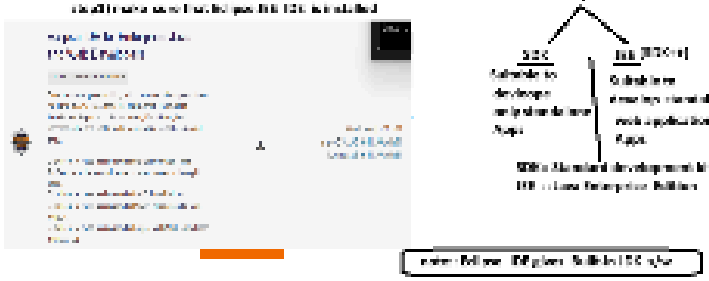
Spring First App Development (Example App on Setter Injection using Xml driven cfgs)



Spring

First App Developement

com.nt.sbeans.Wish MessageGenerator

(target spring bean class)

(Example App on Setter Injection using Xml driven cfgs)

java.time.LocalTime (Dependent spring bean)

(Using setter method Wish MessageGenerator we try to inject LocalTime object (having System time) to Wish MessageGenerator class obj and that injected object in the b.method to generate wish message based on current hour of the day)

Eclipse IDE

Procedure to the above App using Eclipse IDE

==========

step1) make sure that Eclipse JEE IDE is installed

Eclipse IDE for Enterprise Java and Web Developers

530 MB 283,887 DOWNLOADS

Tools for developers working with Java and Web applications, Including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web

Services, JPA and Data Tools, Maven and Gradle, Git, and more.

Click here to raise an Issue with the Eclipse Web Tools

Platform. Maintainers will move opened Issues to the right place.

Click here to raise an issue with the Eclipse Platform.

Click here to raise an issue with Maven integration for web projects.

Click here to raise an issue with Eclipse Wild Web Developer (incubating).

Enter ch

Windows x86 64

macOS x86_64 | AArch64

Linux x86 64 | AArch64

SDK Suitable to devleope

only standalone Apps

JEE (SDK++)

Suitable to

develop standalone apps

web applications, Distributed Apps

SDK:: Standard development kit JEE :: Java Enterprise Edition

note:: Eclipse IDE gives Built-in JDK s/w

Eclipse JEE Latest version :: 2024-09

=> Extract the downloaded zip file for installation and use ecilipse.exe file

for launching the IDE (prefer creating destktop ICON)

step2) keep the following software setup ready

=>Eclipse JEE IDE latest version

=> jdk 17+

=

step3) Launch Eclipse JEE IDE by choosing the workspace folder

In One computer we can install multiple Eclipse IDEs

|---> The folder where our Projects will be saved (E:/workspaces/spring/NTSPBMS515)

step4) Download the spring core module related main and dependent jar files (libraries)

spring-core-6.2.3.jar

spring-aop-6.2.3.jar

spring-jcl-6.2.3.jar

(main jar file)

Dependent jar files

spring-context-support-6.2.3.jar (main jar file)

spring-expression-6.2.3.jar

go to mvnrepository.com ---> search for each jar file with it name -->select jar file name--->select latest vesion--> click jar link --> to start the downloading process mvnrepository.com/search?q=spring+context+support

spring-context-6.2.3.jar

spring-beans-6.2, 3.jar

Depedent jar files

SITORY

spring context support

if java app is using otner tnan jak apis (jee apis, third party apis,user-defined apis, f/w apis)and etc.. then those apis related jar files must be added to CLASSPATH (for normal apps) or build path (For eclipse Apps)

step5) create java Project using Eclipse IDE

DependencyManagement

File menu ---> new ---> Java Project ---> name :: IOCProj01 ---> -->next ---> next --->finish

== ==SS"

IOCProj01-DependencyInjection

> JRE System Library [JavaSE-21]

#src

step6) add Spring core module libraries to Project Build Path

Right click on Project ---> build path---> configure build path ---> libraries tab -->classpath ----> add external jar files ---> browse and select the above jar files

---> ....

Referenced Libraries

spring-context-6.2.3.jar - C:\Users\Nataraz\Downloads >spring-expression-6.2.3.jar - C:\Users\Nataraz\Download

> spring-core-6.2.3.jar - C:\Users\Nataraz\Downloads

> spring-aop-6.2.3.jar - C:\Users\Nataraz\Downloads

>

spring-jcl-6.2.3.jar - C:\Users\Nataraz\Downloads spring-context-support-6.2.3.jar - C:\Users\Nataraz\Dow
spring-beans-6.2.3 .jar - C:\Users\Nataraz\Downloads

ds

step7) create the packages in "src" folder of the Project

#src

com.nt.cfgs (To place spring bean cfg file)

com.nt.main (main class)

com.nt.sbeans (user-defined spring bean classes)

step8) Develop the source code of the application

//WishMessageGenerator.java (Target spring bean class) package com.nt.sbeans;

import java.time.LocalTime;

public class Wish MessageGenerator {

private LocalTime time;

//setter method for setter Injection

public void setTime(LocalTime time) {

this.time=time;

}

}

//b.method

public String showWishMessage(String user) {

// get current hour of the day

int hour-time.getHour();

//generate wish message based on current hour of the day

if(hour<12)

return "Good Morning::"+user;

else if(hour<16)

else if(hour<20)

else

return "Good Afternoon:"+user;

return "Good Evening :"+user;

return "Good Night:"+user;

