Java-Success.com

Industrial strength Java/JEE Career Companion to open more doors

Register | Login | Logout | Contact Us

search here … Go

Home | Java FAQs | 600+ Java Q&As | Career | Tutorials | Member | Why?

Can u Debug? | Java 8 ready? | Top X | Productivity Tools | Judging Experience?

Home › Interview › Core Java Interview Q&A › Differences Between X & Y › Differences between Spring XML and Spring JavaConfig (@Configuration)

Differences between Spring XML and Spring JavaConfig (@Configuration)

Posted on May 30, 2015 by Arulkumaran Kumaraswamipillai

This is a good candidate to make it to the job interview discussions along the debate of checked Vs unchecked exceptions as the industry is split on this very topic. Some favor XML, whilst others love using the JavaConfig.

Q1. What are the Pros and Cons of Spring JavaConfig Vs. Spring XML?
A1.

Pros for XML:

1) Configuration is centralized in a fewer XML files, and not scattered among all different components @Inject,

@Component, @Configuration, @Repository, etc annotations. So with XML you can have a nice overview of beans and their wirings in a single place.

2) The XML is completely independent to the Java artifacts as there's no coupling between the 2. This enables the class to be used in more than one context with different configurations.

Cons for XML:

1) XML is verbose, so configuration files tend to get big. It's a good thing to split them up.

Pros for JavaConfig:

1) JavaConfig can be seen as the XML file equivalent, written in Java to give the power of compile-time checks, code completion, support for finding references in the workspace and refcatoring via the IDE support. The typos generate compile-error, so no more waiting till load time to find XML typos.

Cons for JavaConfig:

1) If not done properly can create annotation mess and confusion.

Q2. For the given code snippet created using the following steps, write how will you go about wiring them up **a)** XML way **b)** JavaConfig way:

Created a MVN project

```
mvn archetype:generate -DgroupId=com.mytutorial -
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.
    xsi:schemaLocation="http://maven.apache.org/
```

Core Java Modif
Differences betw
Java Collection i
⊞ Event Driven Progr
⊞ Exceptions (2)
⊞ Java 7 (2)
⊞ Java 8 (24)
⊞ JVM (6)
⊞ Reactive Programr
⊞ Swing & AWT (2)
⊞ JEE Interview Q&A (3
⊞ Pressed for time? Jav
⊞ SQL, XML, UML, JSC
⊞ Hadoop & BigData Int
⊞ Java Architecture Inte
⊞ Scala Interview Q&As
⊞ Spring, Hibernate, & I
⊞ Testing & Profiling/Sa
⊞ Other Interview Q&A f
⊞ ▶ Free Java Interviev

16 Technical Key Areas

open all | close all
⊞ Best Practice (6)
⊞ Coding (26)
⊞ Concurrency (6)
⊞ Design Concepts (7)
⊞ Design Patterns (11)
⊞ Exception Handling (3
⊞ Java Debugging (21)
⊞ Judging Experience I
⊞ Low Latency (7)
⊞ Memory Managemen
⊞ Performance (13)
⊞ QoS (8)
⊞ Scalability (4)
⊞ SDLC (6)
⊞ Security (13)

```
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.mytutorial</groupId>
    <artifactId>simple-spring</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>simple-spring</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</pro
    </properties>

    <dependencies>
        <!-- Spring Framework -->
        <dependency>
            <groupId>org.springframework</groupI
            <artifactId>spring-aspects</artifact
            <version> 3.2.13.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupI
            <artifactId>spring-context-support</
            <version> 3.2.13.RELEASE</version>
        </dependency>
    </dependencies>
</project>
```

Class with main method

```
package com.mytutorial;

import org.springframework.context.ApplicationCo
import org.springframework.context.ConfigurableA
import org.springframework.context.support.Class

public class App
{
    //injected via XML
    private SimpleService myService;

    public static void main( String[] args )
    {
        ApplicationContext ctx = new ClassPathXm
        App app = (App)ctx.getBean("springApp");
        app.executeApp();
        ((ConfigurableApplicationContext) ctx).c
    }

    public void executeApp() {
        System.out.println("Executing App...");
        myService.executeService();
    }

    //invoked by Spring XML setter
    public void setMyService(SimpleService mySer
        this.myService = myService;
    }
}
```

