Spring framework :-

1) 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 sperate 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

---

1. 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 follow this approch. In Industries.

Dependency means

some object depends on another object.

Object Graph

example :

Laptop building

```
class Laptop {
    HardDrive obj1;
    Ram obj2;
}
```

class Laptop {

HitachiHD obj1 = new HitachiHD();

}

Tight coupling

→ loose coupling :-
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 approche)
we can change .xml file later
if Some one ask for ~~Hard~~Drive object
give him Hitachi HD object.

Its mention in .xml file in Spring Configuration

So there is no tight coupling

② Second way to Spring Boot approche latest approche annotation.

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

```
class Laptop
{
  @Autowired
  HardDrive objL;
}

@Component
class HitachiHD implements HardDrive
{
  ——
}
```

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

@Autowired :-
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 maintainace 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.

① Create a project to work with MySQL we need mysql connector. library jar dependencies

② to work with Spring MVC
we need 10-12 libraries jar dependency

③ to work with hibernate
we need 10-12 libraries jar 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 mvnrepository

Advantage of Maven

① Diffent Goal to achieve
② compiled & build
③ deploy
④ provide dependency jar libraries.

dependency = jar file

pom.xml

*aa/pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-insta
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.telusko</groupId>
  <artifactId>aa</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>aa Maven Webapp</name>
  <url>http://maven.apache.org</url>

  <dependencies>

    Hey I need Hibernate.....

  </dependencies>

  <build>
    <finalName>aa</finalName>
  </build>
</project>
```

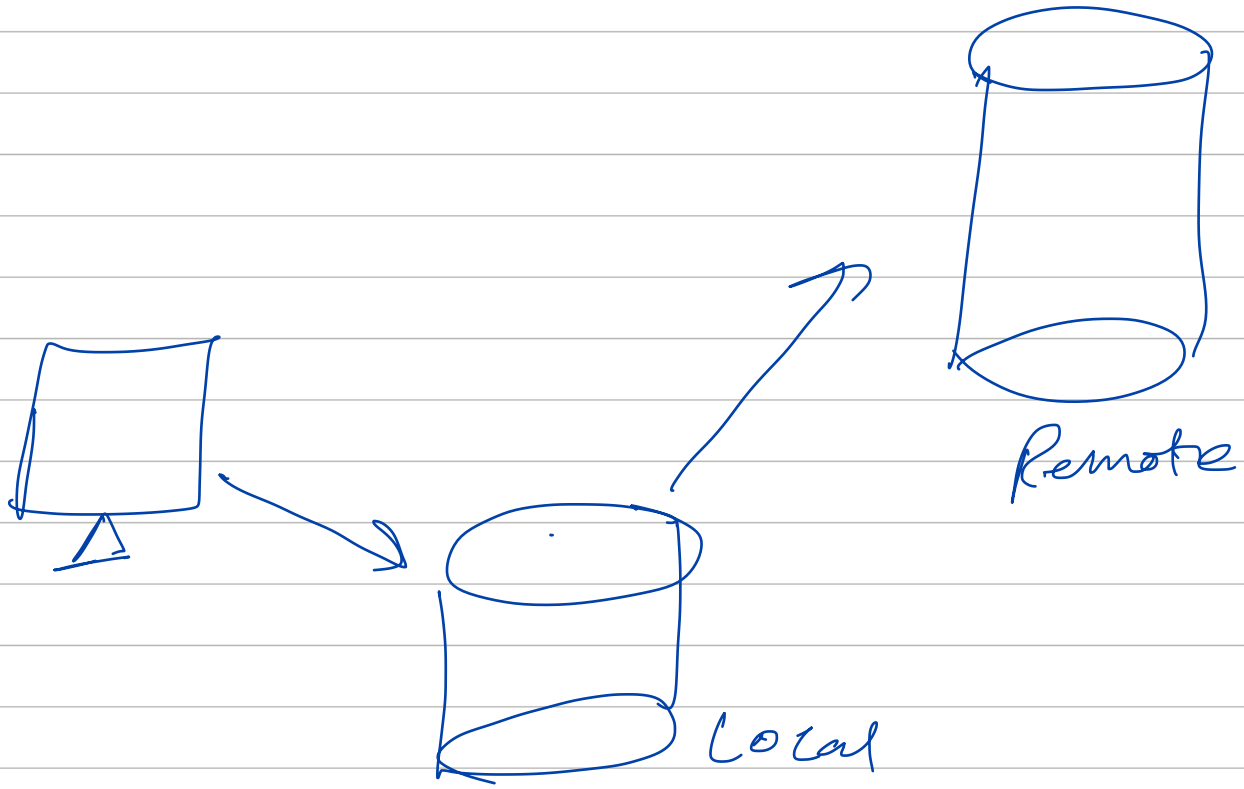
1. Project
2. GroupId - com.telusko
3. ArtifactId - demo
4. Package - com.telusko.demo.web

```
<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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.telusko</groupId>
  <artifactId>aa</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>aa Maven Webapp</name>
  <url>http://maven.apache.org</url>

  <dependencies>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>4.3.8.RELEASE</version>
    </dependency>

  </dependencies>

  <build>
    <finalName>aa</finalName>
  </build>
```

Note:- Instead of going individual website for dependency jar download there is a central website

mvn 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 develope 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 a core java application Simple Maven Project

maven-archetype-quickstart

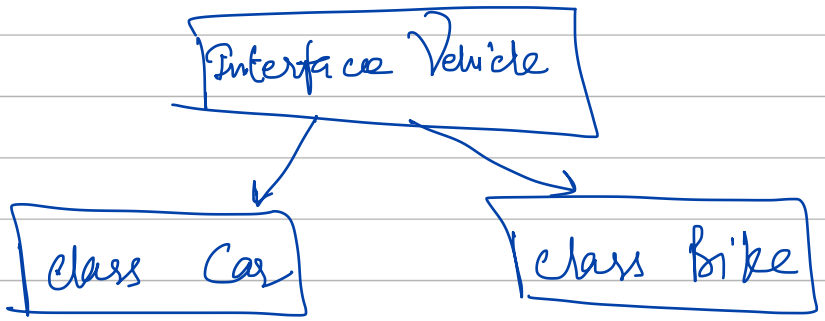
for web application

maven - archetype - web

Spring Core framework Tutorial

Application Context

① Using Interfaces



② Using Spring (.xml file)

App.class →

main method

for small Application

for Enterprize 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();
}
```

