

Spring framework :-

Q) Why do we need it ?

Java Technology

① software development

② web development

③ enterprise Development

→ It provide built in module for

→ Common Issue facing dependency
It provide Dependency Injection

→ provide IoC

web framework

① MVC (Model - View - Controller)

is very important for web app.

Spring have built-in module as Spring MVC

Database

Spring provide separate built in module for
Database connectivity.

Only focus on This tutorial is

Dependency Injection

of Spring Core

so

- ① Dependency Injection
- ② maven
- ③ Spring IoC

]

✓

Q) what is Dependency Injection ?

Design Pattern

① Dependency Injection is a type of Design pattern.

what is Dependency Injection & why to use it .

most people by default follows this approach.

In Industries

Dependency means

some object depends on another object.

Object Graph

example of

Laptop Building

Class Laptop {

HardDrive obj1;

Ram obj2;

}

class Laptop {

 HitachiHD obj1 = new HitachiHD();

} Tight coupling

→ loose coupling if

one object not completely depends on
another object It could be replace.

Interface → HardDrive



Class → HitachiHD

class Laptop {

 HardDrive obj1 = new HitachiHD();

} also Tight coupling

fear

We wants some one else gives that dependency.

We wants one else inject those dependency

Dependency Injection Container. ↗
Spring Container.

Spring Dependency Injection Container

They are responsible to create object for you and inject in the required class.

There are multiple way for configuration

① one way is .xml file (earlier Spring approach)

We can change .xml file later

if some one ask for Employee object
give him Student object.

Its mention in .xml file in Spring.

Configuration

So there is no tight coupling

② Second way to Spring Boot approach
latest approach annotation.

Because as a java programmer we
don't want to focus on .xml

class Laptop

{

@Autowired
HardDrive obj;

}

@Component
class HitachiHD implements HardDrive

{

==

}

@Component of

which makes that class object
component of Spring Boot framework which
will be generated on requirement.

@Autowired of

Spring Boot say some one asking for Harddrive
object I do have object I can connect
them.

Why

- ① we want loose coupling
- ② Testing easy

Ex: HardDrive already Tested by
other company

So while Testing Laptop we should not test HardDrive

This can be achieve only when there is loose coupling through Dependency injection.

using Mock object.

③ Maintenance of Code

Maven Introduction

Maven is a build tool under licence Apache.

Maven Repositories (mvnrepository) there are a lots of libraries present.

Why we need maven

for creating a new project we need a lots of third-party libraries.

Q1 Create a project to work with MySQL
we need MySQL connector library jar
dependencies

① to work with Spring MVC

we need 10-12 libraries for dependency

② to work with Hibernate

we need 10-12 libraries for dependency.

So maven takes responsibility to download and provide all dependency and even version update just need to mentioned in pom.xml.
and it will make sure Spring MVC dependency also match with hibernate dependency. for that there is repository

Advantage of maven

- ① Different goal to achieve
- ② compile & build
- ③ deploy

④ provide dependency jar libraries.

dependency = jar file

Pom.xml

```
*aa/pom.xml ✘
1<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
3  <modelVersion>4.0.0</modelVersion>
4  <groupId>com.telusko</groupId>
5  <artifactId>aa</artifactId>
6  <packaging>war</packaging>
7  <version>0.0.1-SNAPSHOT</version>
8  <name>aa Maven Webapp</name>
9  <url>http://maven.apache.org</url>
10 
11 <dependencies>
12 
13 Hey I need Hibernate.....
14 
15 </dependencies>
16 
17 <build>
18   <finalName>aa</finalName>
19 </build>
20 </project>
21
```