⊞ Transaction Managen

80+ step by step Java Tutorials

open all | close all
⊞ Setting up Tutorial (6)
⊞ Tutorial - Diagnosis (2
⊞ Akka Tutorial (9)
⊞ Core Java Tutorials (2
⊞ Hadoop & Spark Tuto
⊞ JEE Tutorials (19)
⊞ Scala Tutorials (1)
⊞ Spring & HIbernate Tu
⊞ Tools Tutorials (19)
⊞ Other Tutorials (45)

100+ Java pre-interview coding tests

open all | close all
⊞ Can you write code?
⊞ ♦ Complete the given
⊞ Converting from A to I
⊞ Designing your classe
⊞ Java Data Structures
⊞ Passing the unit tests
⊞ What is wrong with th
⊞ Writing Code Home A
⊞ Written Test Core Jav
⊞ Written Test JEE (1)

How good are your .....?

open all | close all
Career Making Know-
Job Hunting & Resum

Service layer

```
package com.mytutorial;

public interface SimpleService {
     void executeService();
}
```

```
package com.mytutorial;

public class SimpleServiceImpl implements Simple

    private SimpleDao myDao;

    public void executeService() {
       System.out.println("Running simple servic
       myDao.executeDao();
    }

    //gets called by the Spring XML to set Dao
    public void setMyDao(SimpleDao dao) {
        this.myDao = dao;
    }
}
```

DAO layer

```
package com.mytutorial;

public interface SimpleDao {
     void executeDao();
}
```

```
package com.mytutorial;

public class SimpleDaoImpl implements SimpleDao {

    public void executeDao() {
       System.out.println("Running simple dao imp
    }

}
```

A2. The XML config gives you a big picture of what is injected where. dao –> service –> springApp.

XML Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/sch
```

```
    xmlns:context="http://www.springframework.or
    xsi:schemaLocation="http://www.springframewo
        http://www.springframework.org/schema/be
        http://www.springframework.org/schema/co
        http://www.springframework.org/schema/co

    <bean id="springApp" class="com.mytutorial.A
        <property name="myService" ref="service"
    </bean>

    <bean id="service" class="com.mytutorial.Sim
        <property name="myDao" ref="dao" />
    </bean>

    <bean id="dao" class="com.mytutorial.SimpleD

</beans>
```

JavaConfig

```
package com.mytutorial;

import org.springframework.context.annotation.Be
import org.springframework.context.annotation.Co

@Configuration
public class SpringConfig {

    @Bean
    public SimpleDao myDao() {
        return new SimpleDaoImpl();
    }

    @Bean
    public SimpleService myService() {
        SimpleServiceImpl simpleService = new Si
        return simpleService;
    }

    @Bean
    public App app() {
        App app = new App();
        return app;
    }

}

```

The App.java will look like with **@Autowired** and **AnnotationConfigApplicationContext**

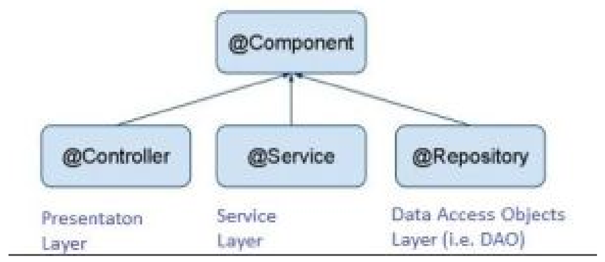
```
package com.mytutorial;

import org.springframework.beans.factory.annotat
import org.springframework.context.ConfigurableA
import org.springframework.context.annotation.An

public class App {

    @Autowired
    private SimpleService myService; //myService

    public static void main(String[] args) {
        AnnotationConfigApplicationContext ctx =
        App app = (App) ctx.getBean(App.class);
        app.executeApp();
        ((ConfigurableApplicationContext) ctx).c
    }

    public void executeApp() {
        System.out.println("Executing App...");
        myService.executeService();
    }

    public void setMyService(SimpleService mySer
        this.myService = myService;
    }
}


```

The **SimpleServiceImpl** requires the @Autowired