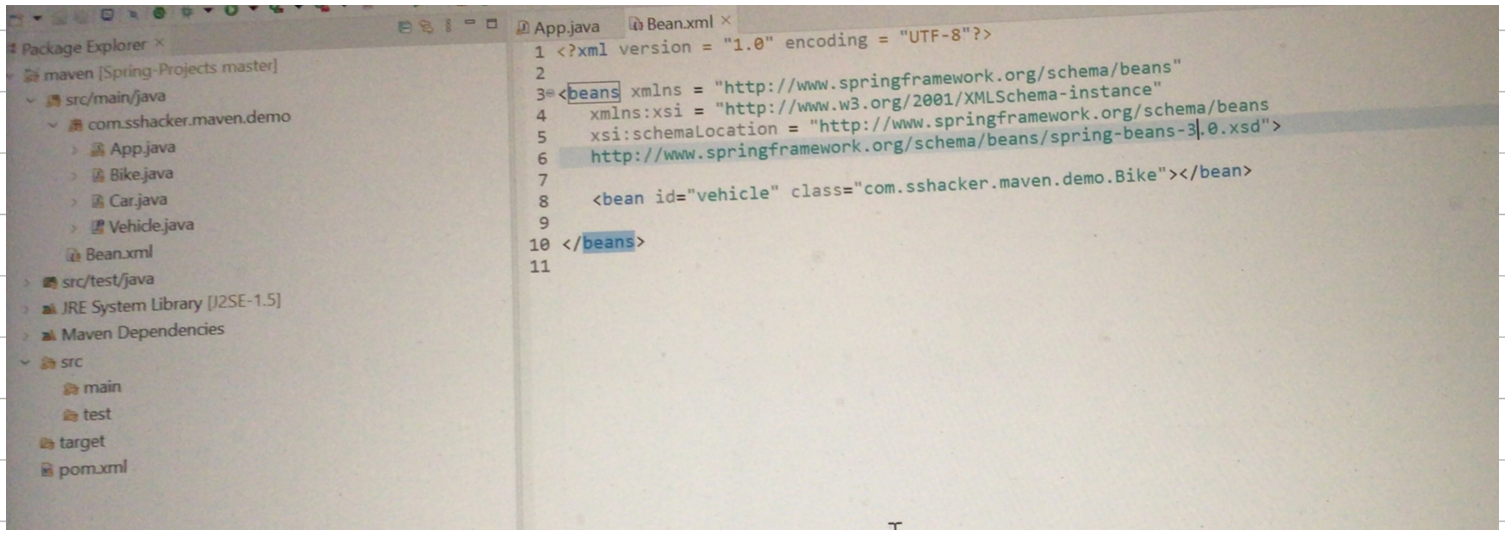
Spring Core framework

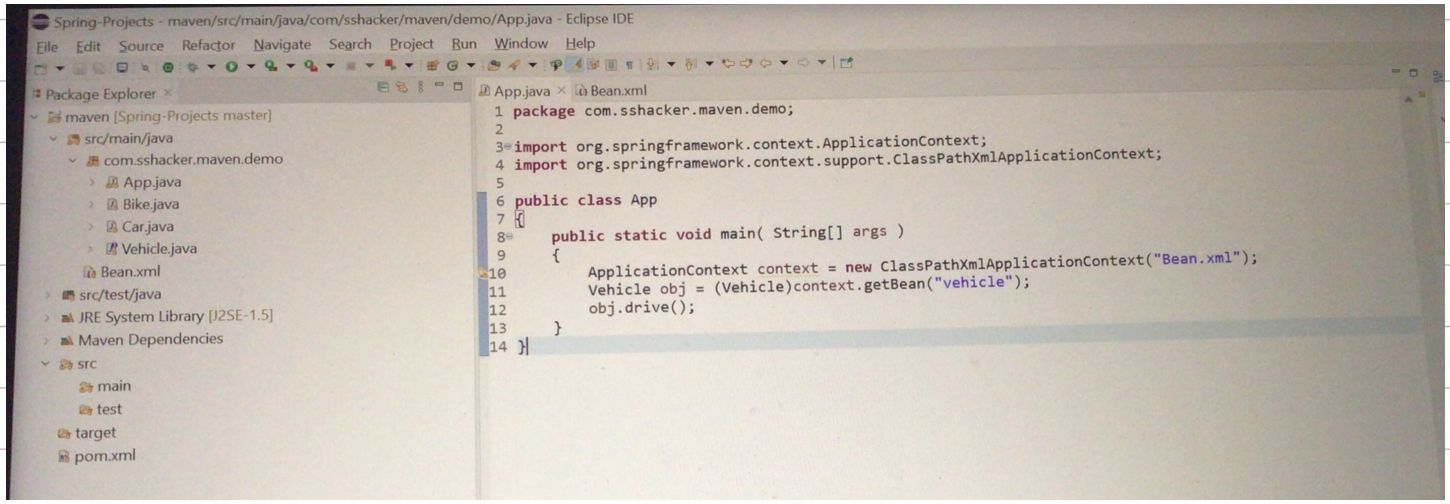
To create xml file to mention the object which should be pass.

We do not needs to recompile java whole program if we change object passing in xml file.

we can take any name.