1. Project

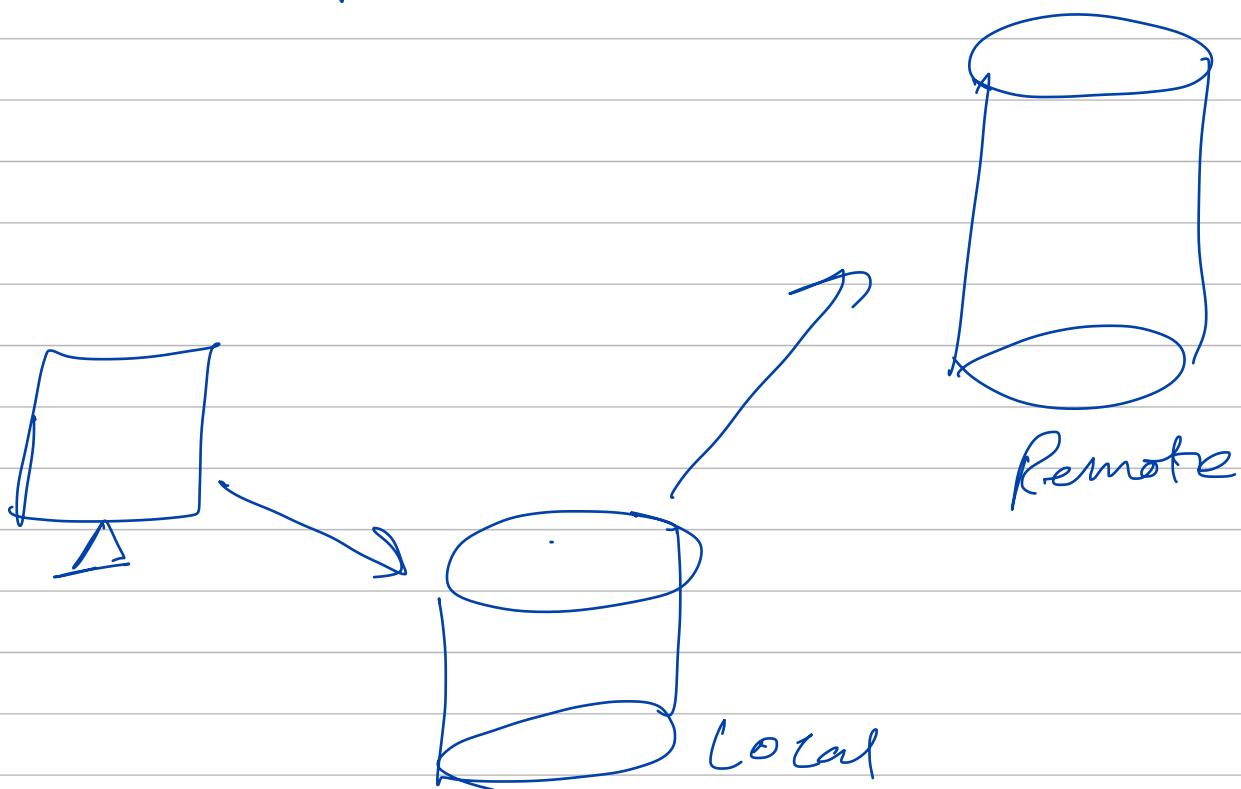
2. GroupId - com.telusko
3. ArtifactId - demo
4. Package - com.telusko.demo.web

```
1<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
3  <modelVersion>4.0.0</modelVersion>
4  <groupId>com.telusko</groupId>
5  <artifactId>aa</artifactId>
6  <packaging>war</packaging>
7  <version>0.0.1-SNAPSHOT</version>
8  <name>aa Maven Webapp</name>
9  <url>http://maven.apache.org</url>
10 
11 <dependencies>
12 
13   <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
14   <dependency>
15     <groupId>org.springframework</groupId>
16     <artifactId>spring-webmvc</artifactId>
17     <version>4.3.8.RELEASE</version>
18   </dependency>
19 
20 
21 </dependencies>
22 
23 <build>
24   <finalName>aa</finalName>
25 </build>
```

Note- Instead of going individual website for dependency jar download

there is a central website
own repository

from where we can get all related dependency and libraries at a place



So Maven to build & get jar files.

All the project we build in spring need maven.

other Gradle → Android

for java & web → mostly use maven.
we can also use gradle.

Maven Practical

Maven → a build tool

for to create a project

- ① Create Project Structure
- ② Compile your project
- ③ Test your Application
- ④ Provide all required library

Ex. if you use Spring then its provide all dependencies / libraries / jar which are required to develop an application

if you are using Hibernate its provide all dependencies which are required for using Hibernate in your project.

