


Creator	Rod Jhonson
	
Type	Java EE framework
current version	5.3.x (27-10-2020)

Spring Definition

Spring is a **java based, open source, light-weight, loosely coupled, aspect oriented and dependency injection-based java EE framework to develop any kind of applications.**

Features of Spring:

Open Source:

- It means not only free s/w but its source code will be available to the programmers along with installation.

Light-Weight:

- Spring applications are light-weight applications. Due to the following reasons.
 1. Spring containers are light-weight containers. Because they can be activated anywhere by just creating objects for certain pre-defined classes.

Note:

Servlet Container, Jsp container are Heavy weight containers.

2. Most of the spring applications can be executed without heavy-weight application server and web server s/ws.
3. Resources of spring applications can be developed without using spring api.
4. While executing spring applications much memory and cpu resources are not required.

Aspect Oriented:

- It is always recommended to develop large scale applications by enabling **middleware service** like **Security, Transaction Management, Logging**, etc.
- Spring provides a new methodology called **Spring AOP** to apply these middleware services on spring applications. This makes the spring as Aspect Oriented.

Loosely Coupled:

- If the degree of dependency is more b/w 2 components then the components are **tightly-coupled** components.
- If the degree of dependency is less b/w two components then we can say components are Loosely coupled.

Dependency Injection:

- It is previously also as called **IOC (Inversion of Control)**
- It is defined as the **underlying server** or **container** or **framework** or **some resource** assign or inject the values to the applications automatically and dynamically.

Eg: 1

If Id card and course material is issued to the student as soon as the student registers for a course without waiting.

Eg: 2

The way JVM calls constructor automatically to assign initial values to object.

Advantage:

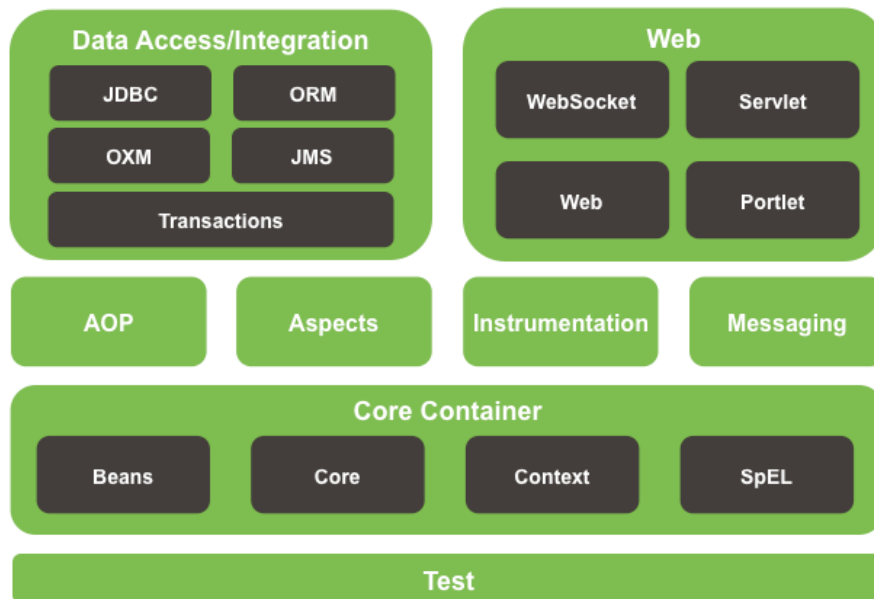
The resource can be injected dependent values directly without spending time to get them.

Spring Versions

1.0	2004
2.0	2006
3.0	2009
4.0	2013
5.0	2017
5.1	2018
5.2	2019
5.3	2020



Spring Framework Runtime



Description of Modules:

1. Spring Core:

- It is the base module for all other modules in Spring Framework.
- It talks about Inversion of Control (IoC) and Dependency Injection

2. Spring DAO/ Spring JDBC:

- It provides **Abstraction Layer on Plain JDBC** to develop persistence Logic.

3. Spring ORM:

- It provides **abstraction layer on other ORM frameworks** (like Hibernate) and allows to develop ORM based persistence logic.

4. Spring AOP:(Aspect Oriented Programming)

- It is given to apply Aspects (Middleware Services) on Spring Applications.
- Middleware Services are additional, optional and configurable services or logics on our applications to make them more perfect and accurate.

Eg:

Security, Transaction Management, Logging, etc.