Bean.xml → at Class Path



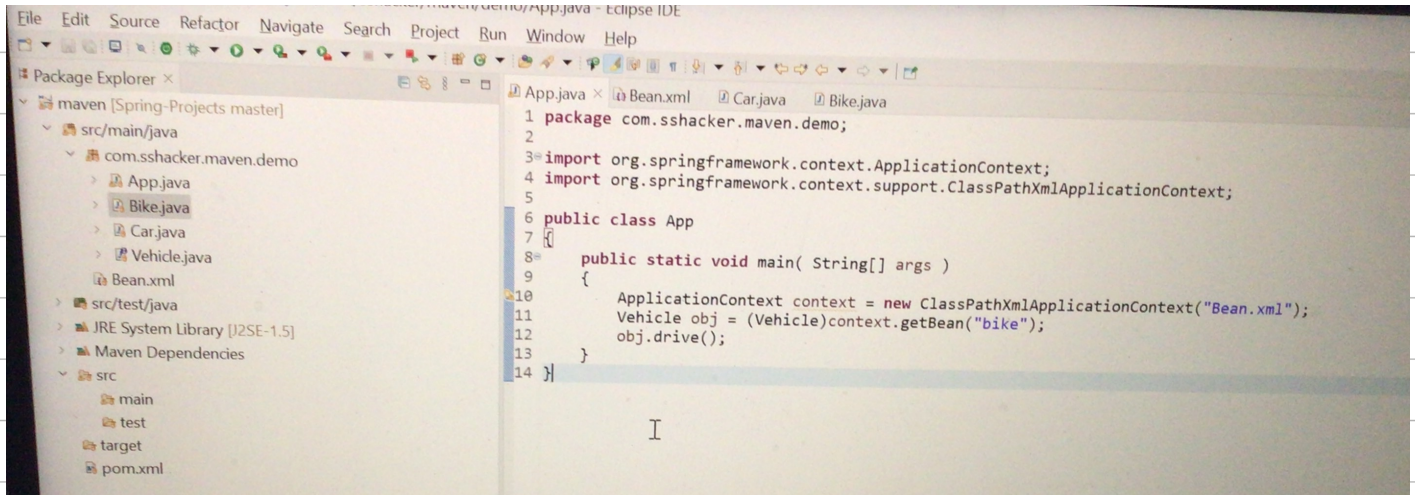


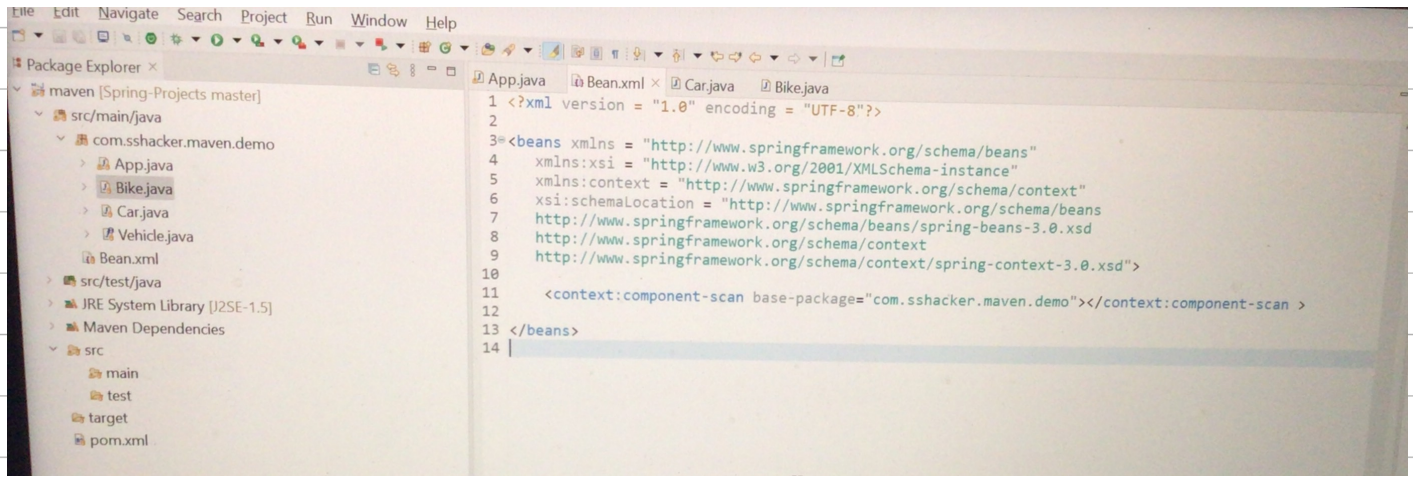
Annotation Based Configuration

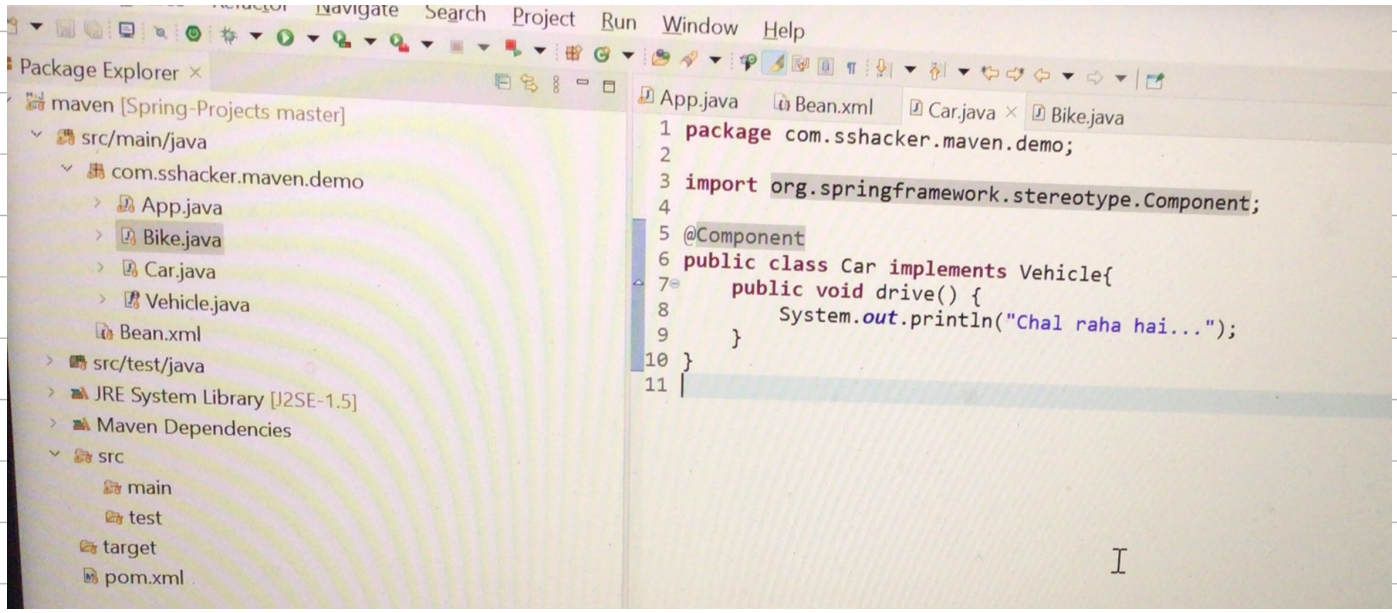
3 types of Configurations

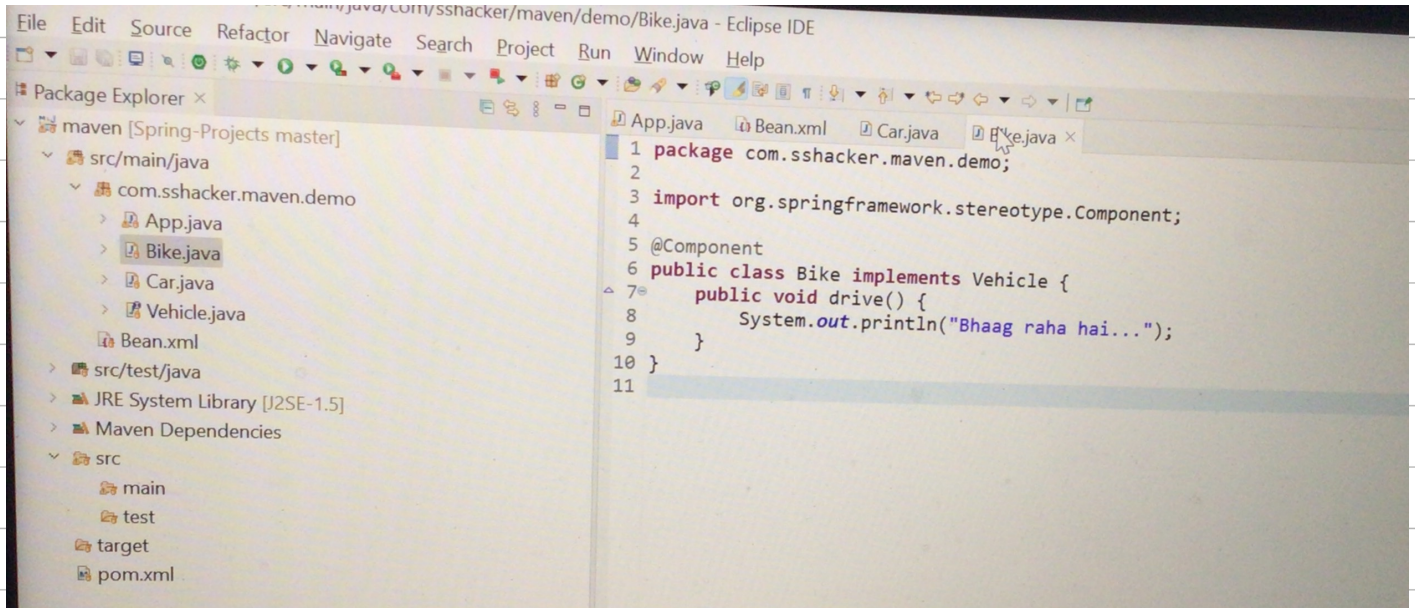
1. xml Based
2. Annotation Based
3. java Based.

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

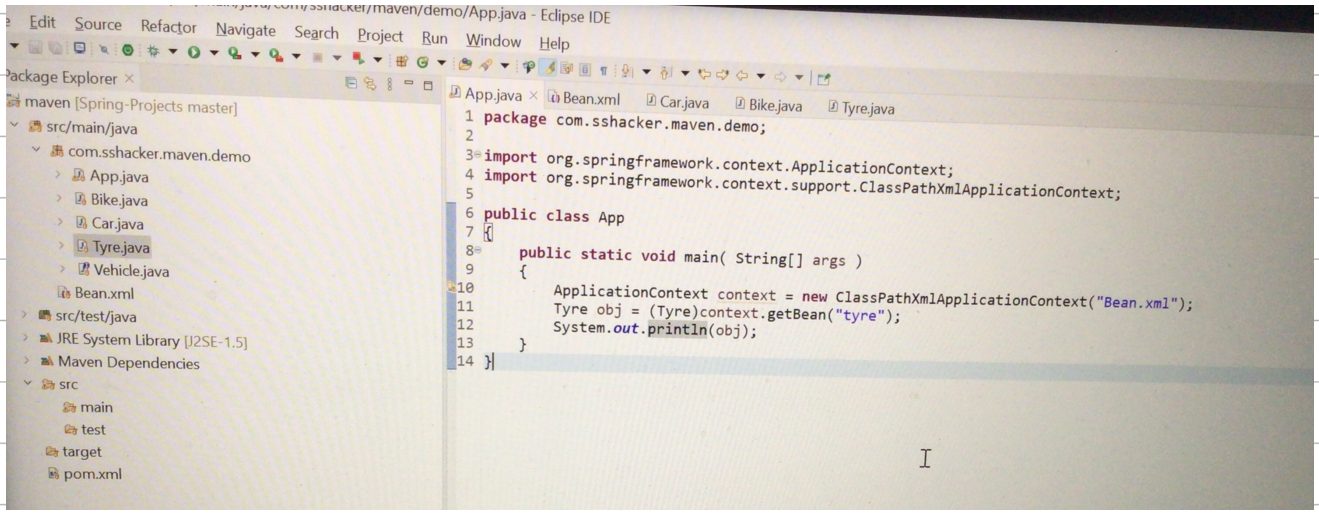


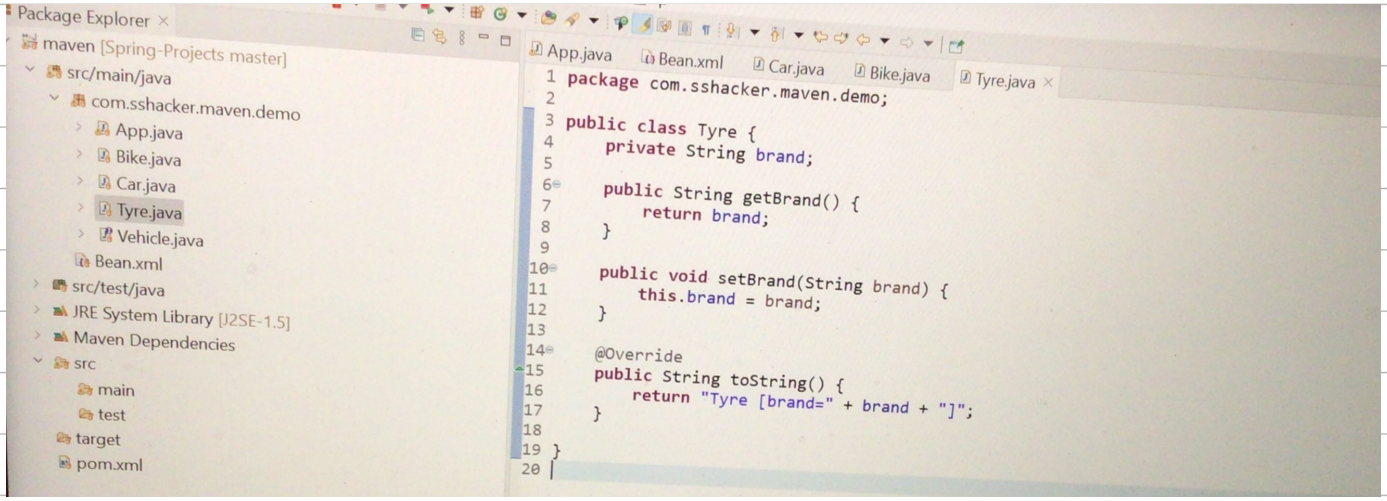






Bean Properties



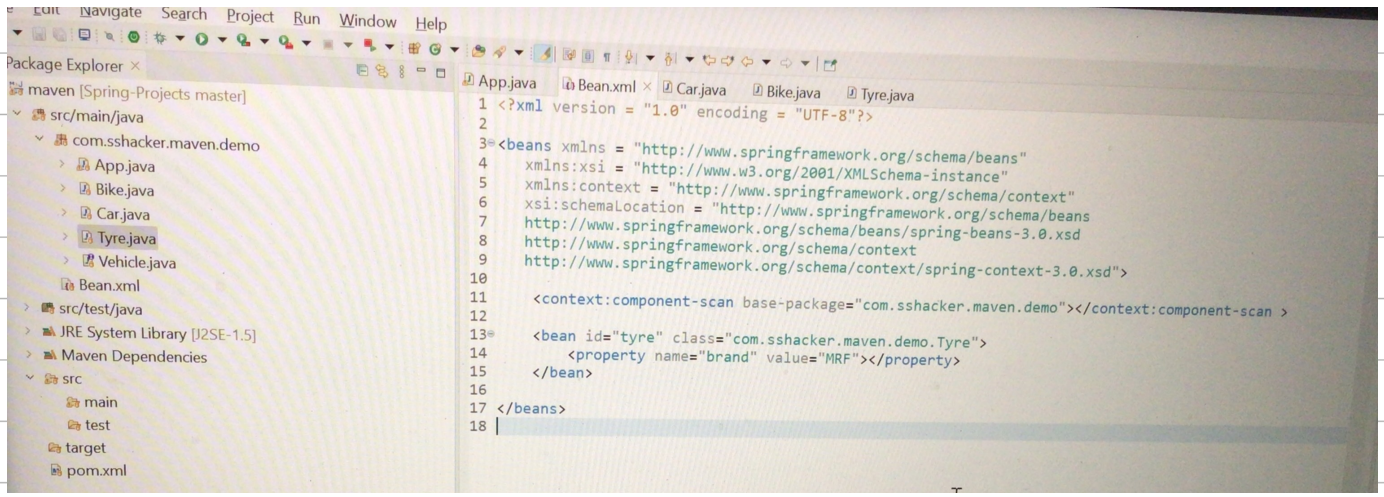


Package Explorer

- 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 | Bean.xml | Car.java | Bike.java | Tyre.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
```



Edit Navigate Search Project Run Window Help

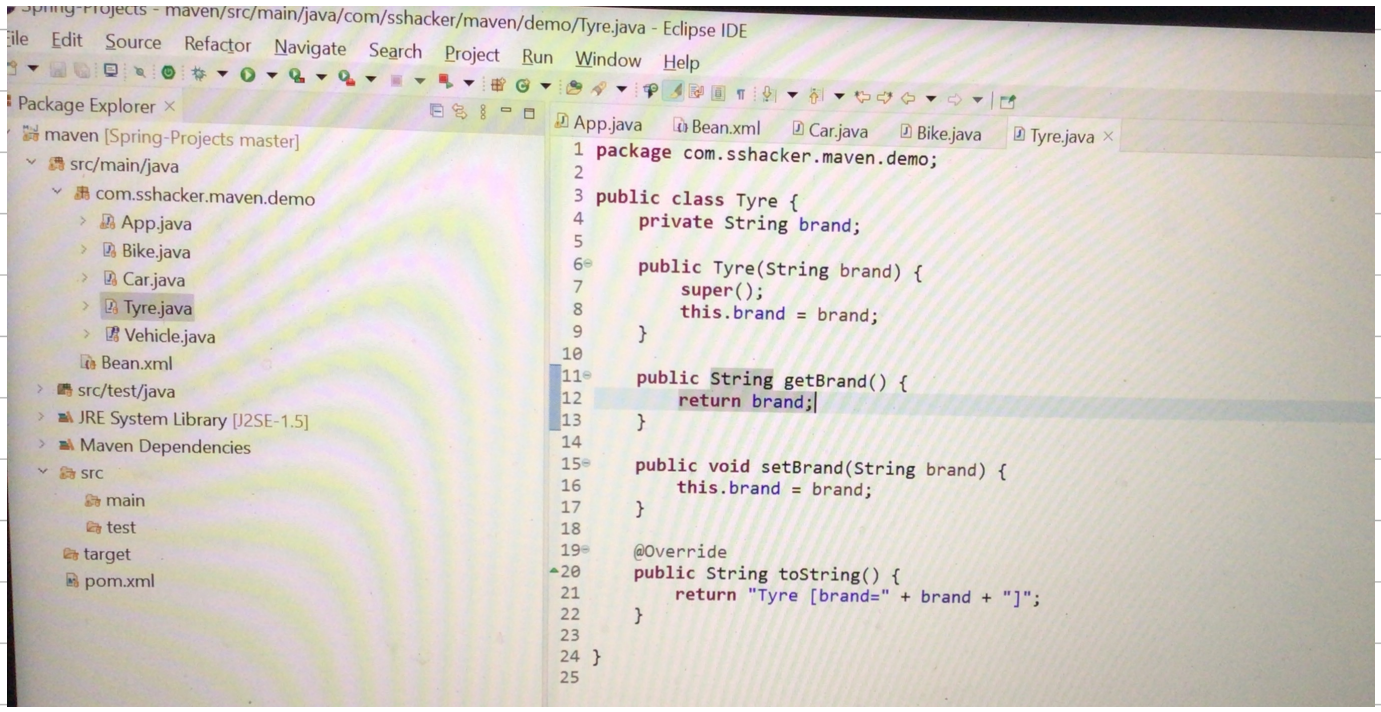
Package Explorer

- 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 | Bean.xml | Car.java | Bike.java | Tyre.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



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

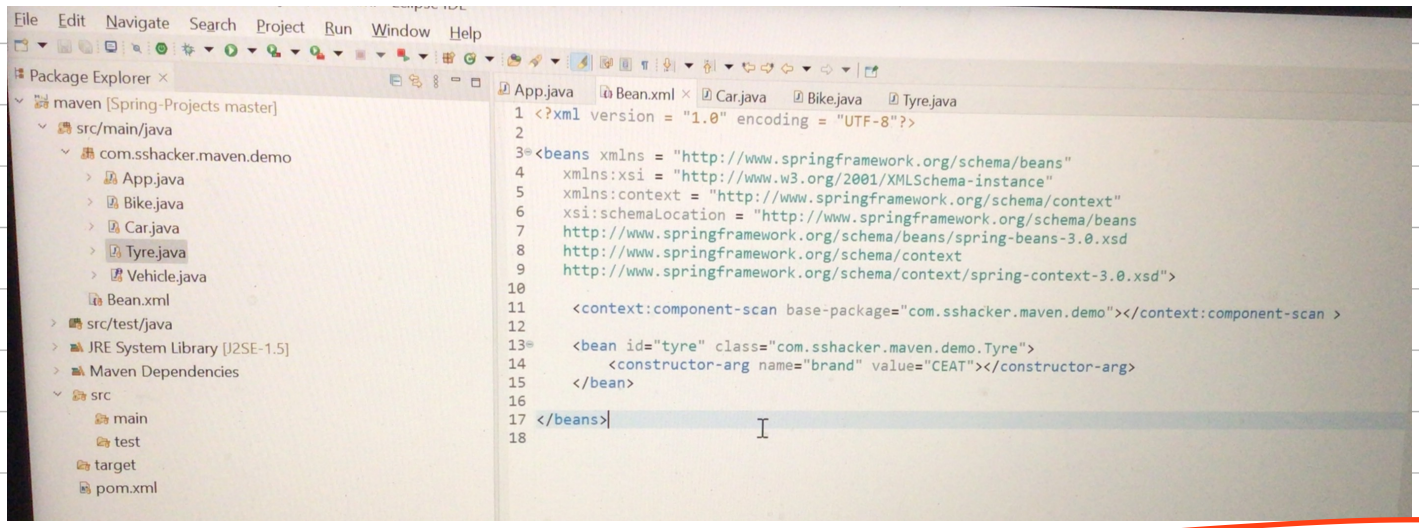
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- 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 | Bean.xml | Car.java | Bike.java | Tyre.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
```



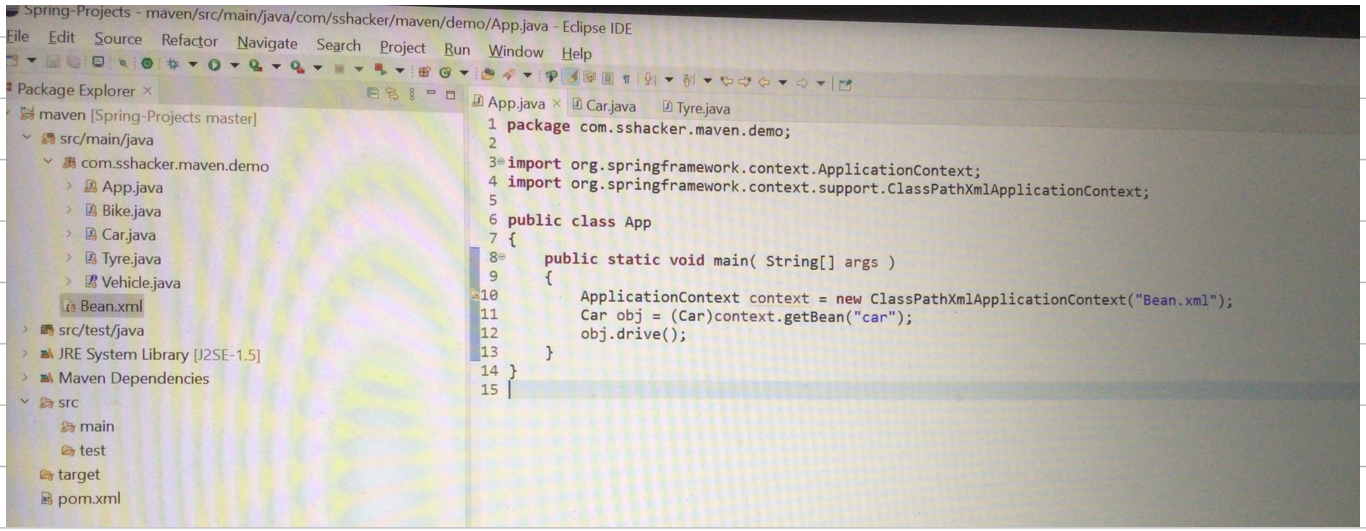
Autowired Annotation

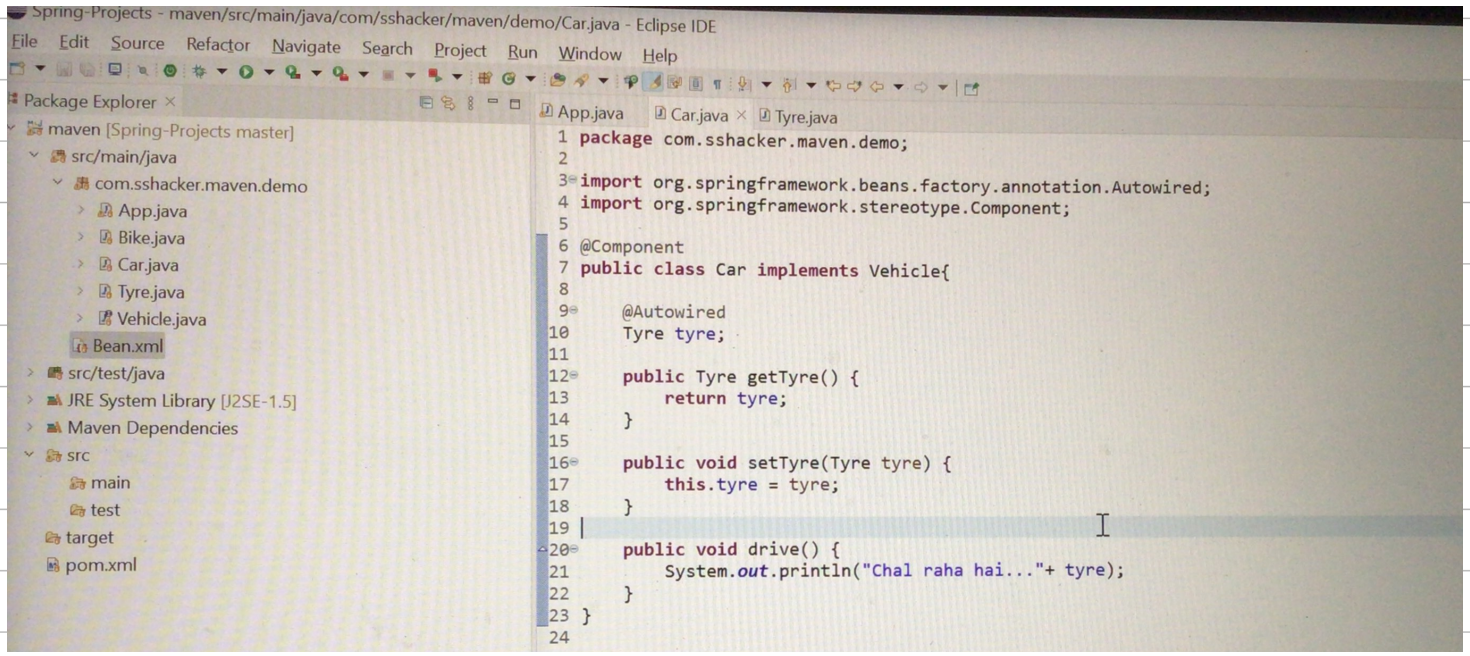
We can used Mixed mode base configuration also.