To Create first Maven Project

Eclipse

- ① file → New → others → Maven → maven project → chose work location
→ Select Archetype → finish

for core java application simple Maven project
maven-archetype-quickstart

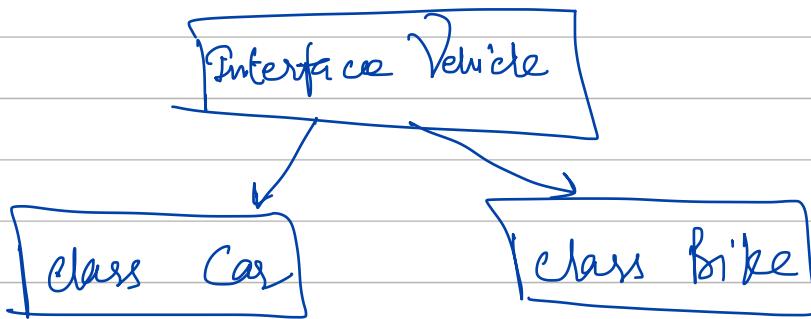
for web application

maven - archetype - web

Spring Core framework Tutorial

ApplicationContext

① Using Interfaces



② Using Spring (.xml file)

App. class :-

main method

for small Application

for Enterprise Application
(Super Set of BeanFactory)

BeanFactory → org.springframework.bean.factory
ApplicationContext → org.springframework.context

Interface

main () {

```
ApplicationContext context = new ClassPathXmlApplicationContext();  
Vehicle obj = (Vehicle) context.getBean("Vehicle");  
obj.drive();
```

Spring Core Framework

To Create Xml file to mention the object which should be pass.

We do not need to recompile java whole program if we change object passing in xml file. we can take any name.

Bean.xml → at Class Path

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays a Maven project named 'maven [Spring-Projects master]' with a 'src/main/java' folder containing 'com.sshacker.maven.demo' package with classes 'App.java', 'Bike.java', 'Car.java', 'Vehicle.java', and 'Bean.xml'. It also shows 'src/test/java', 'JRE System Library [J2SE-1.5]', 'Maven Dependencies', and 'src' folder with 'main' and 'test' subfolders, along with 'pom.xml'. On the right, the editor view shows the 'Bean.xml' file content:

```
<?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="vehicle" class="com.sshacker.maven.demo.Bike"></bean>
</beans>
```

The screenshot shows the Eclipse IDE interface with the same project structure. The 'App.java' file in the editor contains the following code:

```
package com.sshacker.maven.demo;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class App
{
    public static void main( String[] args )
    {
        ApplicationContext context = new ClassPathXmlApplicationContext("Bean.xml");
        Vehicle obj = (Vehicle)context.getBean("vehicle");
        obj.drive();
    }
}
```

Annotation Based Configuration

3 types of Configurations

- ① xml Based
- ② Annotation Based
- ③ java Based.

ApplicationContext interface object is responsible to give the bean (required object) which needed.

The screenshot shows the Eclipse IDE interface with a Maven project named "maven [Spring-Projects master]". The Package Explorer view shows the project structure with src/main/java containing com.sshacker.maven.demo, src/test/java, JRE System Library [J2SE-1.5], Maven Dependencies, and src. The src folder contains main, test, target, and pom.xml. The App.java file is open in the editor, showing the following code:

```
1 package com.sshacker.maven.demo;
2
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6 public class App
7 {
8     public static void main( String[] args )
9     {
10        ApplicationContext context = new ClassPathXmlApplicationContext("Bean.xml");
11        Vehicle obj = (Vehicle)context.getBean("bike");
12        obj.drive();
13    }
14 }
```

The screenshot shows the Eclipse IDE interface with the Bean.xml file open in the editor. The Package Explorer view is the same as the previous screenshot. The Bean.xml file contains the following XML configuration:

```
1 <?xml version = "1.0" encoding = "UTF-8"?>
2
3 <beans xmlns = "http://www.springframework.org/schema/beans"
4   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
5   xmlns:context = "http://www.springframework.org/schema/context"
6   xsi:schemaLocation = "http://www.springframework.org/schema/beans
7   http://www.springframework.org/schema/spring-beans-3.0.xsd
8   http://www.springframework.org/schema/context
9   http://www.springframework.org/schema/context/spring-context-3.0.xsd">
10
11   <context:component-scan base-package="com.sshacker.maven.demo"></context:component-scan >
12
13 </beans>
```

The screenshot shows the Eclipse IDE interface with the 'Package Explorer' view on the left and the 'Bike.java' file in the 'Editor' view on the right. The 'Bike.java' file contains the following Java code:

```
1 package com.sshacker.maven.demo;
2
3 import org.springframework.stereotype.Component;
4
5 @Component
6 public class Bike implements Vehicle{
7     public void drive() {
8         System.out.println("Bhaag raha hai...");
```

The screenshot shows the Eclipse IDE interface with the 'Package Explorer' view on the left and the 'Bike.java' file in the 'Editor' view on the right. The 'Bike.java' file contains the following Java code:

```
1 package com.sshacker.maven.demo;
2
3 import org.springframework.stereotype.Component;
4
5 @Component
6 public class Bike implements Vehicle {
7     public void drive() {
8         System.out.println("Bhaag raha hai...");
```

Bean Properties

The screenshot shows the Eclipse IDE interface with the 'Package Explorer' view on the left and the 'App.java' file in the 'Editor' view on the right. The 'App.java' file contains the following Java code:

```
1 package com.sshacker.maven.demo;
2
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6 public class App
7 {
8     public static void main( String[] args )
9     {
10        ApplicationContext context = new ClassPathXmlApplicationContext("Bean.xml");
11        Tyre obj = (Tyre)context.getBean("tyre");
12        System.out.println(obj);
```

Package Explorer

```

maven [Spring-Projects master]
src/main/java
  com.sshacker.maven.demo
    App.java
    Bike.java
    Car.java
    Tyre.java
    Vehicle.java
    Bean.xml
src/test/java
JRE System Library [J2SE-1.5]
Maven Dependencies
src
  main
  test
target
pom.xml

```

App.java

```

1 package com.sshacker.maven.demo;
2
3 public class Tyre {
4     private String brand;
5
6     public String getBrand() {
7         return brand;
8     }
9
10    public void setBrand(String brand) {
11        this.brand = brand;
12    }
13
14    @Override
15    public String toString() {
16        return "Tyre [brand=" + brand + "]";
17    }
18
19 }
20

```

File Edit Navigate Search Project Run Window Help

Package Explorer

```

maven [Spring-Projects master]
src/main/java
  com.sshacker.maven.demo
    App.java
    Bike.java
    Car.java
    Tyre.java
    Vehicle.java
    Bean.xml
src/test/java
JRE System Library [J2SE-1.5]
Maven Dependencies
src
  main
  test
target
pom.xml

```

App.java

```

1 <?xml version = "1.0" encoding = "UTF-8"?>
2
3 <beans xmlns = "http://www.springframework.org/schema/beans"
4   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
5   xmlns:context = "http://www.springframework.org/schema/context"
6   xsi:schemaLocation = "http://www.springframework.org/schema/beans
7   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
8   http://www.springframework.org/schema/context
9   http://www.springframework.org/schema/context/spring-context-3.0.xsd">
10
11   <context:component-scan base-package="com.sshacker.maven.demo"></context:component-scan >
12
13   <bean id="tyre" class="com.sshacker.maven.demo.Tyre">
14     <property name="brand" value="MRF"></property>
15   </bean>
16
17 </beans>
18

```

Constructor Injection

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

```

maven [Spring-Projects master]
src/main/java
  com.sshacker.maven.demo
    App.java
    Bike.java
    Car.java
    Tyre.java
    Vehicle.java
    Bean.xml
src/test/java
JRE System Library [J2SE-1.5]
Maven Dependencies
src
  main
  test
target
pom.xml

```

App.java

```

1 package com.sshacker.maven.demo;
2
3 public class Tyre {
4     private String brand;
5
6     public Tyre(String brand) {
7         super();
8         this.brand = brand;
9     }
10
11    public String getBrand() {
12        return brand;
13    }
14
15    public void setBrand(String brand) {
16        this.brand = brand;
17    }
18
19    @Override
20    public String toString() {
21        return "Tyre [brand=" + brand + "]";
22    }
23
24 }
25

```

The screenshot shows the Eclipse IDE interface with the 'Package Explorer' view on the left and the 'App.java' file in the center. The 'src/main/java/com.sshacker.maven.demo' package contains several Java files: App.java, Bike.java, Car.java, Tyre.java, and Vehicle.java. It also contains Bean.xml and pom.xml. The 'src/test/java' and 'JRE System Library [J2SE-1.5]' are also visible. The 'App.java' file contains XML configuration code for Spring beans.

```
1 <?xml version = "1.0" encoding = "UTF-8"?>
2
3<beans xmlns = "http://www.springframework.org/schema/beans"
4   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
5   xmlns:context = "http://www.springframework.org/schema/context"
6   xsi:schemaLocation = "http://www.springframework.org/schema/beans
7     http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
8     http://www.springframework.org/schema/context
9     http://www.springframework.org/schema/context/spring-context-3.0.xsd">
10
11   <context:component-scan base-package="com.sshacker.maven.demo"></context:component-scan >
12
13   <bean id="tyre" class="com.sshacker.maven.demo.Tyre">
14     <constructor-arg name="brand" value="CEAT"></constructor-arg>
15   </bean>
16
17 </beans>
18
```

Autowired Annotation

we can used mixed mode base configuration
also.

