

## ▶ Bean、訊息、事件

Spring 的核心是一個容器（Container），實作了 IoC 的概念，可以協助管理各個物件的生命週期，以及物件之間的依賴關係，在核心容器的使用上，熟悉 BeanFactory 與 ApplicationContext 的運用是了解 Spring 的重點所在，在這一個章節中，將可以了解到如何使用 Spring 容器進行各種 Bean 的組態與管理。

在另一方面，作為一個應用程式框架（Application framework），ApplicationContext 除了具備如 BeanFactory 基本的容器管理功能之外，並支援更多應用程式框架的特性，像是資源的取得、文字訊息解析、事件的處理與傳播等功能。

## 3.1 Bean 基本管理

在第 2 章中，您已經實際完成第一個 Spring 程式，並大致了解何謂「依賴注入」（Dependency Injection）、如何使用 Spring 的容器功能來管理 Bean，在這一個小節當中，將會學習到更多有關於 Bean 在 Spring 中的設定方式，以及生命週期。

### 3.1.1 BeanFactory、ApplicationContext

BeanFactory 負責讀取 Bean 定義檔，管理物件的載入、生成，維護 Bean 物件與 Bean 物件之間的依賴關係，負責 Bean 的生命週期，對於簡單的應用程式來說，使用 BeanFactory 就已經足夠來管理 Bean，在物件的管理上就可以獲得許多的方便性。

不過作為一個應用程式框架，只提供 Bean 容器管理的功能是不夠的，若要利用 Spring 所提供的一些特色以及進階的容器功能，則可以使用 `org.springframework.context.ApplicationContext`，ApplicationContext 的基本功能與 BeanFactory 很相似，它也具有負責讀取 Bean 定義檔，維護 Bean 之間的依賴關係等功能，除此 ApplicationContext 還提供一個應用程式所需的更完整的框架功能，例如：

- 提供取得資源檔案（Resource file）更方便的方法。
- 提供文字訊息解析的方法。
- 支援國際化（Internationalization, I18N）訊息。
- ApplicationContext 可以發佈事件，對事件感興趣的 Bean 可以接收到這些事件。

Spring 的創始者 Rod Johnson 建議使用 ApplicationContext 來取代 BeanFactory，在實作 ApplicationContext 的類別中，最常使用的大概是以下三個：

- `org.springframework.context.support.FileSystemXmlApplicationContext` 可指定 XML 定義檔的相對路徑或絕對路徑來讀取定義檔。
- `org.springframework.context.support.ClassPathXmlApplicationContext` 從 Classpath 設定路徑中來讀取 XML 定義檔。
- `org.springframework.web.context.support.XmlWebApplicationContext` 在 Web 應用程式中的檔案架構中，指定相對位置來讀取定義檔。

舉個簡單的例子來說，可以將第 2 章中所完成的第一個 Spring 程式中的 `SpringDemo` 類別修改為以下的內容：

---

```
package onlyfun.caterpillar;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.
    FileSystemXmlApplicationContext;

public class SpringDemo {
    public static void main(String[] args) {
        ApplicationContext context =
            new FileSystemXmlApplicationContext(
                "beans-config.xml");

        HelloBean hello =
            (HelloBean) context.getBean("helloBean");
        System.out.println(hello.getHelloWord());
    }
}
```

---

如果您先前只將 `spring-beans.jar` 與 `spring-core.jar` 加入至 Classpath 中，則為了能編譯與執行以上的程式，還必須將 `spring-context.jar` 加入至 Classpath 的設定中。

在這邊先介紹在撰寫程式時，如何使用 `ApplicationContext` 替換 `Bean-Factory`，`ApplicationContext` 於文字訊息的管理、事件的發佈等功能，也將在往後相關的主題中一一介紹。

### 3.1.2 Type 2 IoC、Type 3 IoC

在第 2 章中所完成的第一個 Spring 程式中，您利用 Bean 的 Setter 方法完成依賴注入，Spring 鼓勵的是 **Setter Injection**，也就是 **Type 2 Dependency Injection**，但也允許使用 **Type 3 Dependency Injection** 的 **Constructor injection**，要使用 Setter 或 Constructor 來注入依賴關係是視您的需求而定，這在稍後再加以討論，這邊先來看看如何在 Spring 中使用 **Constructor injection**，首先看看 `HelloBean` 類別如何撰寫：

↓ Type3Demo

`HelloBean.java`

```
package onlyfun.caterpillar;

public class HelloBean {
    private String name;
    private String helloWord;

    // 建議要有無參數建構方法
    public HelloBean() {
    }

    public HelloBean(String name, String helloWord) {
        this.name = name;
        this.helloWord = helloWord;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

```
public void setHelloWord(String helloWord) {
    this.helloWord = helloWord;
}

public String getHelloWord() {
    return helloWord;
}
}
```

在定義 Bean 類別時，爲了要能讓 Spring 可以有使用無參數建構方法來生成物件的彈性，建議可以定義一個無參數的建構方法，即使目前沒有撰寫任何的實作內容。

在 HelloBean 類別定義中要注意的是，第二個有參數的建構方法上的兩個參數之順序，在 Bean 定義檔中使用 Constructor Injection 時，在設定上必須指定建構方法上參數的順序，如下所示：

↓ Type3Demo

beans-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING/DTD BEAN/EN"
    "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
    <bean id="helloBean"
        class="onlyfun.caterpillar.HelloBean">
        <constructor-arg index="0">
            <value>Justin</value>
        </constructor-arg>
        <constructor-arg index="1">
            <value>Hello</value>
        </constructor-arg>
    </bean>
</beans>
```

在定義檔案中，使用 `<constructor-arg>` 標籤來表示將使用 Constructor Injection，由於使用 Constructor Injection 並不如 Setter Injection 時擁有

setXXX() 這樣易懂的名稱，所以必須指定參數的位置索引，**index 屬性**就是用於指定物件將注入至建構方法中的哪一個位置的參數，參數的順序指定中，**第一個參數的索引值是 0**，第二個是 1，依此類推。

接著是撰寫主程式，使用 `ApplicationContext` 讀取定義檔案內容、生成 `Bean` 實例、完成依賴關係注入，程式的撰寫如下所示：

↓ Type3Demo

SpringDemo.java

```
package onlyfun.caterpillar;

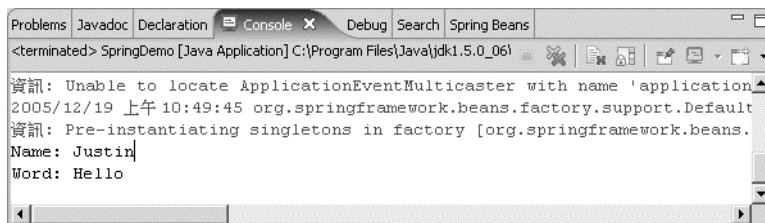
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.
    FileSystemXmlApplicationContext;

public class SpringDemo {
    public static void main(String[] args) {
        ApplicationContext context =
            new FileSystemXmlApplicationContext(
                "beans-config.xml");

        HelloBean hello =
            (HelloBean) context.getBean("helloBean");

        System.out.print("Name: ");
        System.out.println(hello.getName());
        System.out.print("Word: ");
        System.out.println(hello.getHelloWord());
    }
}
```

執行結果如下所示：



```
<terminated> SpringDemo [Java Application] C:\Program Files\Java\jdk1.5.0_061
資訊: Unable to locate ApplicationEventMulticaster with name 'application'
2005/12/19 上午 10:49:45 org.springframework.beans.factory.support.Default
資訊: Pre-instantiating singletons in factory [org.springframework.beans.
Name: Justin]
Word: Hello
```

圖 3.1 Type3Demo 專案執行結果

至於要使用 **Constructor** 或 **Setter** 方法來完成依賴注入這個問題，其實就等於在討論一個古老的問題：要在物件建立時就準備好所有的資源，或是在物件建立好後，再使用 **Setter** 方法來進行設定。

使用 **Constructor** 的好處之一是，可以在建構物件的同時一併完成依賴關係的建立，物件一建立後，它與其它物件的依賴關係也就準備好了，但如果要建立的物件關係很多，使用 **Constructor injection** 會在建構方法上留下一長串的參數，且不易記憶，這時使用 **Setter** 方法會是個不錯的選擇，另一方面，**Setter** 方法具有明確的方法名稱可以瞭解注入的物件會是什麼，像是 `setXXX()` 這樣的名稱，會比記憶 **Constructor** 上某個參數位置的索引代表某個物件來得好，當結合 IDE 的方法提示功能使用時，撰寫程式會更方便且有效率。