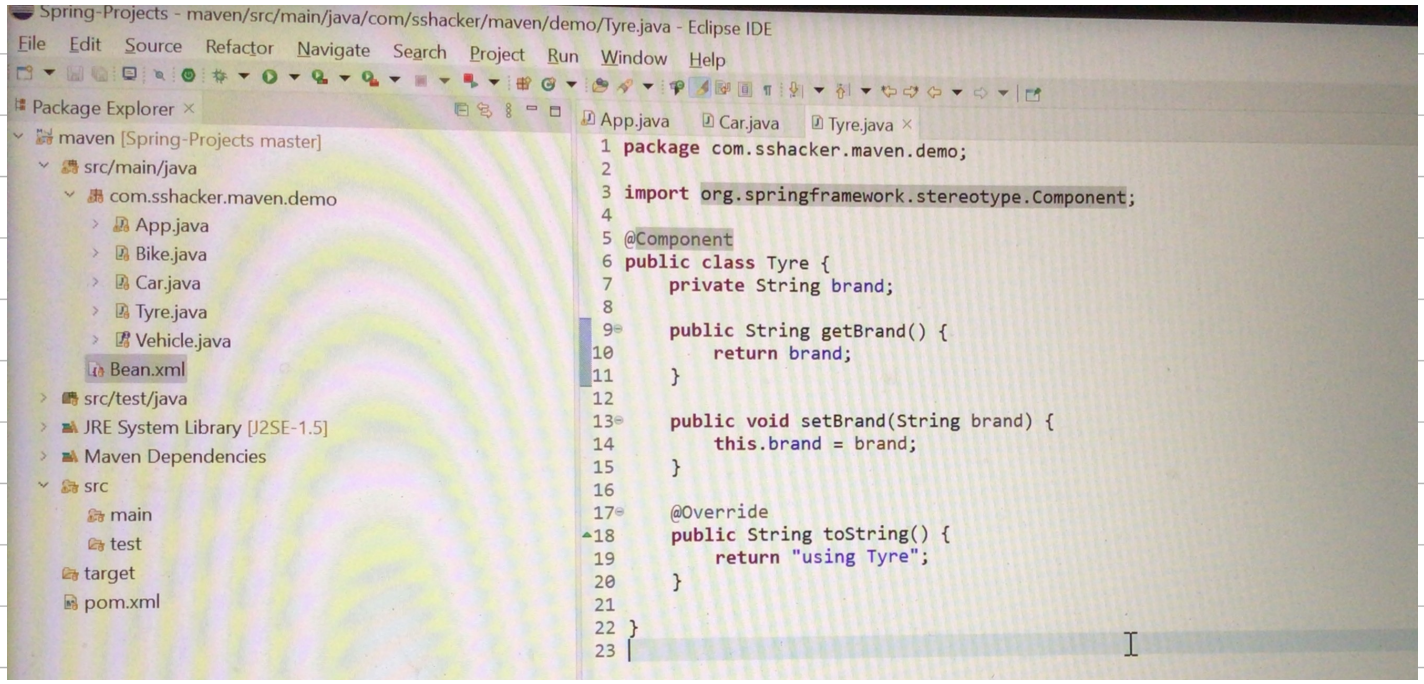
Both using Annotation based & Xml base

using only Annotation Based

@Component & @Autowired







Configuration Bean

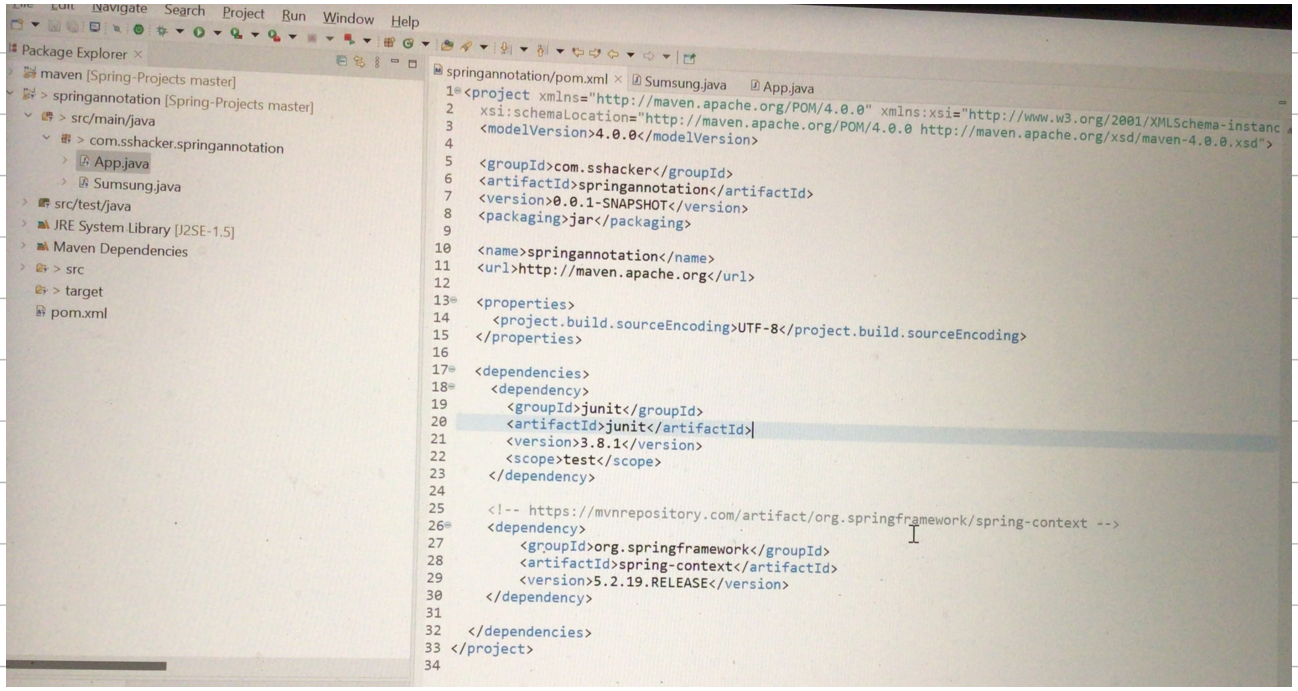
In this

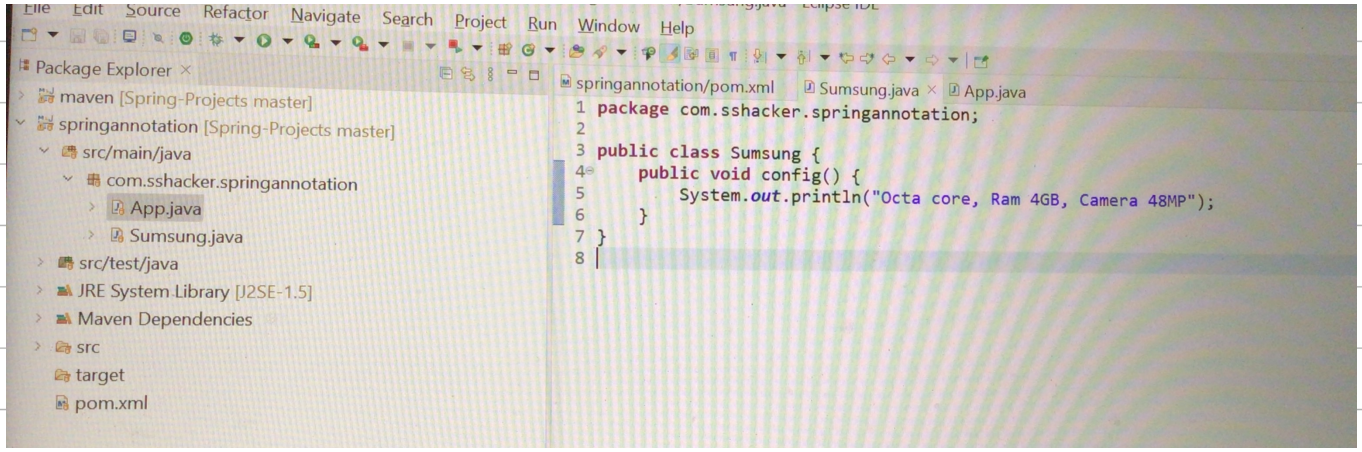
1. what is spring framework
2. How to use maven to create spring application
3. Configure spring Application with the help of Annotation.