Both using Annotation based & XML base

using only Annotation Based

@Component

{ @Autowired

The screenshot shows the Eclipse IDE interface with the 'Package Explorer' view on the left and the 'App.java' file in the center. The 'src/main/java/com.sshacker.maven.demo' package contains several Java files: App.java, Bike.java, Car.java, Tyre.java, and Vehicle.java. It also contains Bean.xml and pom.xml. The 'src/test/java' and 'JRE System Library [J2SE-1.5]' are also visible. The 'App.java' file contains Java code using the @Component and @Autowired annotations for dependency injection.

```
1 package com.sshacker.maven.demo;
2
3import org.springframework.context.ApplicationContext;
4import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6public class App
7{
8    public static void main( String[] args )
9    {
10        ApplicationContext context = new ClassPathXmlApplicationContext("Bean.xml");
11        Car obj = (Car)context.getBean("car");
12        obj.drive();
13    }
14}
```

Spring-Projects - maven/src/main/java/com/sshacker/maven/demo/Car.java - Eclipse IDE

```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer X App.java Car.java Tyre.java
maven [Spring-Projects master]
src/main/java
com.sshacker.maven.demo
App.java
Bike.java
Car.java
Tyre.java
Vehicle.java
Bean.xml
src/test/java
JRE System Library [J2SE-1.5]
Maven Dependencies
src
main
test
target
pom.xml
1 package com.sshacker.maven.demo;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Component;
5
6 @Component
7 public class Car implements Vehicle{
8
9     @Autowired
10    Tyre tyre;
11
12    public Tyre getTyre() {
13        return tyre;
14    }
15
16    public void setTyre(Tyre tyre) {
17        this.tyre = tyre;
18    }
19
20    public void drive() {
21        System.out.println("Chal raha hai..."+ tyre);
22    }
23 }
24
```

Spring-Projects - maven/src/main/java/com/sshacker/maven/demo/Tyre.java - Eclipse IDE

```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer X App.java Car.java Tyre.java
maven [Spring-Projects master]
src/main/java
com.sshacker.maven.demo
App.java
Bike.java
Car.java
Tyre.java
Vehicle.java
Bean.xml
src/test/java
JRE System Library [J2SE-1.5]
Maven Dependencies
src
main
test
target
pom.xml
1 package com.sshacker.maven.demo;
2
3 import org.springframework.stereotype.Component;
4
5 @Component
6 public class Tyre {
7     private String brand;
8
9     public String getBrand() {
10         return brand;
11     }
12
13     public void setBrand(String brand) {
14         this.brand = brand;
15     }
16
17     @Override
18     public String toString() {
19         return "using Tyre";
20     }
21
22 }
23
```