然而使用 **Setter** 方法時，由於提供有 `setXXX()` 方法，所以不能保證相關的資料成員或資源在執行時期不會被更改設定，因為程式開發人員可能直接執行 **Setter** 方法來設定相關屬性，所以如果想要讓一些資料成員或資源變為唯讀或是私有，使用 **Constructor inj** 會是個簡單的選擇。

### 3.1.3 屬性參考

之前的例子在定義 Bean 時，可以在定義檔中直接指定一個字串值給屬性值，如果在 Bean 定義檔中已經有一個定義的 Bean 實例，則可以直接讓某個屬性參考至這個實例，以實際的例子來作示範，假設 HelloBean 類別如下定義：

↓ RefBeanDemo

HelloBean.java

```
package onlyfun.caterpillar;

import java.util.Date;

public class HelloBean {
    private String helloWord;
    private Date date;

    public void setHelloWord(String helloWord) {
        this.helloWord = helloWord;
    }

    public String getHelloWord() {
        return helloWord;
    }

    public void setDate(Date date) {
        this.date = date;
    }

    public Date getDate() {
        return date;
    }
}
```

其中 HelloBean 的 setDate() 方法接受一個 Date 的實例，在以下的 Bean 定義檔中，先定義了一個 dateBean，之後定義的 helloBean 則可以直接參考至 dateBean 實例，只要在定義檔中完成定義，接下來 Spring 就會幫您完成這個依賴關係：

RefBeanDemo

beans-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING/DTD BEAN/EN"
    "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>
  <bean id="dateBean" class="java.util.Date"/>

  <bean id="helloBean" class="onlyfun.caterpillar.HelloBean">
    <property name="helloWord">
      <value>Hello!</value>
    </property>
    <property name="date">
      <ref bean="dateBean"/>
    </property>
  </bean>
</beans>
```

要參考至定義檔中已定義的 Bean 實例，是使用 `<ref>` 標籤，並使用 `"name"` 屬性來指定定義檔中要參考的 Bean 別名，可以撰寫以下的程式來測試 Bean 的依賴關係是否完成：

RefBeanDemo

SpringDemo.java

```
package onlyfun.caterpillar;

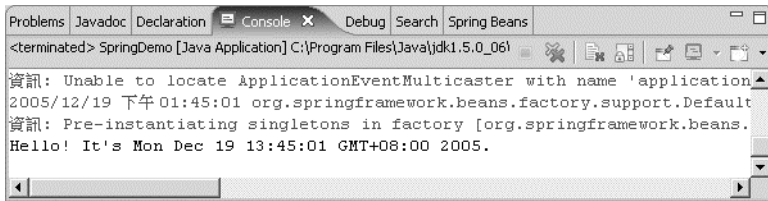
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.
    FileSystemXmlApplicationContext;

public class SpringDemo {
  public static void main(String[] args) {
    ApplicationContext context =
      new FileSystemXmlApplicationContext(
        "beans-config.xml");

    HelloBean hello =
      (HelloBean) context.getBean("helloBean");
    System.out.print(hello.getHelloWord());
  }
}
```

```
        System.out.print(" It's ");
        System.out.print(hello.getDate());
        System.out.println(".");
    }
}
```

執行結果參考畫面如下所示：



► 圖 3.2 RefBeanDemo 專案執行結果

如果某個 Bean 的實例只被某個屬性參考一次，之後在定義檔中不再被其它 Bean 的屬性所參考，那麼也可以直接在屬性定義時使用 `<bean>` 標籤，並僅需指定其 "class" 屬性即可，例如：

```
...
<beans>
  <bean id="helloBean" class="onlyfun.caterpillar.HelloBean">
    <property name="helloWord">
      <value>Hello!</value>
    </property>
    <property name="date">
      <bean class="java.util.Date"/>
    </property>
  </bean>
...

```

Spring 的 IoC 容器會自動生成 Date 實例，並透過 `setDate()` 方法將 Date 實例設定給 `helloBean`。