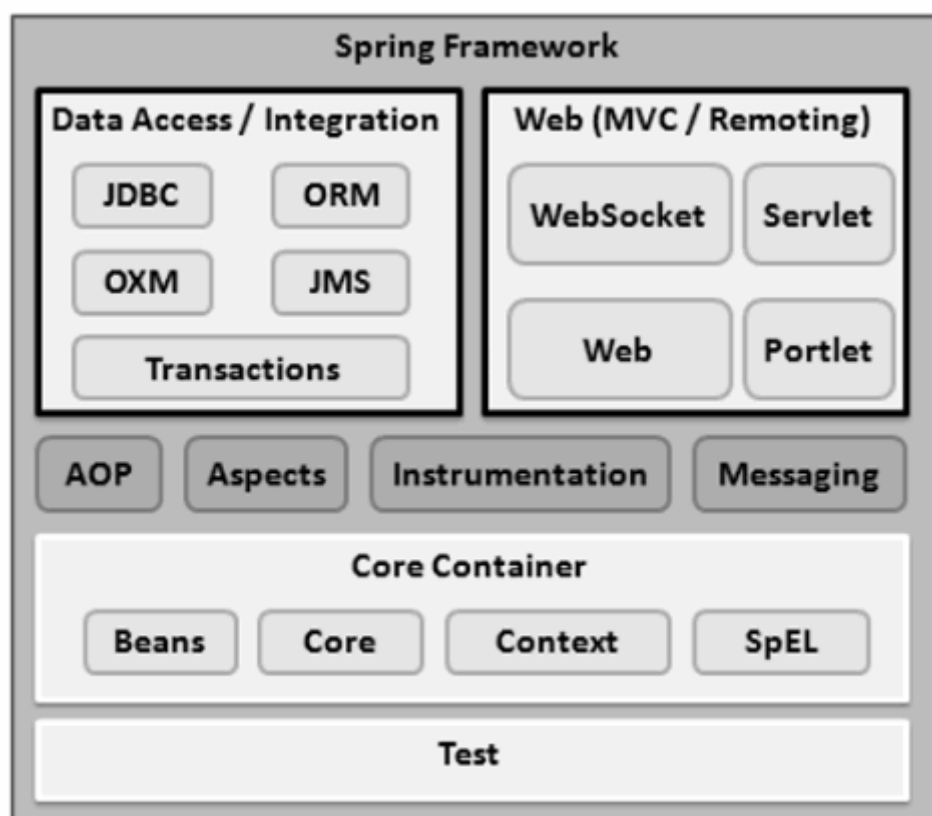


thip

May 2, 2017 · 5 min read

*Spring เป็น framework ที่เหมาะสำหรับนำไปใช้งานกับ Java Enterprise Application เพราะจะ
ช่วยจัดการ object ต่างๆ ให้โดยอัตโนมัติตามที่เรากำหนด Config*

Spring เป็น framework ที่ใหญ่มาก ประกอบไปด้วย Module ต่างๆ มากมาย ซึ่งมีชื่อเรียกแยก
ออกไป เช่น Spring MVC, Spring AOT และในปัจจุบันยังมี Spring Boots ออกมาช่วยให้ dev
เขียนโปรแกรมได้ง่ายขึ้นกว่า Spring ที่ออกมาให้ช่วงแรกๆ อีกด้วย



สำหรับบทความนี้จะพูดถึงตัว Spring Core ที่เป็น Core Basics เพื่อจะได้ทราบถึงพื้นฐาน
concept ของ Spring เพื่อนำไปต่อยอดได้ง่าย

. . .

สิ่งที่ต้องกล่าวถึงเมื่อศึกษา Spring Core

1. **Spring IoC Container** = เป็นตัวจัดการ ควบคุม object ทุกอย่าง ตั้งแต่สร้างจนเลิกใช้งาน
2. **Application Context** = เป็นตัวที่ใช้สร้าง Spring IoC Container
3. **Bean** (Scopes, Life Cycle) = เป็น Object แบบ POJO ที่ถูกสร้างและอยู่ในความควบคุมของ Container
4. **Dependency Injection (DI)** = เป็นการกำหนดความสัมพันธ์ระหว่าง Bean

. . .