=>Xml schema namespace is a library that contains set of readily available xml tags .. if u want

to use the xml tags of certain namespaces in the xml file development, we must import those

namespaces by specifying the their uris/urls on top the of xml files

=> XSD file (Xml Schema Definitation file) is maintains set of xml namespaces.. spring gives nearly 15+ xml

namespaces defined in multiple xsd files (XmlSchemaDefinitation file)

=> XSD files are like jar files representing apis/libraries

=> Xml namespaces are like java packages of those libraries

=> import namespaces by specifying their urls/uris is like importing the java packages to the java app

xml namespace

beans

context

mvc

namespace url

http://www.springframework.org/schema/beans http://www.springframework.org/schema/context http://www.springframework.org/schema/mvc

xsd filename

spring-beans.xsd

spring-context.xsd spring-mvc.xsd

applicationContext.xml

<?xml version="1.0" encoding="UTF-8"?> (xml prolog) <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">

<!-- Dependent spring bean cfg -->

<!-- calls LocalTime.now() method to create LocalTime class obj having system time -->

Organization HomePage

Spring IO

https://github.com/spring-projects/spring-fra

Date

Feb 13, 2025

Files

pom (2 KB)

jar (171 KB) View All

Importing xml namespaces

and schema definitations

not

(Do type this,collect from internet)

<bean id="ltime" class="java.time.LocalTime" factory-method="now"/>

|--static factory method for spring bean instantiation

Go to google.com ---> search for sample spring bean cfg file --> open the website ---> search for

<!-- Target spring bean class cfg -->

spring-beans ---> collect the header content

and paste in applicationContext.xml file

<bean id="wmg" class="com.nt.sbeans.WishMessageGenerator">

<property name="time" ref="ltime"/>

→

C

docs.spring.io/spring-framework/docs/4.2.x/spring-framework-reference/html/xsd-configuration.html

<!-- setter injection to inject Local iine obj(ltime) to time property of WishMessageGenerator class obj(wmg) --> </bean> Dependent spring bean class id (obj) to inject

</beans>

</beans>

HAS-A property

name in target spring

bean class

Client App (main class of the standalone app)

//DependencyInjectionTest.java

package com.nt.main;

import org.springframework.context.support.FileSystemXmlApplicationContext;

import com.nt.sbeans.WishMessageGenerator;

public class DependencyInjection Test {

public static void main(String[] args) {

//crete IOC Container

FileSystemXmlApplicationContext ctx=

new FileSystemXmlApplicationContext("src/com/nt/cfgs/applicationContext.xml");

//get Target Spring bean class obj ref

WishMessageGenerator generator=ctx.getBean("wmg",WishMessageGenerator.class); //invoke the b.method

String result-generator.showWishMessage("raja"); System.out.println("Result is ::"+result);

//close the IOC cotnainer

ctx.close();

}

}

The equivalent file in the XML Schema-style would be...

<?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.xs

<!-- bean definitions here -->

=>if the xml document (file) is satisfying basic syntax rules then it is called well-formed XML document

-> basis syntax rules are

-> every open tag should have closing tag

-> all the content must be placed in b/w root open tag and root closing tag

-> xml tag names, attribute names are case-sensitive

-> attribute values must be kept in quotes

and etc..

=>if the xml document (file) is satisfying imported schema name space rules and guidelines then it is called Valid XML document

</beans>

=> To check whether xml doc is well-formed or not, valid or not and to read, process the XML document we have XML Parser given by XML APIs

eg:: SAX Parser given by SAX API

eg:: DOM Parser given by DOM API eg:DOM4JParser given byDOM4API eg:: JDOM Parser given by JDOM API and etc...

SAX :: Simple API for Xml Processing

DOM :: Document Object Model

DOM4J:: DOM for Java

JDOM: Java DOM

=>IOC container gives SAX API as the built-in API having SAX Parser

=> if we given XML document to XML Parser of XML API, it will perform the following operations

a) Loads the XML file

b) checks weather XML doc is well-formed or not (if not throws Exception)

c) checks weather XML doc is valid or not (if not throws Exception)

d) reads the XML document content and creates InMemory Metadata in the JVM Memory of the RAM where the IOC container / Java App is running to access and use XML content for multiple times fastly

Machine1

MetaData :: Data About Data

=> MetaData of Xml file(spring bean cfg file) is nothing but gathering more info about spring bean classes, xml tags, attributes and etc...

RAM (RandomAcess Memory)

JVM Memory

Proj1 Memory

OC container InMemory MetaData

of applicationContext.xml

HDD (HardDisk Drive)

proj1

|--> applicationContext.xml

JVM :: Java Virtual Machine

What is the difference b/w pool and cache?

Pool

[Gives reusability of same items]

To implement this

we can use List collection or Array

Cache/Buffer (12)

[Gives reusability of different items]

=>After creating spring bean class objs they will be maintained in the IOC container in the internal cache of IOC container having bean ids as the keys and spring bean class obj refs as the values

wmg ltime

...

Internal cache of IOC container WishMessageGenrator class obj ref LocalTime class obj ref

....

To implement this we can use map collection [key -values pairs -- values as the items and each value is identified by the key]

bean ids(keys) bean class obj refs

Assignment:: Inject LocalDate class obj to Season Finder class obj and to find current season name

based on current month of the year