SpringTuts

Spring provides dependency injection
Spring provides MVC Framework
Very easy to work with maven

Maven

maven handels all library dependency from internet
new ->others ->maven->maven project

give groupId and artifactId

create maven project

for spring framework add following into the dependency tag in pom.xml file

taken from maven repository https://mvnrepository.com/artifact/org.springframework/spring-context/5.2.2.RELEASE

```
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.2.2.RELEASE</version>
        </dependency>
```

save and update maven
itwill download spring dependency from maven

Dependency Injection

Car.java

```
package subhajit.com.blog;

public class Car implements Vehicle {

        public void drive()
        {
                System.out.println("car is running");
        }

}
```

Bike.java

```
package subhajit.com.blog;

public class Car implements Vehicle {

        public void drive()
        {
                System.out.println("car is running");
        }

}
```

Interface Vehicle.java

```
package subhajit.com.blog;

public interface Vehicle {
        void drive();

}
```

App.java

```
package subhajit.com.blog;
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
        Vehicle vehicle =new Bike();
        vehicle.drive();
    }
}
```

Spring provides better dependency Injection through ApplicationContext

```
package subhajit.com.blog;

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

public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
        ApplicationContext context=new ClassPathXmlApplicationContext("spring.xml");
        Vehicle vehicle =(Vehicle)context.getBean("vehicle");
        vehicle.drive();
    }
}
```

we need spring xml file for 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.xsd">

        <bean id="vehicle" class="subhajit.com.blog.Car"></bean>
</beans>
```

Anotation based configuration

spring .xml file

```
<?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.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">
    <context:component-scan base-package="subhajit.com.blog"></context:component-scan>


 <bean id="vehicle" class="subhajit.com.blog.Car"></bean>
</beans>
```

App.java

```
package subhajit.com.blog;

import org.springframework.context.ApplicationContex
import org.springframework.context.support.ClassPath

public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
        ApplicationContext context = new ClassPathX
        Vehicle vehicle =(Vehicle)context.getBean("
        Vehicle bike =(Vehicle)context.getBean("bik
        vehicle.drive();
        bike.drive();
    }
}
```

bike.java

```
package subhajit.com.blog;

import org.springframework.stereotype.Component;

@Component
public class Bike implements Vehicle  {
        public void drive()
        {
                System.out.println("Bike is running");
        }

}
```

Pr+operty in Deopendency injection (setter injection)

```
<bean id="tyre" class="subhajit.com.blog.Tyre">
 <property name="brand" value="MRF"></property>
 </bean>
```

Annotation

App.java
MobileProcessor.java

```java
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
        ApplicationContext factory = new AnnotationConfigApplicationContext(AppConfig.class);
        Samsung s7=factory.getBean(Samsung.class);
        s7.call();
        s7.StartJob();

    }
}
```

```java
package Mobi

public inter

        void

}
```

Samsung.java
AppConfig.java

```java
package Mobiles.samsung;
@Component
public class Samsung {
        @Autowired
        @Qualifier("snapDragon")
        MobileProcessor sd;
        void call()
        {
                System.out.println("Calling samsung HQ");
        }
        void StartJob()
        {
                System.out.println("Start the job ...");
                sd.process();
        }
}
```

```java

@Configuration
@ComponentScan(basePackages = "Mobiles.samsung")
public class AppConfig {
//      @Bean
//      Samsung getSamsung()
//      {
//              return new Samsung();
//      }
//      @Bean
//      MobileProcessor getSnapDragon()
//      {
//              return new SnapDragon();
//      }

}
```

SnapDragon.java
Mediatek.java

```java
@Component
public class SnapDragon implements MobileProcessor{
        public void process()
        {
                System.out.println("SnapDragon Processsing");
        }
}
```

```java
@Component
@Primary
public class mediaTek implements MobileProce
        public void process() {
                System.out.println("Mediatak

        }
}
```

Annotations

@Configuration :-->To Declare it is an Configuration Class

@Component :→ To Declare it an Component Alternative you have to use @Bean tag with beanGetter

@Bean :-->To declare it is a Bean

@ComponentScan(basePackages) :->To declaure where to get those components

@AutoWired :--->Autometicaly get Beans of type

@Primary :-->inclase of Beans of multipleSame type declaire one primary

@Quailifier (string name) :->also in case of conflict (multiple Same type) use to specify