การสร้าง Bean

Bean จะถูกสร้างผ่านการ config เรียกว่า Configuration Metadata ทำได้ 3 วิธี

1. **XML Based Configuration** ← config ผ่าน XML files
2. **Annotation Based Configuration** ← config โดยใช้ Annotation
3. **Java Based Configuration** ← config ด้วย Java

โดยการสร้าง Bean นั้นจำเป็นต้องบอกให้ container รู้ 3 สิ่งต่อไปนี้

1. ต้องการสร้าง Bean ประเภทใด
2. ต้องการให้ Bean มี LifeCycle อย่างไร
3. ต้องการให้ Bean ใดสัมพันธ์กับ Bean ใดบ้าง

. . .

Bean Scopes

เป็นการกำหนดขอบเขตการสร้าง instance ของ Bean ใน Container

- **singleton** → bean จะมีแค่ 1 instance ใน container เมื่อมีการเรียกใช้งานก็จะใช้งาน instance เดิม ไม่มีการสร้าง instance ใหม่ขึ้นมา

- **prototype** → bean จะถูกสร้าง instance ขึ้นใหม่ได้หลายๆ instance ทุกครั้งที่มีการเรียกใช้งาน
- **request** → bean จะถูกสร้างขึ้นสำหรับการใช้งาน HTTP request
- **session** → bean จะถูกสร้างขึ้นสำหรับการใช้งาน HTTP session
- **global-session** → bean จะถูกสร้างขึ้นสำหรับการใช้งาน HTTP session ที่มีการเข้าถึงได้แบบ global

. . .

Bean Life Cycle

เมื่อ bean ถูกสร้างขึ้นมา อาจจำเป็นต้องมีการทำงานบางอย่างเพื่อเตรียมความพร้อมให้ bean นั้นพร้อมใช้งาน ในที่นี้เราเรียกว่า “Initialization callbacks” หรือเมื่อ bean นั้นใช้งานเสร็จแล้ว อาจจำเป็นต้องมีการทำงานเพื่อเคลียร์ค่าบางอย่างเช่นกัน ในที่นี้เราเรียกว่า “Destruction callbacks” ซึ่งการกำหนดค่าจะเป็นการระบุชื่อ method การทำงานของ Bean ที่ต้องการให้ทำงานก่อนหรือหลังลงไปขั้นตอนการ config สร้าง Bean

ต่อไปเราจะมาดูวิธีการกำหนดค่า Bean Life Cycle ทั้ง 2 ประเภทกัน

1. **Initialization callbacks** = เป็นการระบุชื่อ method ที่ต้องการให้ container ทำงาน method นั้นเมื่อมีการสร้าง bean instance สำเร็จแล้ว โดยสามารถระบุได้ 2 วิธี
 - ผ่านการ implements interface *org.springframework.beans.factory.InitializingBean* แล้ว override method *afterPropertiesSet()* เพื่อเขียน code การทำงานในส่วนนี้
 - ผ่านการ config ใน XML โดยใช้ attribute **init-method** ใน tag `<bean>` แล้วใส่ค่าเป็นชื่อ method ใน bean นั้นที่ต้องการให้ทำงาน
2. **Destruction callbacks** = เป็นการระบุชื่อ method ที่ต้องการให้ container ทำงาน method นั้นเมื่อ bean instance นั้นถูกลบออกจาก container แล้ว โดยสามารถระบุได้ 2 วิธี
 - ผ่านการ implements interface *org.springframework.beans.factory.DisposableBean* แล้ว override method *destroy()* เพื่อเขียน code การทำงานในส่วนนี้
 - ผ่านการ config ใน XML โดยใช้ attribute **destroy-method** ใน tag `<bean>` แล้วใส่ค่าเป็นชื่อ method ใน bean นั้นที่ต้องการให้ทำงาน

