Uttara InfoSolutions

www.uttarainfo.com

Spring Practicals 1

Please do the following:

First unzip the zip given to you

1) In Eclipse, create a Java Project with name FirstApp
a) Right click and create folder -> lib.
copy paste jar files from PerformerIdolApp->lib folder
b) Right click on project->build path->libraries->add jars-> select lib folder jars to add to build path

1) Create a Car class with name and int bhp as instance variables (package com.uttara.test).Add a dummy drive() with SOP.
Have a no-arg constructor and a param constr that accepts name and bhp as parameter. Generate setters&getters as well. Override equals(),hashCode(),toString()
2) Right click on project -> file->new->Spring Bean configuration file -> name it spring.xml
3) Configure in beans element this:

```
<bean id="nano" class="com.uttara.test.Car"/>
```

4) Create a TestCars class with main().
Code this:

```
ClassPathXmlApplicationContext ctx =
new
ClassPathXmlApplicationContext("spring.xml");
Car c1 = (Car) ctx.getBean("nano");
c1.drive();
```

Run this and test. Invoke ctx.getBean("nano") multiple times and verify if different obj or same obj ref are getting injected. Why? How to change this?

5) Create a constructor-arg sub element in bean to inject a name and bhp as values.

```
<bean id="nano" class="com.uttara.test.Car">
```

```
        <constructor-arg value="nano"/>
        <constructor-arg value="40"/>
</bean>
```
Verify if the car object has the injected state or not.

6) Using property sub element, inject properties into another car object
7) Create Engine interface and TruckEngine and NanoEngine classes that implement this.
8) Add an instance variable in Car of type Engine engine; Ask SpC to inject NanoEngine into one car and TruckEngine into another.
9) Try out other examples by looking at the demoed ex code.

Turning on Annotations Steps for the prior example:

1) In spring.xml-> add context namespace
2) remove all bean configurations
3) turn on component scanning using <context:component-scan base-package="com.uttara.spring”/>

Your spring xml should look like this now:

```
<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-4.0.xsd">

    <context:component-scan base-package="com.uttara.test"></context:component-scan>

</beans>
```

4) Use @Component("<whatever id>") in Car and Engine implementation classes. Use @AutoWired on setter methods for property injection. Use @Qualifier("<id>”) to choose which bean to be injected based on id filtering.
Use @Value next to your fields to inject values during object construction.

Ex:

```
@Component("myCar")
public class Car {

    @Value("MyCar")
    String name;
    @Value("150")
    int bhp;

    Engine eng;

    @Autowired
    @Qualifier("eng")
    public void setEng(Engine eng) {
```

```
        this.eng = eng;
        System.out.println("inside setEng() "+eng);
    }
…
…
}
```

5) Run the same tester class that you had coded earlier to test dependency injection with auto wiring using annotation configurations!