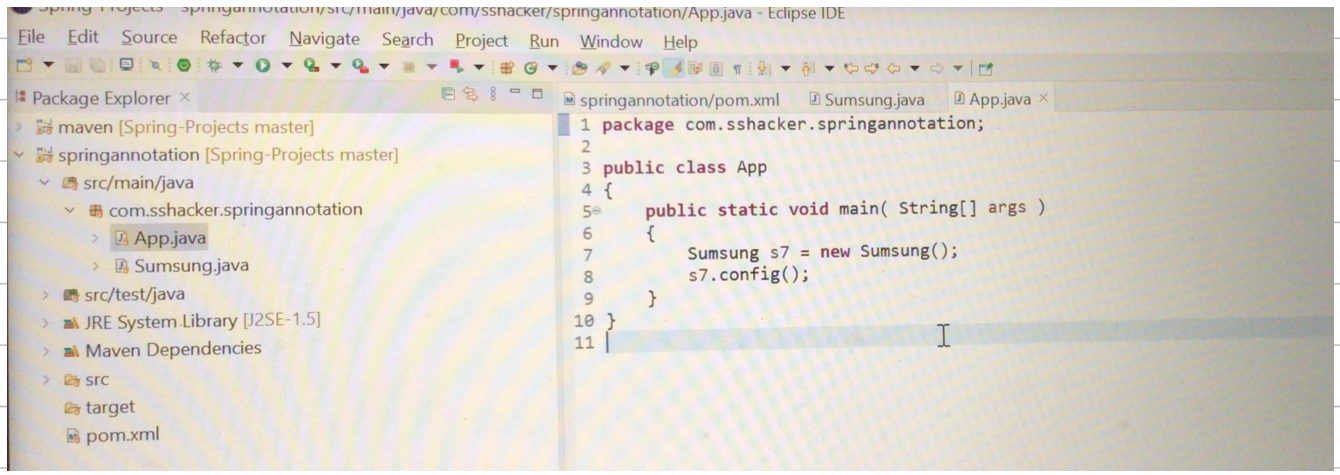
New Maven Project creating

Springannotation

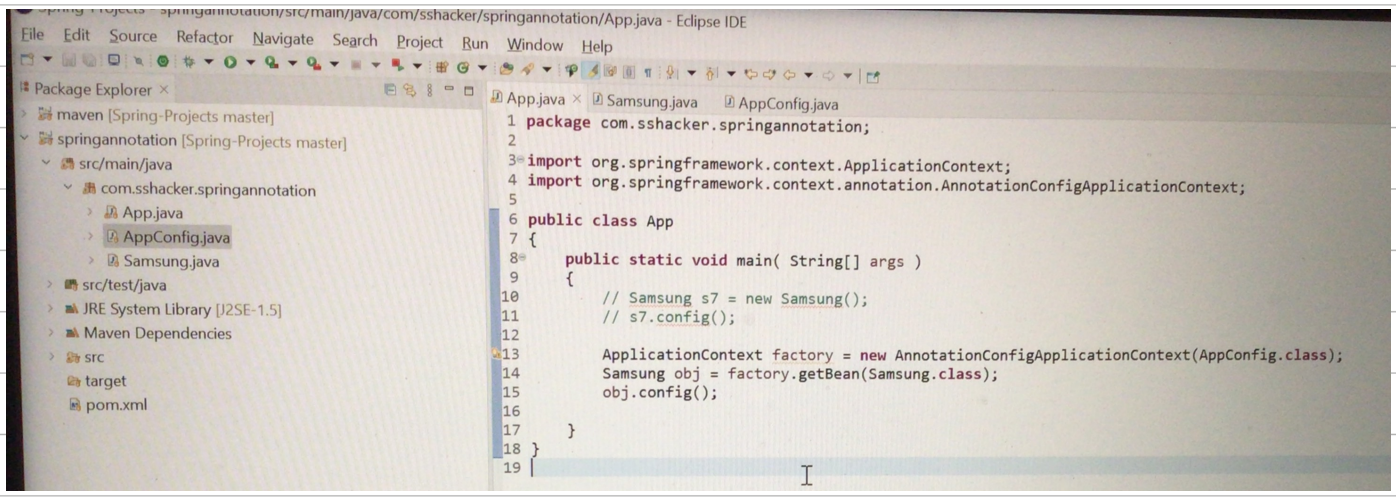
we are creating maven project because Don't want to download required libraries. maven will provide all dependencies.

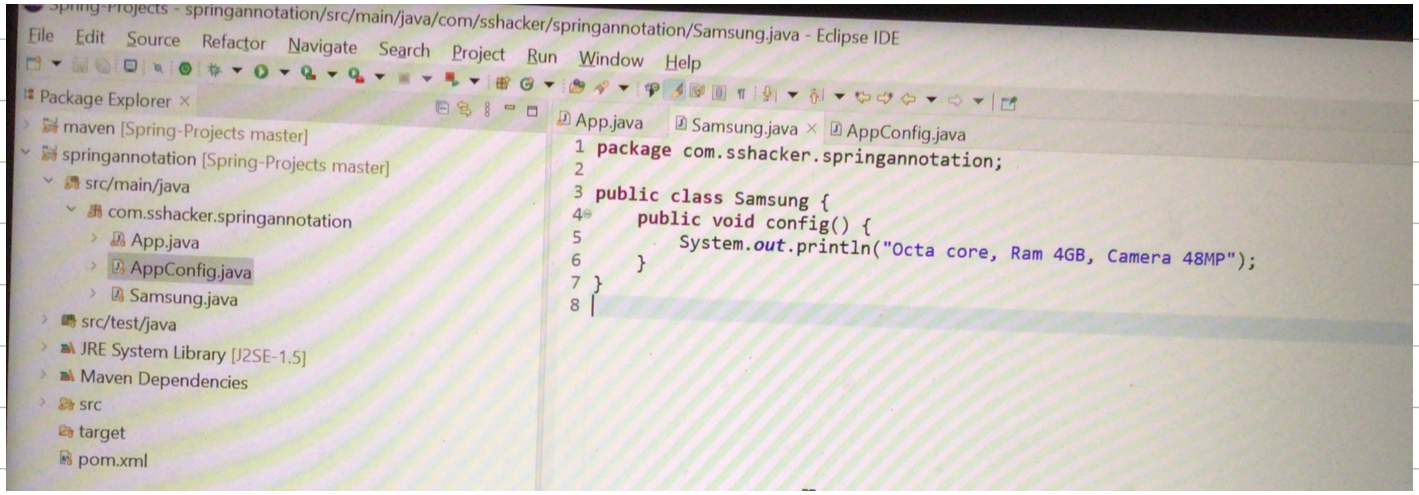


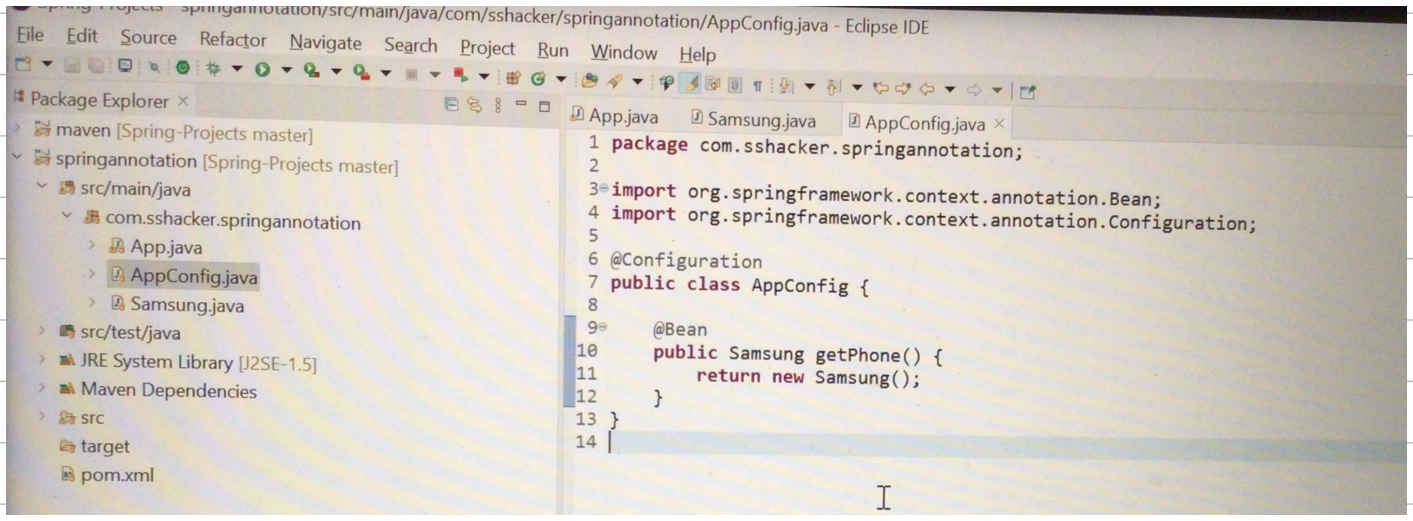




Adding Annotation Based AppConfig file and marking as @Configuration and creating @Bean.







Annotation Component

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

App.java × Samsung.java AppConfig.java MobileProcessor.java Snapdragon.java Mediatek.java

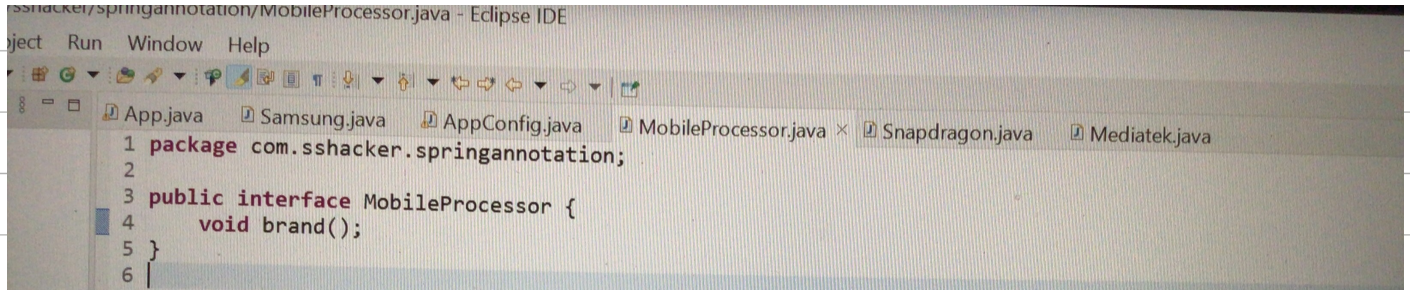
```java
package com.sshacker.springannotation;

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

public class App
{
    public static void main( String[] args )
    {
        // Samsung s7 = new Samsung();
        // s7.config();

        ApplicationContext factory = new AnnotationConfigApplicationContext(AppConfig.class);
        Samsung obj = factory.getBean(Samsung.class);
        obj.config();
    }
}
```

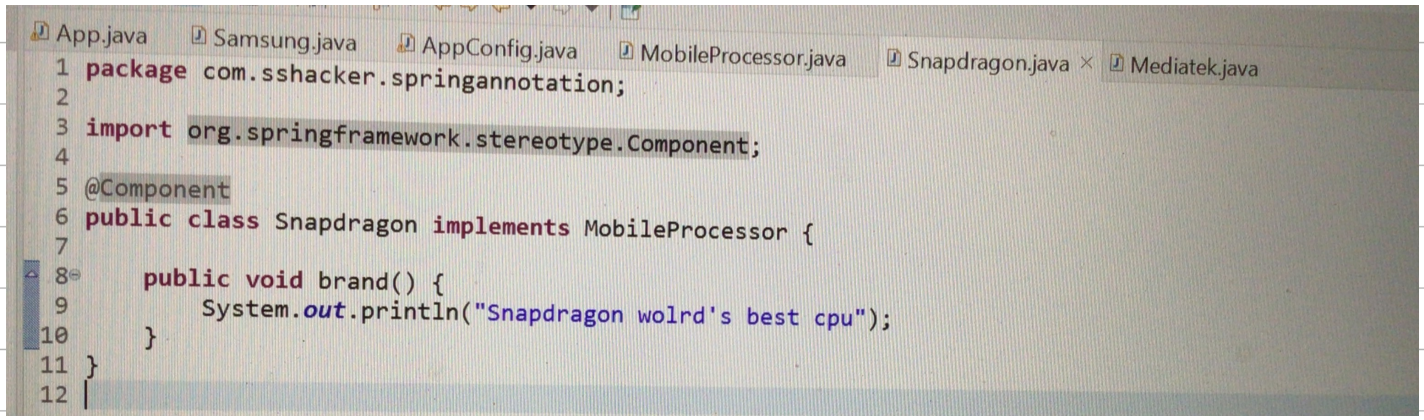
App.java Samsung.java × AppConfig.java MobileProcessor.java Snapdragon.java Mediatek.java

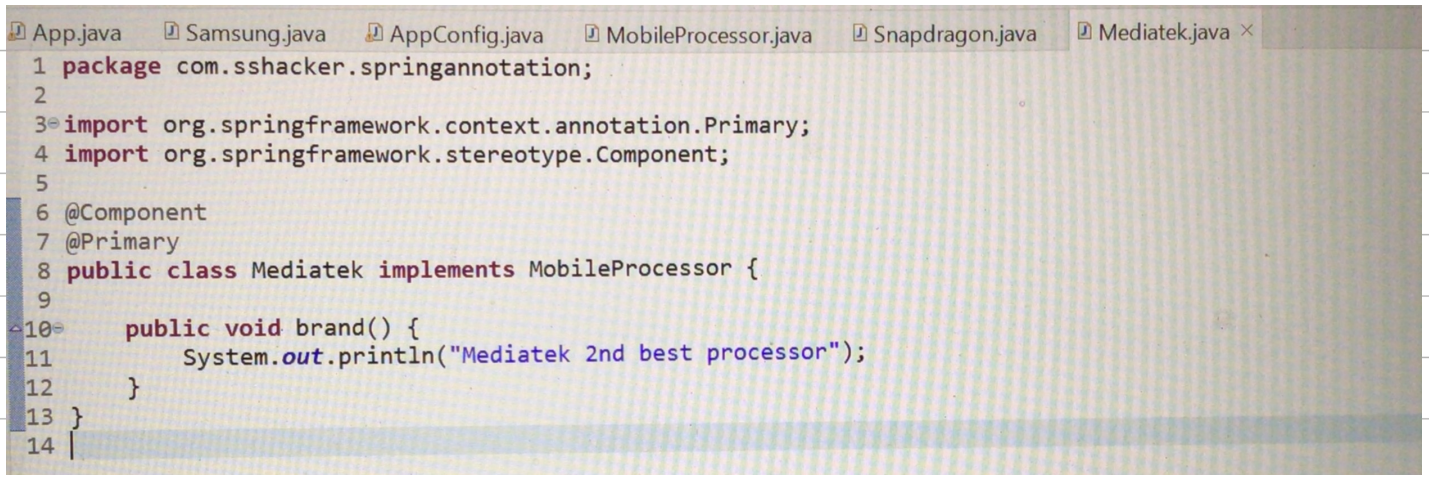
```java
package com.sshacker.springannotation;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Component;

@Component
public class Samsung {

    @Autowired
    @Qualifier("snapdragon")
    MobileProcessor cpu;

    public void config() {
        System.out.println("Octa core, Ram 4GB, Camera 48MP");
        cpu.brand();
    }
}
```

App.java Samsung.java AppConfig.java × MobileProcessor.java Snapdragon.java Mediatek.java

```java
package com.sshacker.springannotation;

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

@Configuration
@ComponentScan(basePackages = "com.sshacker.springannotation")
public class AppConfig {

//  @Bean
//  public Samsung getPhone() {
//      return new Samsung();
//  }
//
//  @Bean
//  public MobileProcessor getProcessor() {
//      return new Snapdragon();
//  }

}
```


sshacker/springannotation/MobileProcessor.java - Eclipse IDE
ject Run Window Help
App.java
Samsung.java
AppConfig.java
MobileProcessor.java
Snapdragon.java
Mediatek.java
1 package com.sshacker.springannotation;
2
3 public interface MobileProcessor {
4 void brand();
5 }
6


App.java
Samsung.java
AppConfig.java
MobileProcessor.java
Snapdragon.java
Mediatek.java
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


App.java
Samsung.java
AppConfig.java
MobileProcessor.java
Snapdragon.java
Mediatek.java
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