หมายเหตุ! กรณีที่เราใช้การ config แบบ XML และมี bean หลายตัวที่ต้องระบุ init/destroy method เหมือนๆกัน ให้เราตั้งชื่อ method ของทุก bean เป็นชื่อเดียวกัน จากนั้นเราจะสามารถกำหนดชื่อ method เป็น default ไว้ด้านบนสุดใน tag <beans> ได้ โดยไม่จำเป็นต้องตามใส่ init/destroy ทุกๆ tag <bean> ให้เมื่อยมือ

```
<beans xmlns = "http://www.springframework.org/schema/beans"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"
  default-init-method = "init"
  default-destroy-method = "destroy">

  <bean id = "..." class = "...">
    <!-- collaborators and configuration for this bean go here -->
  </bean>

</beans>
```

นอกจากการทำงาน init/destroy แล้ว เราสามารถเพิ่มการทำงานก่อนการ init bean และหลังการ init bean แทรกเข้าไปได้ด้วย โดยการเขียน bean ที่ implements interface ที่ชื่อ **BeanPostProcessor** แล้ว override การทำงานใน method `postProcessBeforeInitialization` และ `postProcessAfterInitialization`

. . .

Bean Inheritance

เมื่อ java มีการสืบทอดคุณสมบัติ จึงทำให้เราสามารถ config Bean ให้มีความเป็นแม่เป็นลูกเป็นญาติกันได้ ผ่าน attribute ที่ชื่อว่า **parent** โดยใส่ที่ child Bean เพื่อบ่งบอกว่าใครเป็นแม่ของนาง ดังตัวอย่างด้านล่างนี้

```
<?xml version = "1.0" encoding = "UTF-8"?>

<beans xmlns = "http://www.springframework.org/schema/beans"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-
    3.0.xsd">

  <bean id = "helloWorld" class = "com.tutorialspoint.HelloWorld">
    <property name = "message1" value = "Hello World!"/>
    <property name = "message2" value = "Hello Second World!"/>
  </bean>

  <bean id = "helloIndia" class = "com.tutorialspoint.HelloIndia"
    parent = "helloWorld">
    <property name = "message1" value = "Hello India!"/>
    <property name = "message3" value = "Namaste India!"/>
  </bean>

</beans>
```

```
</bean>  
</beans>
```

. . .

Dependency Injection (DI)

เมื่อเราพัฒนาระบบที่มีขนาดใหญ่ มีจำนวนคลาสมากๆ คลาสแต่ละคลาสไม่ควรผูกติดกันมากเกินไป เพราะจะทำให้ code ไม่สามารถ reuse ได้ และยากต่อการทำ unit test เพราะเหตุนี้ DI จึงเป็น concept การทำงานที่เข้ามาช่วยแก้ปัญหาเหล่านี้ได้ โดยพูดง่ายๆคือ DI เป็นวิธีการฉีดความสัมพันธ์เพื่อสร้างความเชื่อมโยงแบบหลวมๆระหว่าง object

การทำ DI ใน Spring สามารถทำได้ 2 รูปแบบ

1.ผ่าน Constructor Method = เป็นการระบุ bean เข้าไปเป็น argument ตัวหนึ่งใน constructor แล้ว config XML ผ่านการใช้ attribute ref

```
public class TextEditor {  
    private SpellChecker spellChecker;  
  
    public TextEditor(SpellChecker spellChecker) {  
        System.out.println("Inside TextEditor constructor." );  
        this.spellChecker = spellChecker;  
    }  
  
    public void spellCheck() {  
        spellChecker.checkSpelling();  
    }  
}
```

```
<!-- Definition for textEditor bean -->  
<bean id = "textEditor" class = "com.tutorialspoint.TextEditor">  
    <constructor-arg ref = "spellChecker"/>  
</bean>  
  
<!-- Definition for spellChecker bean -->  
<bean id = "spellChecker" class = "com.tutorialspoint.SpellChecker"></bean>
```

2.ผ่าน Setter Method = เป็นการระบุ bean เข้าไปเป็น argument ตัวหนึ่งใน setter method แล้ว config XML ผ่านการใช้ attribute ref

```
public class TextEditor {  
    private SpellChecker spellChecker;  
  
    // a setter method to inject the dependency.  
    public void setSpellChecker(SpellChecker spellChecker) {  
        System.out.println("Inside setSpellChecker." );  
        this.spellChecker = spellChecker;  
    }  
  
    // a getter method to return spellChecker  
    public SpellChecker getSpellChecker() {  
        return spellChecker;  
    }  
}
```

```

        return spellChecker;
    }
    public void spellCheck() {
        spellChecker.checkSpelling();
    }
}

```

```

<!-- Definition for textEditor bean -->
<bean id = "textEditor" class = "com.tutorialspoint.TextEditor">
    <constructor-arg ref = "spellChecker"/>
</bean>

<!-- Definition for spellChecker bean -->
<bean id = "spellChecker" class = "com.tutorialspoint.SpellChecker"></bean>

```

กรณีที่พัฒนาระบบใหญ่ๆ ความสัมพันธ์ของ Bean ก็ย่อมยุบยิบซับซ้อนไปตามขนาดของระบบ ดังนั้น Spring จึงออกแบบมาให้มีโหมดการทำงานที่สามารถ Scan หาความเป็นญาติกันของ Bean ได้ เรียกว่า “Autowire” เพื่อทดแทนการนั่ง config reference bean ให้ง่ายสะดวกขึ้น

วิธีทำ → ระบุ Scan Mode ที่ต้องการให้ IoC ทำงาน ผ่าน tag ที่ชื่อว่า **autowire**

```

<!-- Definition for textEditor bean -->
<bean id = "textEditor" class = "com.tutorialspoint.TextEditor" autowire = "byName">
    <property name = "name" value = "Generic Text Editor" />
</bean>

<!-- Definition for spellChecker bean -->
<bean id = "spellChecker" class = "com.tutorialspoint.SpellChecker"></bean>

```

Mode ที่สามารถระบุได้ คือ

- *byName* = scan หาจาก property name ที่เหมือนกัน
- *byType* = scan หาจาก property datatype ที่เหมือนกัน
- *constructor* = scan หาจาก property datatype ที่ระบุใน argument ของ constructor
- *autodetect* = scan หาจาก constructor ถ้าไม่เจอจะหาจาก property datatype ที่เหมือนกัน

ข้อควรระวังในการใช้ Autowire !!

- กรณีที่มีการใช้ Autowire แต่เกิดมี dev บางคน config ผ่าน <constructor-arg> และ <property> จะทำให้ Autowire ที่ตั้งค่าไว้ไม่ทำงาน
- เนื่องจาก Autowire ไม่ได้มีการระบุความสัมพันธ์ของ Bean ชัดเจนใน XML อาจทำให้เกิดความสับสนระหว่าง dev ได้

. . .

Annotation Based Configuration

ความยุ่งยากของการ config XML จะหมดไป เมื่อเราใช้ Annotation วิธีการคือ เราสามารถระบุ Annotation ลงไปใน Bean ได้เลย ทั้งที่ class, method และ property แต่ก่อนอื่นเราต้องบอกให้ container รู้ก่อน ว่าเราต้องการ config แบบ Annotation Based โดยการระบุใน XML ตามตัวอย่างด้านล่างนี้

```
<?xml version = "1.0" encoding = "UTF-8"?>

<beans xmlns = "http://www.springframework.org/schema/beans"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context = "http://www.springframework.org/schema/context"
  xsi:schemaLocation = "http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">

  <context:annotation-config/>
  <!-- bean definitions go here -->

</beans>
```

Annotation มีให้ใช้อยู่หลายตัว เช่น @Required, @Autowired, @Qualifier เป็นต้น แต่ละตัวก็จะมีความหมายมาทดแทนการ config แบบ XML นั้นเอง ไม่ขอลงรายละเอียดนะคะ

. . .

Java Based Configuration

อีกทางเลือกหนึ่งในการ config คือการใช้ Annotation ของ Java ระบุไปที่ class, method, attribute ได้เลย ลองเปรียบเทียบการ config ทั้ง 2 แบบตามด้านล่าง

```
@Configuration
public class HelloWorldConfig {
    @Bean
    public HelloWorld helloWorld(){
        return new HelloWorld();
    }
}
```

Java Based Configuration

```
<beans>
  <bean id = "helloWorld" class = "com.tutorialspoint.HelloWorld" />
</beans>
```

XML Based Configuration

Java Annotation ได้แก่ @Configuration, @Bean, @Import, @Scope เป็นต้น ซึ่งแต่ละตัว
นี้จะบ่งบอกการทำงานได้อยู่แล้ว ดูยากกว่าแบบอื่นๆไหมคะ

. . .

ตัวอย่างการเขียน code โดยใช้ Spring

ก่อนใช้งาน Spring เราต้อง download Spring Library มาใส่ Project ก่อนนะจะ version
ล่าสุดตอนนี้คือ 4.3.8 ที่ออกมาเมื่อวันที่ 17/04/2017 นี้เอง

Spring Hello World

1. Java Class

```
public class HelloWorld {  
    private String message;  
  
    public void setMessage(String message){  
        this.message = message;  
    }  
    public void getMessage(){  
        System.out.println("Your Message : " + message);  
    }  
}
```

2. XML Configuration

```
<?xml version = "1.0" encoding = "UTF-8"?>  
  
<beans xmlns = "http://www.springframework.org/schema/beans"  
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"  
    xsi:schemaLocation = "http://www.springframework.org/schema/beans  
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">  
  
    <bean id = "helloWorld" class = "com.tutorialspoint.HelloWorld">  
        <property name = "message" value = "Hello World!"/>  
    </bean>  
  
</beans>
```


3. Main Class Application Context

```
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class MainApp {  
    public static void main(String[] args) {  
        ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");  
        HelloWorld obj = (HelloWorld) context.getBean("helloWorld");  
        obj.getMessage();  
    }  
}
```

. . .

สรุปตามความเข้าใจ

Spring Core เป็นframework ที่ทำให้เราเขียนJava Bean แบบPOJO ธรรมดาธรรมดา แต่สามารถเพิ่มคุณสมบัตินี้ให้Bean ได้โดยผ่านการconfiguration metadata เช่น กำหนดการทำงานของBean, กำหนดการสืบทอดคุณสมบัติและความสัมพันธ์ของBean เป็นต้น มันทำให้POJO บ้างๆ ตัวหนึ่งที่เราเขียนขึ้นมา สามารถใช้งานได้หลากหลายตามการย้าconfig ของเราเอง เรียกได้ว่าwrite less, do more and reuseable ส่วนจะเลือกconfig แบบไหนก็แล้วแต่ทีม แล้วแต่ขนาดของApplication และอย่างที่เกริ่นไปแล้วตอนต้นว่า โลกของSpring นั้นข่างยิ่งใหญ่นัก จบจากSpring Core แล้ว เรายังคงต้องศึกษาSpring's World กันต่อไปค่ะ เพื่อให้เราทำงานได้สะดวกง่ายขึ้น ^_^//

ปล. หากดีและมีประโยชน์ กดปุ่ม  ปรบมือ เพื่อเป็นกำลังใจให้ด้วยนะคะ ^^

Spring Framework Java Programming Developer

About Help Legal