Configuration Bean

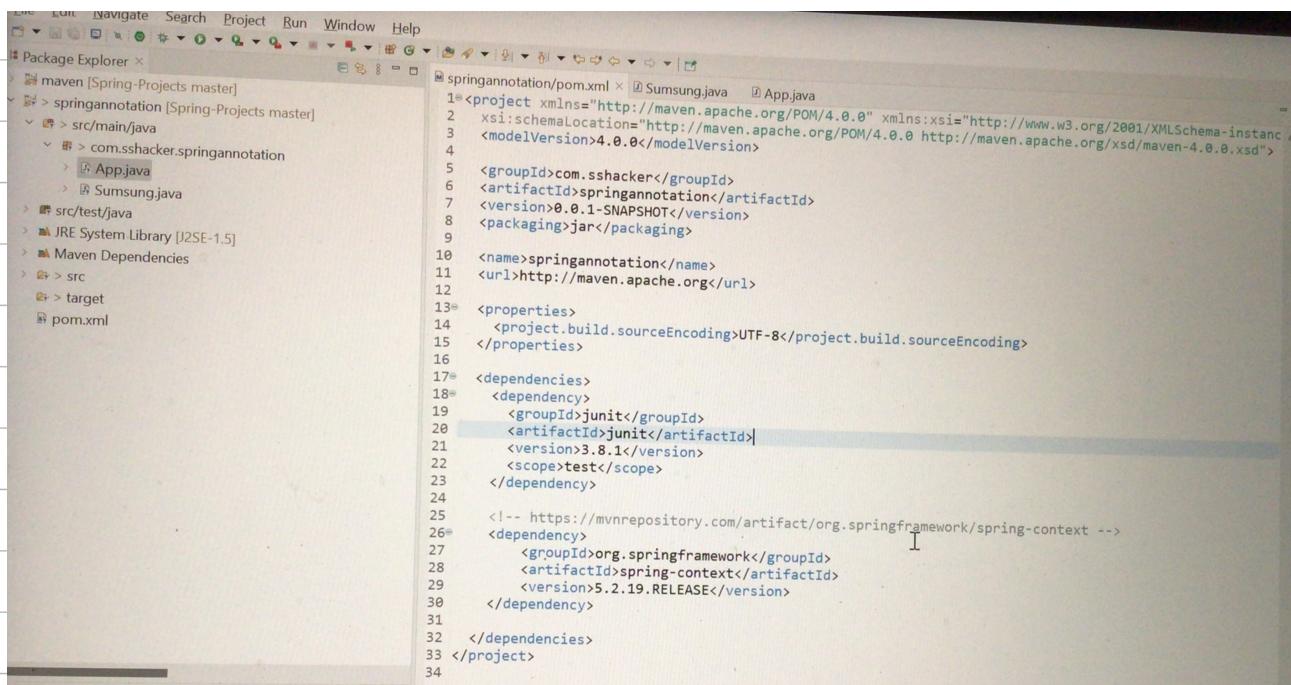
In this

- (1) what is spring framework
- (2) How to use maven to create spring application
- (3) Configure spring Application with the help of Annotation

New Maven Project Creating

Springannotation

we are creating maven project because
Don't want to download required libraries.
maven will provide all dependencies.

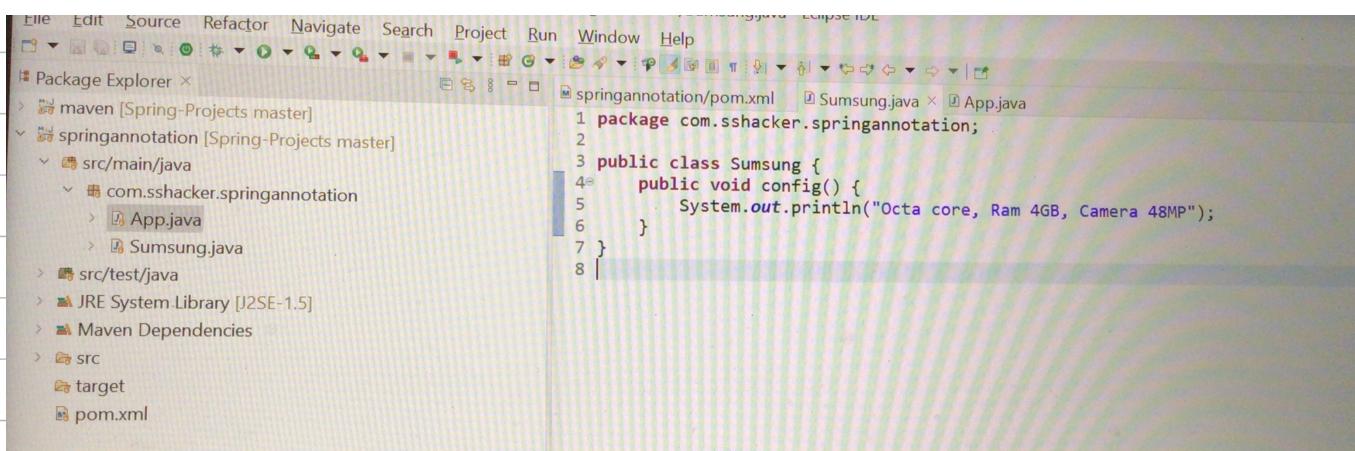


```

File Edit Navigate Search Project Run Window Help
Package Explorer ×
maven [Spring-Projects master]
  > springannotation [Spring-Projects master]
    > src/main/java
      > com.sshacker.springannotation
        > App.java
        > Samsung.java
    > src/test/java
  > JRE System Library [J2SE-1.5]
  > Maven Dependencies
  > src
  > target
  pom.xml

springannotation/pom.xml × Sumsung.java App.java
1<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://www.w3.org/2001/XMLSchema-instance"
3  <modelVersion>4.0.0</modelVersion>
4
5  <groupId>com.sshacker</groupId>
6  <artifactId>springannotation</artifactId>
7  <version>0.0.1-SNAPSHOT</version>
8  <packaging>jar</packaging>
9
10 <name>springannotation</name>
11 <url>http://maven.apache.org</url>
12
13 <properties>
14   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15 </properties>
16
17 <dependencies>
18   <dependency>
19     <groupId>junit</groupId>
20     <artifactId>junit</artifactId>
21     <version>3.8.1</version>
22     <scope>test</scope>
23   </dependency>
24
25   <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
26   <dependency>
27     <groupId>org.springframework</groupId>
28     <artifactId>spring-context</artifactId>
29     <version>5.2.19.RELEASE</version>
30   </dependency>
31
32 </dependencies>
33 </project>
34

```

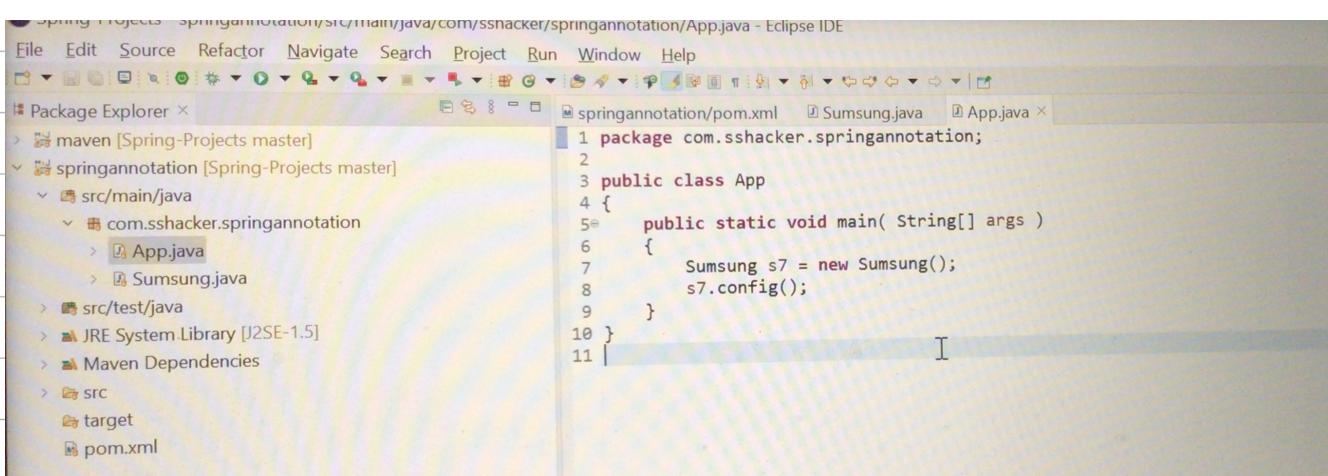


```

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer ×
maven [Spring-Projects master]
  > springannotation [Spring-Projects master]
    > src/main/java
      > com.sshacker.springannotation
        > App.java
        > Samsung.java
    > src/test/java
  > JRE System Library [J2SE-1.5]
  > Maven Dependencies
  > src
  > target
  pom.xml

springannotation/pom.xml × Samsung.java App.java
1 package com.sshacker.springannotation;
2
3 public class Samsung {
4     public void config() {
5         System.out.println("Octa core, Ram 4GB, Camera 48MP");
6     }
7 }
8

```



```

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer ×
maven [Spring-Projects master]
  > springannotation [Spring-Projects master]
    > src/main/java
      > com.sshacker.springannotation
        > App.java
        > Samsung.java
    > src/test/java
  > JRE System Library [J2SE-1.5]
  > Maven Dependencies
  > src
  > target
  pom.xml

springannotation/pom.xml × Samsung.java App.java
1 package com.sshacker.springannotation;
2
3 public class App {
4 {
5     public static void main( String[] args )
6     {
7         Samsung s7 = new Samsung();
8         s7.config();
9     }
10}
11

```

Adding Annotation Based AppConfig file and marking as @Configuration and creating @Bean.

Eclipse Projects - springannotation/src/main/java/com/sshacker/springannotation/App.java - Eclipse IDE

```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer × App.java × Samsung.java × AppConfig.java
maven [Spring-Projects master] src/main/java com.sshacker.springannotation
springannotation [Spring-Projects master]
src/test/java JRE System Library [J2SE-1.5]
Maven Dependencies
src target pom.xml
1 package com.sshacker.springannotation;
2
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
5
6 public class App
7 {
8     public static void main( String[] args )
9     {
10        // Samsung s7 = new Samsung();
11        // s7.config();
12
13        ApplicationContext factory = new AnnotationConfigApplicationContext(AppConfig.class);
14        Samsung obj = factory.getBean(Samsung.class);
15        obj.config();
16
17    }
18
19 }
```

Eclipse Projects - springannotation/src/main/java/com/sshacker/springannotation/Samsung.java - Eclipse IDE

```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer × App.java × Samsung.java × AppConfig.java
maven [Spring-Projects master] src/main/java com.sshacker.springannotation
springannotation [Spring-Projects master]
src/test/java JRE System Library [J2SE-1.5]
Maven Dependencies
src target pom.xml
1 package com.sshacker.springannotation;
2
3 public class Samsung {
4     public void config() {
5         System.out.println("Octa core, Ram 4GB, Camera 48MP");
6     }
7 }
```

Eclipse Projects - springannotation/src/main/java/com/sshacker/springannotation/AppConfig.java - Eclipse IDE

```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer × App.java × Samsung.java × AppConfig.java
maven [Spring-Projects master] src/main/java com.sshacker.springannotation
springannotation [Spring-Projects master]
src/test/java JRE System Library [J2SE-1.5]
Maven Dependencies
src target pom.xml
1 package com.sshacker.springannotation;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5
6 @Configuration
7 public class AppConfig {
8
9     @Bean
10    public Samsung getPhone() {
11        return new Samsung();
12    }
13
14 }
```

Annotation Component

- @Autowired
- @Primary
- @Qualifier ("snapdragon")
- @ComponentScan (basePackage = "com.sshacker")

```
App.java Samsung.java AppConfig.java MobileProcessor.java Snapdragon.java Mediatek.java
1 package com.sshacker.springannotation;
2
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
5
6 public class App
7 {
8     public static void main( String[] args )
9     {
10        // Samsung s7 = new Samsung();
11        // s7.config();
12
13        ApplicationContext factory = new AnnotationConfigApplicationContext(AppConfig.class);
14        Samsung obj = factory.getBean(Samsung.class);
15        obj.config();
16    }
17 }
18 |
```

```
App.java Samsung.java AppConfig.java MobileProcessor.java Snapdragon.java Mediatek.java
1 package com.sshacker.springannotation;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.beans.factory.annotation.Qualifier;
5 import org.springframework.stereotype.Component;
6
7 @Component
8 public class Samsung {
9
10    @Autowired
11    @Qualifier("snapdragon")
12    MobileProcessor cpu;
13
14    public void config() {
15        System.out.println("Octa core, Ram 4GB, Camera 48MP");
16        cpu.brand();
17    }
18 }
19 |
```

```
App.java Samsung.java AppConfig.java MobileProcessor.java Snapdragon.java Mediatek.java
1 package com.sshacker.springannotation;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.ComponentScan;
5 import org.springframework.context.annotation.Configuration;
6
7 @Configuration
8 @ComponentScan(basePackages = "com.sshacker.springannotation")
9 public class AppConfig {
10
11    // @Bean
12    // public Samsung getPhone() {
13    //     return new Samsung();
14    // }
15    //
16    // @Bean
17    // public MobileProcessor getProcessor() {
18    //     return new Snapdragon();
19    // }
20
21 }
22 |
```

sshacker/springannotation/MobileProcessor.java - Eclipse IDE

Object Run Window Help

```
1 package com.sshacker.springannotation;
2
3 public interface MobileProcessor {
4     void brand();
5 }
```

```
1 package com.sshacker.springannotation;
2
3 import org.springframework.stereotype.Component;
4
5 @Component
6 public class Snapdragon implements MobileProcessor {
7
8     public void brand() {
9         System.out.println("Snapdragon wolrd's best cpu");
10    }
11 }
12 |
```

```
1 package com.sshacker.springannotation;
2
3 import org.springframework.context.annotation.Primary;
4 import org.springframework.stereotype.Component;
5
6 @Component
7 @Primary
8 public class Mediatek implements MobileProcessor {
9
10    public void brand() {
11        System.out.println("Mediatek 2nd best processor");
12    }
13 }
14 |
```