5. Spring Web:

- It is given to make Spring applications communicate or integrate to other web framework s/w applications like **Struts App, JSF app, etc.**

6. Spring Web-MVC:

- It is given spring's own web framework to develop **MVC architecture-based Web applications.**

7. Spring Context/Spring JEE:

- It provides abstraction layer on multiple core technologies like **JNDI, EJB, RMI, JMS, Java Mail** etc.
- It is used to develop enterprise applications, distributed applications and other applications.

- It is base module for other Modules of spring.
- By using this module, we can develop only **Standalone Applications**.

Spring Container or IOC Container:

- Container is a software application that can take care of the whole **life cycle of the given resource**.
- Spring Container is responsible for following activities-
 1. To read spring configuration xml files and validate them.
 2. To create instances of POJO classes (model classes) based on xml file or annotations configuration.
 3. To manage Life Cycle of POJO class or Spring Bean class. (A java class is called as Spring Bean in Spring Framework)
 4. To perform **Dependency Injection** through either **Setter Injection** or **Constructor Injection**.

Types of Spring Containers:

- There are 2 containers in spring.
 1. **BeanFactory**
 2. **ApplicationContext**.

- It is enhancement of **BeanFactory** container.
- It is a **JAVA EE CONTAINER**.
- It is an **interface** present in **org.springframework.context.***;
- It creates object while loading configuration xml file itself if the scope is 'singleton'.
- It is Eager Container.
- It is also Light Weight container.
- If the scope is 'prototype' then both are same.
- It is most commonly used spring container.
- There are 3 important implementation classes of **ApplicationContext** interface.
 1. **FileSystemXmlApplicationContext**
 2. **ClassPathXmlApplicationContext**
 3. **WebXmlApplicationContext**

FileSystemXmlApplicationContext:

- It activates Spring Container by locating spring configuration file from the specified path of **File System**.
- It is present in **org.springframework.context.support.***;

```
String file = "src/com/seshusoft/cfgs/applicationContext.xml";  
ApplicationContext context = new FileSystemXmlApplicationContext(file);
```

ClassPathXmlApplicationContext:

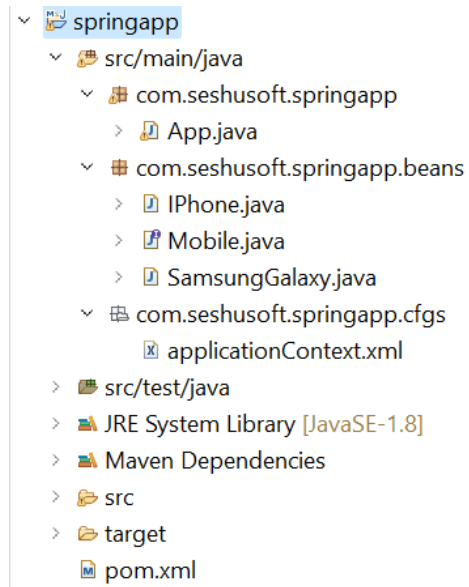
- It activates spring container by locating spring configuration file from the values added to the **classpath**.
- It is most commonly used one.
- It is present **org.springframework.context.support.***;

```
String file = "com/seshusoft/cfgs/applicationContext.xml";  
ApplicationContext context = new ClassPathXmlApplicationContext(file);
```

WebXmlApplicationContext:

- It will load the xml files from a web application.

```
ApplicationContext appContext =  
WebApplicationContextUtils.getWebApplicationContext(servletContext);
```



Add the following dependencies in pom.xml file

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.seshusoft</groupId>
    <artifactId>springapp</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>springapp</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
        <dependency>
```

```
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>5.3.15</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.15</version>
    </dependency>

</dependencies>
</project>
```

Mobile.java

```
package com.seshusoft.springapp.beans;

public interface Mobile {
    void call();
}
```

IPhone.java

```
package com.seshusoft.springapp.beans;

public class IPhone implements Mobile{
    public IPhone(){
        System.out.println("IPhone instance is created");
    }

    @Override
    public void call() {
        System.out.println("calling with iPhone...");
    }
}
```


SamungGalaxy.java

```
package com.seshusoft.springapp.beans;

public class SamungGalaxy implements Mobile{
    public SamungGalaxy() {
        System.out.println("SamungGalaxy instance is created");
    }
    @Override
    public void call() {
        System.out.println("calling with Samung Galaxy...");
    }
}
```

applicationContext.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:context="http://www.springframework.org/schema/context"
        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-4.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd">

    <bean class="com.seshusoft.springapp.beans.IPhone" id="iphone" />

    <bean class="com.seshusoft.springapp.beans.SamungGalaxy" id="galaxy" />

</beans>
```

App.java

```
package com.seshusoft.springapp;

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

import com.seshusoft.springapp.beans.Mobile;

public class App {
    public static void main( String[] args ){
        String file = "com/seshusoft/springapp/cfgs/applicationContext.xml";
        ApplicationContext ctx = new ClassPathXmlApplicationContext(file);
        Mobile mobile = (Mobile)ctx.getBean("iphone");
        mobile.call();

        Mobile mobile2 = (Mobile)ctx.getBean("galaxy");
        mobile2.call();
    }
}
```

```
}  
}
```

Run it App.java as Java Application

```
iPhone instance is created  
SamsungGalaxy instance is created  
calling with iPhone...  
calling with Samsung Galaxy...
```

- It is defined as the **underlying server** or **container** or **f/w** or **some resource** assigns values to the applications automatically and dynamically.
- It is previously also as called **IOC (Inversion of Control)**

Eg: 1

If Id card and course material is issued to the student as soon as the student registers for a course without waiting.

Eg: 2

The way JVM calls constructor automatically to assign initial values to object.

Advantage:

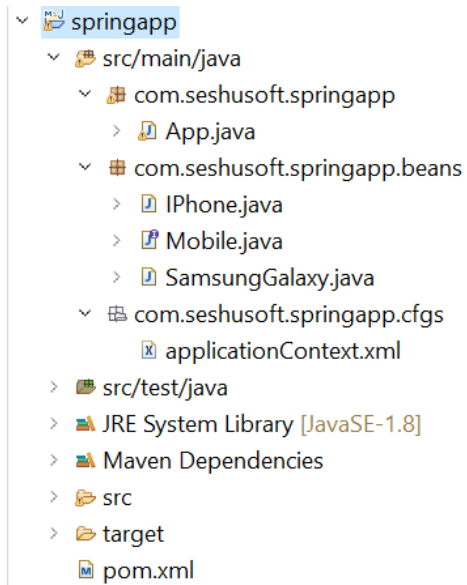
The resource can be injected dependent values directly without spending time to get them.

Types of Dependency Injections in Spring:

Spring supports 2 types of Dependency Injections:

1. **Setter Injection**
2. **Constructor Injection**

Example: Constructor Injection.



Mobile.java

```
package com.seshusoft.springapp.beans;

public interface Mobile {
    void call();
}
```

IPhone.java

```
package com.seshusoft.springapp.beans;

public class IPhone implements Mobile{
    private String modelNo;
    private String color;

    public IPhone(String modelNo, String color){
        this.modelNo = modelNo;
        this.color = color;
        System.out.println("IPhone instance is created");
    }

    @Override
    public void call() {
        System.out.println("calling with iPhone...");
    }

    public void display() {
        System.out.println("Model Number: "+modelNo);
        System.out.println("Color: "+color);
    }
}
```

```
}
```

SamungGalaxy.java

```
package com.seshusoft.springapp.beans;

public class SamsungGalaxy implements Mobile{
    private String modelNo;
    private String color;

    public SamsungGalaxy(String modelNo, String color) {
        this.modelNo = modelNo;
        this.color = color;
        System.out.println("SamsungGalaxy instance is created");
    }
    @Override
    public void call() {
        System.out.println("calling with Samsung Galaxy...");
    }

    public void display() {
        System.out.println("Model Number: "+modelNo);
        System.out.println("Color: "+color);
    }
}
```

applicationContext.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    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-4.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd">

    <bean class="com.seshusoft.springapp.beans.IPhone" id="iphone">
        <constructor-arg value="M1234" />
        <constructor-arg value="Black" />
    </bean>

    <bean class="com.seshusoft.springapp.beans.SamsungGalaxy" id="galaxy">
        <constructor-arg value="M1010" />
        <constructor-arg value="Silver" />
    </bean>
</beans>
```

App.java

```
package com.seshusoft.springapp;
```

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.seshusoft.springapp.beans.IPhone;
import com.seshusoft.springapp.beans.Mobile;
import com.seshusoft.springapp.beans.SamsungGalaxy;

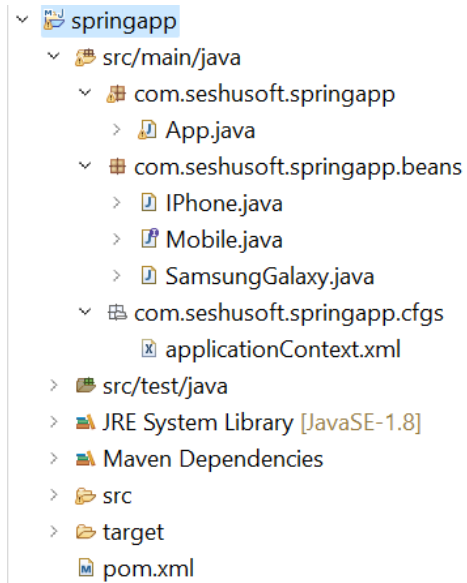
public class App {
    public static void main( String[] args ){
        String file = "com/seshusoft/springapp/cfgs/applicationContext.xml";
        ApplicationContext ctx = new ClassPathXmlApplicationContext(file);
        IPhone mobile = (IPhone)ctx.getBean("iphone");
        mobile.display();

        SamsungGalaxy mobile2 = (SamsungGalaxy)ctx.getBean("galaxy");
        mobile2.display();
    }
}
```

Run it App.java as Java Application

```
IPhone instance is created
SamsungGalaxy instance is created
Model Number: M1234
Color: Black
Model Number: M1010
Color: Silver
```

Example: Setter Injection



Mobile.java

```
package com.seshusoft.springapp.beans;

public interface Mobile {
    void call();
}
```

IPhone.java

```
package com.seshusoft.springapp.beans;

public class IPhone implements Mobile {
    private String modelNo;
    private String color;

    public IPhone() {
        System.out.println("IPhone instance is created");
    }

    public String getModelNo() {
        return modelNo;
    }

    public void setModelNo(String modelNo) {
        this.modelNo = modelNo;
    }

    public String getColor() {
        return color;
    }
}
```

```

    }

    public void setColor(String color) {
        this.color = color;
    }

    @Override
    public void call() {
        System.out.println("calling with iPhone...");
    }

    public void display() {
        System.out.println("Model Number: " + modelNo);
        System.out.println("Color: " + color);
    }
}

```

SamungGalaxy.java

```

package com.seshusoft.springapp.beans;

public class SamungGalaxy implements Mobile {
    private String modelNo;
    private String color;

    public SamungGalaxy() {
        System.out.println("SamungGalaxy instance is created");
    }

    public String getModelNo() {
        return modelNo;
    }

    public void setModelNo(String modelNo) {
        this.modelNo = modelNo;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    @Override
    public void call() {

```



```

        System.out.println("calling with Samsung Galaxy...");
    }

    public void display() {
        System.out.println("Model Number: " + modelNo);
        System.out.println("Color: " + color);
    }
}

```

applicationContext.xml

```

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    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-4.0.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-4.0.xsd">

    <bean class="com.seshusoft.springapp.beans.IPhone" id="iphone">
        <property name="modelNo" value="M1234" />
        <property name="color" value="Black" />
    </bean>

    <bean class="com.seshusoft.springapp.beans.SamsungGalaxy" id="galaxy">
        <property name="modelNo" value="M1010" />
        <property name="color" value="Silver" />
    </bean>

</beans>

```

App.java

```
package com.seshusoft.springapp;

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

import com.seshusoft.springapp.beans.IPhone;
import com.seshusoft.springapp.beans.SamsungGalaxy;

public class App {
    public static void main(String[] args) {
        String file = "com/seshusoft/springapp/cfgs/applicationContext.xml";
        ApplicationContext ctx = new ClassPathXmlApplicationContext(file);
        IPhone mobile = (IPhone) ctx.getBean("iphone");
        mobile.display();

        SamsungGalaxy mobile2 = (SamsungGalaxy) ctx.getBean("galaxy");
        mobile2.display();
    }
}
```

Run it App.java as Java Application

```
IPhone instance is created
SamsungGalaxy instance is created
Model Number: M1234
Color: Black
Model Number: M1010
Color: Silver
```

When should go for Constructor Injection and Setter Injection?

- When spring bean has only one property, then go for Constructor Injection
- When spring bean has many properties, then go for Setter Injection.

- Wiring is process of performing **Dependency Injection** on Bean properties.

Types of wirings:

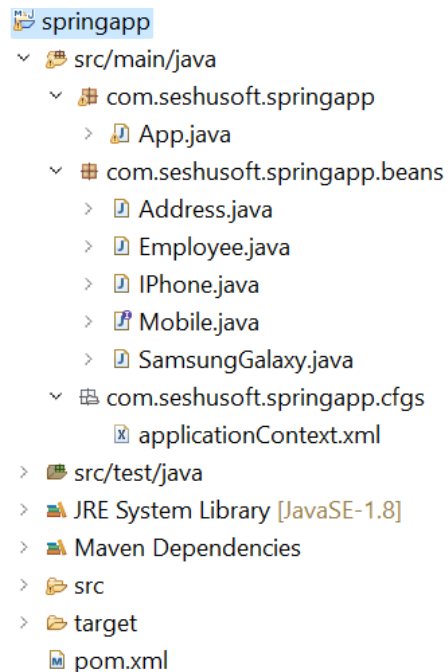
1. Explicit Wiring

- Here, bean properties will be configured explicitly for wiring using following elements.
 - <property> element for setter injection**
 - <constructor-arg> element for constructor injection**

2. Auto Wiring

- To enable auto wiring on bean properties, use `@Autowired` annotation
- It was introduced in **spring 2.5 version**.
- Auto wiring is not possible on **simple types** and **String types** but only on **reference type bean properties**.

Example:



```
package com.seshusoft.springapp.beans;

public class Address {
    private String city, state, country;

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    @Override
    public String toString() {
        return "Address [city=" + city + ", state=" + state + ", country=" +
country + "];"
    }
}
```

```
package com.seshusoft.springapp.beans;

import org.springframework.beans.factory.annotation.Autowired;

public class Employee {
    private int employeeId;
    private String employeeName;

    @Autowired
    private Address employeeAddress;

    public int getEmployeeId() {
        return employeeId;
    }
    public void setEmployeeId(int employeeId) {
        this.employeeId = employeeId;
    }
    public String getEmployeeName() {
        return employeeName;
    }
    public void setEmployeeName(String employeeName) {
        this.employeeName = employeeName;
    }
    public Address getEmployeeAddress() {
        return employeeAddress;
    }
    public void setEmployeeAddress(Address employeeAddress) {
        this.employeeAddress = employeeAddress;
    }
    @Override
    public String toString() {
        return "Employee [employeeId=" + employeeId + ", employeeName=" +
employeeName + ", employeeAddress="
        + employeeAddress + "]";
    }
}
```

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:context="http://www.springframework.org/schema/context"
  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-4.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-4.0.xsd">

  <context:annotation-config />

  <!-- <bean class="com.seshusoft.springapp.beans.IPhone" id="iphone">
    <property name="modelNo" value="M1234" />
    <property name="color" value="Black" />
  </bean>

  <bean class="com.seshusoft.springapp.beans.SamsungGalaxy" id="galaxy">
    <property name="modelNo" value="M1010" />
    <property name="color" value="Silver" />
  </bean> -->

  <bean class="com.seshusoft.springapp.beans.Address" id="address">
    <property name="city" value="Hyderabad" />
    <property name="state" value="Telangana" />
    <property name="country" value="India" />
  </bean>

  <bean class="com.seshusoft.springapp.beans.Employee" id="employee">
    <property name="employeeId" value="101" />
    <property name="employeeName" value="Jones" />
    <!-- <property name="employeeAddress" ref="address" /> -->
  </bean>
</beans>
```

```
package com.seshusoft.springapp;

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

import com.seshusoft.springapp.beans.Employee;

public class App {
    public static void main(String[] args) {
        String file = "com/seshusoft/springapp/cfgs/applicationContext.xml";
        ApplicationContext ctx = new ClassPathXmlApplicationContext(file);
        Employee employee = (Employee)ctx.getBean("employee");
        System.out.println(employee);
    }
}
```

```
Employee [employeeId=101, employeeName=Jones, employeeAddress=Address [city=Hyderabad, state=Telangana, cou ^
```

Java Based Configuration

Spring application can be configured using java-based configuration with following annotations;

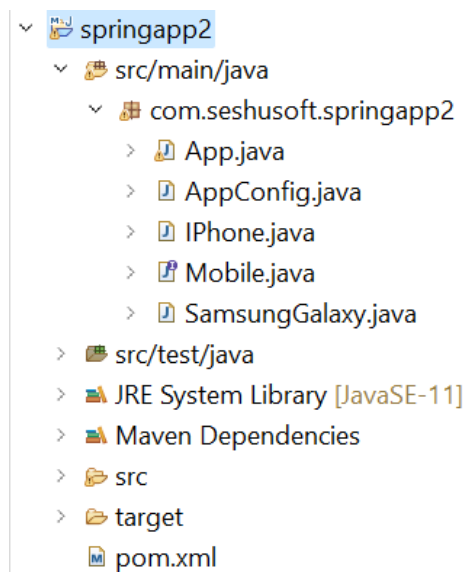
@Configuration

Indicates that a class declares one or more @Bean methods and may be processed by the Spring container to generate bean definitions and service requests for those beans at runtime.

@Bean

Indicates that a method produces a bean to be managed by the Spring container.

Example:




```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.seshusoft</groupId>
    <artifactId>springapp2</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>springapp2</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>

    </properties>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>5.3.15</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.15</version>
        </dependency>

    </dependencies>
</project>
```

```
package com.seshusoft.springapp2;

public interface Mobile {
    void call();
}
```

```
package com.seshusoft.springapp2;

public class iPhone implements Mobile{
    private String modelNo;
    private String color;

    public iPhone(String modelNo, String color){
        this.modelNo = modelNo;
        this.color = color;
        System.out.println("iPhone instance is created");
    }

    @Override
    public void call() {
        System.out.println("calling with iPhone...");
    }

    public void display() {
        System.out.println("Model Number: "+modelNo);
        System.out.println("Color: "+color);
    }
}
```

```
package com.seshusoft.springapp2;

public class SamsungGalaxy implements Mobile{
    private String modelNo;
    private String color;

    public SamsungGalaxy(String modelNo, String color) {
        this.modelNo = modelNo;
        this.color = color;
        System.out.println("SamsungGalaxy instance is created");
    }

    @Override
    public void call() {
        System.out.println("calling with Samsung Galaxy...");
    }
}
```

```
}

    public void display() {
        System.out.println("Model Number: "+modelNo);
        System.out.println("Color: "+color);
    }
}
```

```
package com.seshusoft.springapp2;

public class Address {
    private String city, state, country;

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    @Override
    public String toString() {
        return "Address [city=" + city + ", state=" + state + ",
country=" + country + "]";
    }
}
```

```
package com.seshusoft.springapp2;

import org.springframework.beans.factory.annotation.Autowired;

public class Employee {
    private int employeeId;
```

```

    private String employeeName;

    @Autowired
    private Address employeeAddress;

    public int getEmployeeId() {
        return employeeId;
    }
    public void setEmployeeId(int employeeId) {
        this.employeeId = employeeId;
    }
    public String getEmployeeName() {
        return employeeName;
    }
    public void setEmployeeName(String employeeName) {
        this.employeeName = employeeName;
    }
    public Address getEmployeeAddress() {
        return employeeAddress;
    }
    public void setEmployeeAddress(Address employeeAddress) {
        this.employeeAddress = employeeAddress;
    }
    @Override
    public String toString() {
        return "Employee [employeeId=" + employeeId + ", employeeName="
+ employeeName + ", employeeAddress="
        + employeeAddress + "];"
    }
}

```

```
package com.seshusoft.springapp2;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfig {
    @Bean
    public iPhone getIphone() {
        return new iPhone("iPhone 14", "Black");
    }

    @Bean
    public SamsungGalaxy getSamsung() {
        return new SamsungGalaxy("Galaxy 14", "Black");
    }

    @Bean
    public Address getAddress() {
        Address address = new Address();
        address.setCity("Hyderabad");
        address.setState("Telangana");
        address.setCountry("India");
        return address;
    }

    @Bean
    public Employee getEmployee() {
        Employee employee = new Employee();
        employee.setEmployeeId(1001);
        employee.setEmployeeName("Raj");
        employee.setEmployeeAddress(getAddress());
        return employee;
    }
}
```

```
package com.seshusoft.springapp2;

import org.springframework.context.ApplicationContext;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext
;

public class App {
    public static void main(String[] args) {
        ApplicationContext ctx = new
AnnotationConfigApplicationContext(AppConfig.class);
        iPhone iphone = ctx.getBean(IPhone.class);
        iphone.display();

        SamsungGalaxy samsung = ctx.getBean(SamsungGalaxy.class);
        samsung.display();

        Employee employee = ctx.getBean(Employee.class);
        System.out.println("Employee Id: "+employee.getEmployeeId());
        System.out.println("Employee Name:
"+employee.getEmployeeName());
        System.out.println("Employee Address:
"+employee.getEmployeeAddress());
    }
}
```