```
package com.mytutorial;

import org.springframework.beans.factory.annotat

public class SimpleServiceImpl implements Simple

    @Autowired
    private SimpleDao myDao; // myDao() is invok

    public void executeService() {
       System.out.println("Running simple servic
       myDao.executeDao();
    }

    //gets called by the Spring XML to set Dao
    public void setMyDao(SimpleDao dao) {
        this.myDao = dao;
    }
}
```

Use of auto wiring is not a best practice for very large projects. So, lets use the new Java **@Inject** annotation.

JavaConfig and annotations like @Inject, @component, @Repository, @Service, etc



Spring annotations for DI

Add new Java dependency in the pom.xml file to be able to use **@Inject** annotation.

```
<dependency>
       <groupId>javax.inject</groupId>
       <artifactId>javax.inject</artifactId>
        <version>1</version>
</dependency>

```

The **SpringConfig** only having the app(), most importantly “**@ComponentScan**(basePackages=”com.mytutorial”) ” to be able to scan for classes with the spring annotations like @Component in the base package “com.mytutorial”.

App.java with @Inject and @Component annotations

```
package com.mytutorial;

import javax.inject.Inject;

import org.springframework.context.ConfigurableA
import org.springframework.context.annotation.An
import org.springframework.stereotype.Component;

@Component
public class App {

    @Inject
    private SimpleService myService;

    public static void main(String[] args) {
        AnnotationConfigApplicationContext ctx =
        App app = (App) ctx.getBean(App.class);
        app.executeApp();
        ((ConfigurableApplicationContext) ctx).c
    }

    public void executeApp() {
        System.out.println("Executing App...");
        myService.executeService();
    }

    // invoked by Spring XML setter
    public void setMyService(SimpleService mySer
        this.myService = myService;
    }
}


```

SimpleServiceImpl.java with @Inject and @Service annotations

```
package com.mytutorial;

import javax.inject.Inject;

import org.springframework.stereotype.Service;

@Service
public class SimpleServiceImpl implements Simple

    @Inject
    private SimpleDao myDao;

    public void executeService() {
       System.out.println("Running simple servic
       myDao.executeDao();
    }

    //gets called by the Spring XML to set Dao
    public void setMyDao(SimpleDao dao) {
        this.myDao = dao;
    }
}
```

SimpleDaoImpl.java with the @Repository annotation

```
package com.mytutorial;

import org.springframework.stereotype.Repository

@Repository
public class SimpleDaoImpl implements SimpleDao

    public void executeDao() {
       System.out.println("Running simple dao im
    }

}
```

XML and annotations like @Inject, @component, @Repository, @Service, etc

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/sch
    xmlns:context="http://www.springframework.or
    xsi:schemaLocation="http://www.springframewo
        http://www.springframework.org/schema/be
        http://www.springframework.org/schema/co
        http://www.springframework.org/schema/co

    <context:component-scan base-package="com.my

</beans>
```

App.java

```
package com.mytutorial;

import javax.inject.Inject;

import org.springframework.context.ApplicationCo
import org.springframework.context.ConfigurableA
import org.springframework.context.support.Class
import org.springframework.stereotype.Component;

@Component
public class App {

    @Inject
    private SimpleService myService;

    public static void main(String[] args) {
        ApplicationContext ctx = new ClassPathXm
        App app = (App) ctx.getBean(App.class);
        app.executeApp();
        ((ConfigurableApplicationContext) ctx).c
    }

    public void executeApp() {
        System.out.println("Executing App...");
        myService.executeService();
    }

    // invoked by Spring XML setter
    public void setMyService(SimpleService mySer
        this.myService = myService;
    }
}
```

all the other implementation classes same as before.

Popular Posts

♦ 11 Spring boot interview questions & answers

824 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

765 views

18 Java scenarios based interview Questions and Answers

399 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

285 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts interview questions & answers

201 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ Part #3: JPA Hibernate Spring Maven Eclipse Tutorial – testing data access with HSQLDB

♥ Top 6 Spring wiring via JavaConfig [i.e. @Configuration] examples ›

Posted in Differences Between X & Y, member-paid

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer