Depedency Managent/IOC

Dependency lookup
(Target searches and gets Dependent class object)

Dependency injection (Here underlying server/container/ framework/IRE/JVM Dynamically assigngs dependent class obj to target class object)
->setter injection
->constructor injection
->Interface injection/aware injection
-> Lookup Method Injection
-> Method Injection /Method Replacer

setter injection
==========

If the underlying Spring Container/IOC container uses setter method of target class to inject/assign dependent class object to target class object then it called setter injection.

WishMessageGenerator (target class) ----------> java.util.Date (dependet class)

gets current of the day from the system date given by the Dependent java.util.Date class object and generates wish messages like good morning, good evening and etc..

Resourcees to develope
==========

Any java class can be taken as target/ dependent spring bean class..

IOCProj3-SetterInjection
|-->src
|--com.nt.beans
|-->WishMessageGenerator.java (taget class)
|-->com.nt.cfgs
|-->applicationContext.xml (spring Bean cfg file)
|--com.nt.test
|-->SetterInjectionTest.java (Client App)

jar files in classpath/built path :: same as first App

=> File -->new -->project --> Java Project -> name:: IOCProj3-SetterInjection
=> create the above packages in "src" folder "com.nt.beans" ,"com.nt.cfgs" ,"com.nt.test"
=>Add the jar files :: Right click on the project --> build path -->cfg buildpath -->libraries tab -->classpath --> add external jars -->select jars..

(download from internet)
from<Spring_home>\libs folder)

STS Plugin (Eclipse Plugin)
|-->Spring Tool Suite
|-->Makes spring App development in eclipse very easy
|-->provides base to develp spring boot apps..

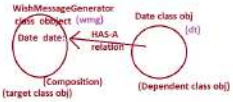
STS -->seperate IDE (not popular)

[plugin is patch s/w that provides addtional functionalaties to the existing s/w to enhance the facilities of the existing s/w]

To install STS in eclipse IDE

->Help menu --> Eclipse market place --> search for STS -->go --> select STS 3.x version (not 4) -->install --> accept terms and conditions --> restart IDE -->finish.

We make taget class getting Dependent class obj by designing those classes through Composition(HAS-A Relationship).



note:: WishMessageGenerator class will have java.util.Date class reference variable to hold java.util.Date class object given Spring/Container , this is nothing but composition (HAS-A relationship)

WishMesageGenerator.java

```
public class WishMessageGenerator{
  private Date date;    // bean property

  public void setDate(Date date){
    this.date=date;
  }

  ....
  ....
  ....
}
```

setter method supporting setter injection.

To make two java classes as target and dependent spring beans of spring App supporting setter injection and constructor injection then we design them based composition (HAS- A) relationship.

ctrl+shift+ O :: To import the package
alt+shift+s, r :: To generate getter and setter methods

In spring bean cfg file
<property> tag indicates setter injection on the given property
<constructor-arg> indicates constructor injection

Code

//WishMessageGeneator.java (Target class)

```
package com.nt.beans;

import java.util.Date;

public class WishMessageGenerator {
      //bean property/attribute
 private Date date;   //after injection it holds sys date and time

  static {
       System.out.println("WishMessageGenerator:: static block");
  }
  public WishMessageGenerator() {
       System.out.println("WishMessageGenerator:: 0-param constructor");
  }

   //setter method for setter injection
  public void setDate(Date date) {
       System.out.println("WishMessageGenerator.setDate(-)");
       this.date = date;
  }

  // b.method
  public  String generateWishMessage(String user) {
       System.out.println("WishMessageGenerator.generateWishMessage()");
       //get hour of the day
       int hour=date.getHours(); // 24 hours format
       //generate wish Message
        if(hour<12)
             return "Good Morning::"+user;
       else if(hour<16)
             return "Good AfterNoon::"+user;
       else if(hour<20)
             return "Good Evening::"+user;
       else
             return "Good Night::"+user;
  }

  }
```

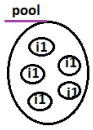
applicationContext.xml

```
<?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 https://www.springframework.org/schema/beans/spring-beans-4.3.xsd">
  <!-- cfg  java.util.Date class as spring bean (dependent calss) -->
  <bean id="dt" class="java.util.Date"/>

   <!-- cfg WishMessageGeneator as Spring bean (target class) -->
   <bean id="wmg"  class="com.nt.beans.WishMessageGenerator">
     <property name="date" ref="dt"/> <!-- performs setter injection of "date" property with "dt" bean id reffered java.util.Date class spring bean obj-->
   </bean>

</beans>
```

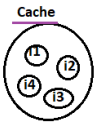
SetterInjectionTest.java

```
package com.nt.test;

import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.FileSystemResource;

import com.nt.beans.WishMessageGenerator;

public class SetterInjectionTest {

      public static void main(String[] args) {
           // Hold the name and location of spring bean file
        FileSystemResource res=new FileSystemResource("src/com/nt/cfgs/applicationContext.xml");
           //create IOC container
           XmlBeanFactory factory=new XmlBeanFactory(res);
           //get target Spring bean class object
           WishMessageGenerator generator=(WishMessageGenerator) factory.getBean("wmg");
           //invoke the b.method
           System.out.println("result="+generator.generateWishMessage("raja"));
      }//main
}//class
```

What is the difference between pool and cache?



Pool gives reusablitiy of same items (To implement pool take the support of ListCollection)



Cache gives reusability of different items

(To implement cache , take the support of Map Collection)

keys	values
a	i1
b	i2
c	i3
d	i4

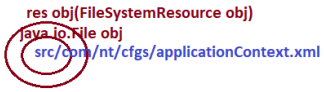
(map collection)

Flow of execution (The Standalone App flow begins with main(-) method and ends with main(-) method)

a) Programmer runs the Applicaiton through client App
b) main(-) method client Apps executes
c) `FileSystemResource res=new FileSystemResource("src/com/nt/cfgs/applicationContext.xml");`

relative path

=>FileSystemResource obj internally uses java.io.File class object to hold the name and location given spring bean cfg file and Later it helps IOC Container /Spring Container (BeanFactory Container) to locate the spring bean cfg file from the specified path of System dirves (FileSyystem)

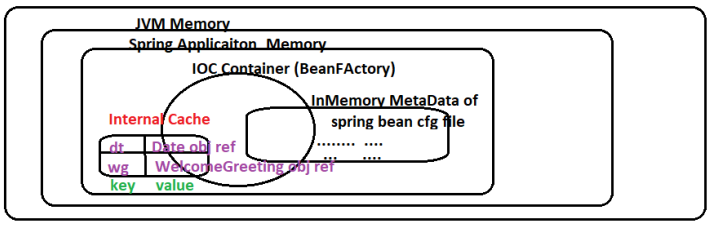


note:Here we can give either Absoute path or relative path

d) `XmlBeanFactory factory=new XmlBeanFactory(res)`

i) Based on given res obj (FileSystemResoruce obj) , it locates and loads given spring bean file from the specified path of the file system . (if not avaiable throws java.io.FileNotFoundException)
ii) Checks wheather spring cfg file is well-formed or not , valid or not .. if not Xml Parsing Exception will be thrown..
ii) Creates InMemory Medata of spring bean cfg file in the JVM Memory of RAM where the application is executing
iv) creates IOC/SpringContainer of type BeanFactory having this InMemoryMetaData of Spring cfg file
v) returns XmlBeanFactory object representing the just created IOC/Spring container

RAM of the computer

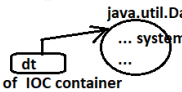


e) `Date d=(Date)factory.getBean("dt");`

(i) factory.getBean("dt") method takes the given spring bean id "dt" searches for Spring Bean class obj in the internalCache of IOC container , Since not available then it goes to InMemory MetaData of spring cfg file to search for Spring bean class cfg having bean id "dt" and finds "java.util.Date" class as spring bean class.

ii) IOC Container/Spring Container loads the spring bean class (java.util.Date) and creates the object by using 0-param constructor with the support of newInstance() method

=>Class c=Class.forName("java.util.Date"); //Loads the class
=> java.util.Date dt=(java.util.Date) c.newInstance(); //creates the object loaded class



iii) keeps the SpringBean class obj ref in the internal cache of IOC container having bean id as the the key(dt) and Spring bean class obj ref the value (Data class obj ref) for resuablity of spring class objs

ref

iv) returns java.util.Date class obj back to Client App as java.lang.Object class ref .. and u r type casting with java.util.Date class

f) `System.out.println(" d obj data ::"+d);`

=>Internall calls d.toString() method and displays the available System date and time.. (becoz IOC container is creating Date claass object usisng 0-param constructor and this process holds sys date and time as date of the java.util.Date class obj)

F3:: To get The source code..

g) `WelcomeGreetings greetings=(WelcomeGreetings)factory.getBean("wg");` (Similar to previous factory.getBean("dt")

i)

ii)

iii)

h) `System.out.println("message::"+greetings.welcome("raja"));`

On the recieved WelcomeGreetings class obj ref.. we invoking b.method (welcome(-)) to execute the b.logic

i) End of main(-) method .. all objects will be destroyed like spring bean objs including factory object that reprsents IOC container .. So IOC container its InMemeory metadata, Internal cache and everying will be vanished.. at the end of main(-)
j) JVM memory of spring App will be vanished and that is end of the Applicaiton.

Limitation with "new" operator
==========

Test t=new Test();
"new" operator creates the object of java class at runtime.. but expects the presence of java class from compile time onwords. i.e we can not use new operator to create the object of java class at runtime whose class name is coming to the app from compiletime onwords.

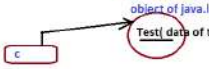
Alternate is newInstance() method (deprecated from java.9) of java.lang.Class or newInstance(varags..) (best) method of java.lang.reflect.Constructor

note:: ServletContainer gets Servlet comp class name from web.xml file dynamically at runtime
So it can not use "new" operator to create object of servlet comp class.. i.e it internally uses newInstance() methods.
note:: Spring/IOC container gets spring bean class names from spring bean file dynamically at runtime,
So it can not use "new" operator to create object of spring bean class.. i.e it internally uses newInstance() methods.

java.lang.Class
|--->newInstance() (deprecated from java .9.)
(Can create object only by using 0-param constructor)
java.lang.reflect.Constructor
|--->newInstance(vararg ...) (not deprecated)
(can create object by using 0- param or more param constructor)

Sample code Using newInstance() method java.lang.Class

```
//Load java class
Class c=Class.forName("Test");   //Class.forName(-) method loads given java class dynamically
                                 //at runtime and returns the object lang.lang.Class having
                                 //the loaded class name as the data of the object.
//create the object for loaded class
Test t=(Test)c.newInstance();  //creating object for Test class
```




```
int a=10;
a=a+10;
```

The object of java.lang.Class can hold class name/interface name/annotation name/ enum name in the Java App as the data of object.

alft+shift+s+s :: To get toString()
sout/sysout + ctrl+space :: S.o.p(-)
systrace +ctrl+space :: gives S.o.p(-) with message
F3:: To get Source code.

Example App using on newInstance() methods of java.lang.Class and java.lang.reflect.Constructor class
==========

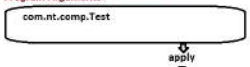
```
package com.nt.comp;

public class Test {

    private int a, b;

    static {
        System.out.println("Test:static block()");
    }

    public Test() {
        System.out.println("Test:: 0-param constructor");
    }

    public Test(int a,int b) {
        System.out.println("Test::2-param constructor");
        this.a=a;
        this.b=b;
    }

    @Override
    public String toString() {
        return "Test [a=" + a + ", b=" + b + "]";
    }
}
```

NewInstanceTest.java

```
package com.nt.test;

public class NewInstanceTest1 {

    public static void main(String[] args) throws Exception{
        //Load class
        Class c=Class.forName(args[0]);
        //create object
        Object obj=c.newInstance();
        System.out.println("data ::"+obj);
    }
}
```

To run this App from Eclipse passing cmd line args

Right click main(-) method class --->run as ---->
run configurations ---> arguments tag --->
Program Arguments

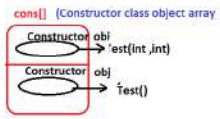


apply
run

NewInstnaceTest2.java

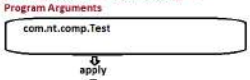
```
package com.nt.test;

import java.lang.reflect.Constructor;

public class NewInstanceTest2 {

    public static void main(String[] args) throws Exception{
        //Load class
        Class c=Class.forName(args[0]);
        //get all declared constructor of Loaded class
    Constructor cons[]=c.getDeclaredConstructors();
    //create object using 0-param constructor
     Object obj1=cons[1].newInstance();
     System.out.println("obj1 data::"+obj1);
     System.out.println("----------------------");
     //create object using 0-param constructor
     Object obj2=cons[0].newInstance(10,20);
     System.out.println("obj2 data::"+obj2);
     }//main
}//class
```

cons[] (Constructor class object array



To run this App from Eclipse passing cmd line args

Right click main(-) method class --->run as ---->
run configurations ---> arguments tag --->
Program Arguments



apply
run

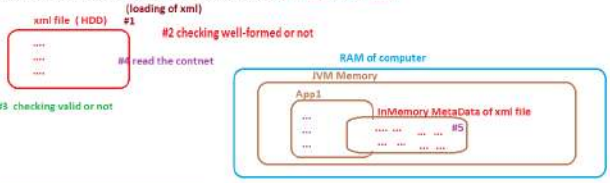
note:: Spring/IOC container internally uses newInstance() method to create object of our spring bean calsses by collecting spring bean clsses name from spring cfg file dynamically at runtime..

=> if the xml document/file that is satisfying xml syntax rules then it is called well-formed Xml document...
the syntax rules
-->tags /attributes are case-sensitve
--> All tags must be nested properly
--> all attributes must be quoted.
--> All tags must be closed
and etc..
=> if the xml document/file that is satisfying imported XSD/DTD rules then it is called valid Xml document...

=> Xml parser is a software program that can load xml file , can check well-formness and validness of xml file and read and process xml document/file

Examples of xml parsers ::
SAX Parser (Simple API Xml parser)
DOM parser (Document Object Model Parser)
JDOM Parser (Java DOM Parser)
DOM4J Parser (DOM for Java Parser)
and etc..

=> After Loading xml , checking wellformness and validness of xml file, the Xml parsers reads the xml file content and prepares InMemory MetaData of that xml file in the memory Where the current Java App runs (JVM Memory of RAM)



MetaData :: data about data (more info)

All containers will have built-in Xml parser.

What is a Container?

=>Container is a software program that manages the life cycle of given comp birth to death (Object creation to object destruction)

=> It is like an aquirium managing the life cycle of fishes..

=>Servlet container manages the life cycle servlet comps

=>jsp container manages the life cycle jsp comps

=> SpringContainers/IOC container manages the life cycle of spring beans.
|-->BeanFactory (Basic)
|-->ApplicationContext (advanced) (Best)

IOC ::Inversion of Control

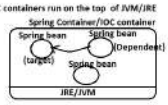
Servlet ,jsp containers are heavy are weight becoz they need the heavy weight webSErver or Application server softwares..
Spring/ IOC containers are not alternate for servlet,jsp containers

=>Spring Containers are light weight .. i.e by creating object for one pre-defined class we can create IOC container/spring container

To create BeanFactory container to create object for the class that implements org.springframework.beans.factory.BeanFactory(I)
-->XmlBeanFactory (impl class) (deprecated from spring 3.x)
-->DefaultListableBeanFactory(impl class) (Recomanded to use)

Spring/IOC containers are given for spring bean life cycle management and for Dependency Management (Arranging Dependent class object to Taget class object)

=>Spring /IOC containers run on the top of JVM/JRE



=> The java class whose obj is created and managed by spring container is called spring bean.

4 approaches of spring App development
a) Using xml driven configurations
b) Using Annotation driven configurations
c) 100% Code driven configurations
d) Using Spring Boot

=>While Developing spring application using Xml driven configurations we need to give inputs /instructions to SpringContainer/IOC container by spring bean cfg file (xml file)

=> any <file-name>.xml can be taken as spring bean cfg file.. but the recomanded name is applicationContext.xml
=> we need to pass spring bean cfg file as the input value while creating IOC containers /Spring Containers.
=> Generally Spring Bean cfg file contains the following details
a) configuring java classes as spring beans having bean ids (object names)
b) Mode of Dependency Management configurations
and etc..

making underlying container recognizing certain class/inteface/code or file with certain purpose is called confuguration....
eg:: By using <bean> tag in Spring bean file we try to make spring container to recognize given java class as spring bean .
eg:: By using <servlet>,<servlet-mapping> tags web.xml we make servlet container recognizing certain java class servlet comp class.

applicationContex.xml (com/nt/cfgs)

```
<beans ....>
  <bean id="dt" class="java.util.Date"/>
        bean id (or)   fully qualified java class to take as spring bean
</beans>  obj name
```

=> when IOC container creates spring bean class object the bean id becomes object name (refer variable names pointing to spring bean class obj)

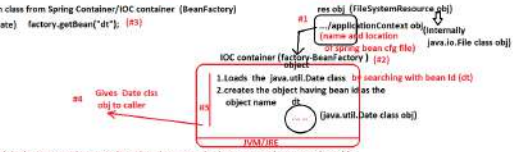
To create BeanFactory IOC container

```
//Locate and hold SpringBean cfg file
#1  FileSystemResource res=new FileSystemResource("...../applicationContext.xml");
```

Internally uses java.io.File to locate and hold given spring cfg file from the specified path of the computer drive (File system)

=>All files and folders of a computer belonging to different drives together is called filesystem

```
//create BeanFactory IOC container
  XmlBeanFactory factory=new XmlBeanFactory(res);  (#2)

// get Spring Bean class from Spring Container/IOC container (BeanFactory)
  #4  Date d1=(Date)  factory.getBean("dt");  (#3)
```



note1:: Bean Ids with in the IOC container must be unique i.e Two spring bean can not have same bean ids.
note2: BeanFactory Container perform lazy instantation of spring beans i.e until we call factory.getBean(-) method It will not attempt to create spring bean class objects.

Procedure to develop first Spring Application showing IOC container creation and spring bean management?

step1) Download and install Eclipse JEE Ide (either 2020-06 or 2020-09)
download as zip file from thios url ::https://www.eclipse.org/downloads/packages/release/2020-06/r and extract the zip file .
eclipse-jee-2020-06-R-win32-x86_64.zip

step2) keep the s/w setup ready in u r computer
jdk 1.8+ (java 13)
spring 5.3.x (5.3.1) (as zip file from https://repo.spring.io/release/org/springframework/spring/5.3.1/) (select ...dist.zip file)
Eclipse 4.x (4.17 - 2020-06) (as zip file from https://www.eclipse.org/downloads/packages/release/2020-06/r)

step3) Launch Eclipse IDE by choosing workspace folder
G:/workspaces/spring/ntsp613 (the foder where eclipse proects will be saved)
use eclipse.exe file from zip file extraction to start eclipse

step4) create Java Project in Eclipse IDE having name IOCProj1-SpringBeanBasics
File menu --> new --> project --> java project --> name:: IOCProj1-springBeanBasics --> next --> say no to modules --> finish

step5) add spring core module libraries to the project
Right click project --> buildpath --> configure build path --> Librbraries tab-->select classpath --> add external jars --> select following jar files

from <spring_home>\libs folder

(from Internet as supporting jar file doing logging operation)
collect from ::https://mvnrepository.com/artifact/commons-logging/commons-logging/1.2)

Step6) Add packages to src folder and source code/files to those packages.

```
IOCProj1-SpringBeanBasics
  |--src
     |--com.nt.beans
        |--> WelcomeGreetings.java (User-defined Spring bean class)
     |-->com.nt.cfgs
        |-->applicationContext.xml (Spring bean cfg file)
     |--com.nt.test
        |-->BasicTest.java (Client App)
```

=>Spring Bean is a any java class provided by vendor but it must be configured in spring bean cfg file using <bean> tag.

sysout/sout :: To get S.o.println()
systrace :: To get S.o.p(-) with message

```
// WelcomeGreetings.java ( User-defined java class as spring bean)
package com.nt.beans;
public class WelcomeGreetings {
    static {
        System.out.println("WelcomeGreetings.static block");
    }

    public WelcomeGreetings() {
        System.out.println("WelcomeGreetings: 0-param constructor");
    }

    public String welcome(String user) {
        return "welcome to String:: "+user;
    }
}
```

=>To develop xml file by suing vendor supplied rules and guidelines then we need import that vendor supplied DTD/XSD Rules in the xml file
DTD :: DocumentTypeDefinitation (old -- slowly outdated)
XSD :: Xml Schema Definitation (new and recomanded)

=> In XSD all rules and guidelines are available in the form of XSD namespaces (it is like java package) and every name space is identified with is URL /URI.. In order any XSD namespace rules and guidline in our xml file.. we need to import XSD namespace by specifying their name space URI/URL

Spring framework has supplied 10+ XSD namespaces

name space	name space url	xsd file
beans	http://www.springframework.org/schema/beans	spring-beans-<ver>.xsd
c	http://www.springframework.org/schema/c	spring-beans-<ver>.xsd
p	http://www.springframework.org/schema/p	spring-beans-<ver>.xsd
context	http://www.springframework.org/schema/context	spring-context-<ver>.xsd
aop	http://www.springframework.org/schema/aop	spring-aop-<ver>.xsd
..	.	.
.	.	.
.	.	.

applicationContext.xml

copy from <spring_home>\docs\reference\pdf\core.pdf by searching for spring-beans

```
<?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
https://www.springframework.org/schema/beans/spring-beans.xsd">

  <!-- cfg  java.util.Date class as spring bean -->
  <bean id="dt"  class="java.util.Date"/>

  <!-- cfg  WelcomeGreetings class as spring bean -->
  <bean id="wg"  class="com.nt.beans.WelcomeGreetings"/>

</beans>
```

xml prolog

Importing "beans" xsd namespace by specifying its namespace uri and xsd file

Configuring java classes as spring beans

XSD
|-->xsd namespaces (1 or more)
|-->namespace is identified with its uri ,
|-->namespace is available is .xsd file

SpringBasicsTest.java (Client App)

```
package com.nt.test;
import java.util.Date;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.FileSystemResource;

import com.nt.beans.WelcomeGreetings;

public class SpringBasicsTest {

    public static void main(String[] args) {
        //Locate and hold spring bean cfg file
        FileSystemResource res=new FileSystemResource("src/com/nt/cfgs/applicationContext.xml");
        //Create IOC container (XmlBeanFactory)
        XmlBeanFactory factory=new XmlBeanFactory(res);
        //get Spring bean class objects from SpringContainer/IOC container
        Date d=(Date)factory.getBean("dt");
        System.out.println(" d obj data ::"+d);
        System.out.println("-----------------------");
        WelcomeGreetings greetings=(WelcomeGreetings)factory.getBean("wg");
        System.out.println("message:: "+greetings.welcome("raja"));
    }//main
}//class
```

<name>-<uniqueid>

step7) Run the client Application..
=>use ctrl+f11 from Client app (or)
=>use right click in client App -->run as --> java App.

Bean class /Component class
==========

=>The java class that is having both state and behaviour and state is used inside the behaviour is called Bean class /component class..

```
public class BankService{
  private  String branchLocation;
  private String bankName;
  private String ifscCode;
  private  long locationPin;

  public   String openAccount(CustomerDetails details){
     ... //IfscCode and branchLocation and bankName Details will be utilized
     ...
  }

  public   String transferMoney(int srcAcno, int destAcno , float amt){
     ... //transfermoney logic.  here also ifsc code, bankName,brachLocation
     ..    details will be used.
  }

}
```

State is given

Component/Bean class generally maintain lots of reusable logics.. in form of methods..

behaviour (methods)

=>Component class can be a POJO class or non-POJO class

=>Component/Bean class can not be Java Bean class.

SpringBean cnfig
==========

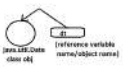
=>The Java class whose object is created and managed by spring container is called spring bean ..
=>Spring Bean class object life cycle should taken care by Springcontainer i.e right from Object creation to Object destruction the spring Container should take care of every thing.
=>Spring Bean can be a user-defined class or pre-defined class or third party supplied class.
=> Spring bean can be a Java Bean class or Bean /component class.
=>Spring bean can be a POJO class or non-POJO class.
=>Spring bean can be a invasive class or non-invasive class.

=> Every Java class should be cfg in spring bean cfg file (xml file) using <bean> tag.

eg1:: <bean id="dt" class="java.util.Date"/> (id = object name; class = class name)

Spring container creates object for given Spring bean class having given bean id as the object name

In xml env...



eg2:: <bean id="bankService" class=" com.rt.proj1.BankService"/> (bean id)

Annotation env..

```
@Component("bankService")
public class  BankService{
  ...
  ...
  ...
}
```

bean id

=> When spring container cretes the object for spring bean class the object name will be bean id. and spring bean is identified with its bean id.

=>every Java bean class is POJO class? (yes)
=>every bean /component class is POJO class? (may be or may not be)
=> Java bean class can be a spring bean ?(yes)
=> Component class can be a spring bean ?(yes)
=> Abstract class/Interface can be taken as spring bean?(Directly not possible)
=> Java class with setters,getters and b.methods is called what ?
    a) Java bean class   b) Bean /component class   (Answer is (a))
=>POJO class should have getters and setters? (no)
=>Spring bean must have setters && getters ? (no)
=> Any class can be taken as spring bean ? (yes)
=>By default every Java class is component class ?(no)

=>POJO class is Java class with out any specialities
=>POJO is a interface with out any specialities
=>Java Bean is a Java class with standards (mainly getters ,setter methods)
=>Component/Bean class is Java class that is having both state and behaviour and state is used in the b.logic of the behaviour (methods)
=>Any class Java class whose object is created and managed by Spring container is called spring bean.

Spring Core
==========

=>Base module for other modules
=> It is standard module in spring application development.. (This module is required as minimum module)
=> If this module is used alone to develop spring Apps.. we can develop only standalone Apps.
=> This module is given mainly for two operations
   a) To supply spring containers /IOC containers
      i)BeanFactory Container (basic)
      ii) ApplicationContext Container (advanced)

IOC: Inversion Of Control

DependencyManagement is also called as IOC (Inversion of Control) . Since SpringContainer are capable of performing the Dependency Management ,So they are also called IOC containers..

   b) spring Bean life cycle management and Dependency Management (IOC)
      (-> Assinging Dependent spring bean class object to Target Spring bean class object.

Taking care of all activities on spring bean class from birth to death (Object creation to object destruction).

Target class :: The Main class that uses other class services
dependent class :: The helper class used by main class..

There will be Composition(HAS-A relation) between target class and dependent i.e Target class will maintain the object dependent class.

Target class	Dependent class
Student	<-------> Course Material
Flipkart	DTDC
Employee	Department
Vehicle	Engine
Student	Faculty
Student	LibraryMembership
Person	Passport

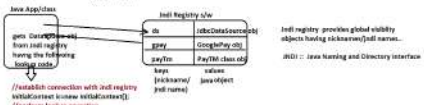
The target/main class needes Dependent class So The target class obj maintain dependent class objet.

Two types Dependency Management
   a) Dependency Lookup
   b) Dependency Injection

Dependency Lookup
==========

Here Target /main class writes logics and spends times to search and get Dependent/helper class object.

=>Here Target class pulls dependent class object from different resources..
eg1:: Student getting coursematerial by asking/requesting for it from a organization..
(target) (dependent)
eg2:: Java App /Class getting jdbc DataSource object from jndi registry through jndi lookup operation
(target) (Dependent)




```
//establish connection with jndi registry
InitialContext ic=new InitialContext();
//perform lookup operation
DataSource ds=(DataSource)ic.lookup("ds");
```

Limitation of DL (Dependency lookup) ::
   =>Here target class should spend some time and logics to search and get dependent class object.

advantage::
   -> Here target class will get only required Dependent class object.

b) Dependency Injection

HEre the underlying server /container/framework/ JVM/ any Runtime env.. dynamically assigns/injects Dependent class object target class object.. In this process it only creates both target and Dependent class objects if needed.

   eg1:: Student getting course material the moment he joins for the course
   eg2:: Servlet container assinging ServletConfig object to Servlet class object the moment it creates the Servlet class object
   eg3:: JVM assinging default values to object once the object is created..

Limitation::

   -> Here the underlying server/container/JVM/... etc.. may inject both neccessary and unnecessary values

advantage:
   ->Target class can use dependent directly with out searching for it.
   -> It is faster to get and use Dependents.

Spring support both Dependendency Lookup and dependency injection. (spring uses more of this)

Dec 5th and 6th::
Junit 5, mockito5 and HttpUnit
dec 5th :: 5pm
dec 6th : 10 am

Spring is giving support for Dependency injection in 5 modes
   ->a) Setter injection
   ->b) Constructor injection
   ->c) Aware injection/Interface injection/ Contextual Dependency Injection
   ->d) Lookup Method Injection
   ->e) Method Injection/Method Replacer

Def of Spring ?

Spring an Open Source , light weight , looPY coupled , Aspect oriented , Easily integratable,Dependency Managemenent based ,Non-Invasive Application framework given by Interface 21 (pivotal team) to develop different types of Java Applications by having abstraction layer on various java ,jee technologies and java frameworks..

OpenSource (Not only it is free, it source code will be given to programmers)

=>Spring frame is not only free framework.. its source code will be given to programmers along with spring framework installation.

Light weight

=> Spring s/w comes in the for zip file having less size
=> Spring Containers are light weight containers
   note: To create Spring container there is not WebServers/Application Servers..
          we just need to create object for one pre-defined java class.
=> To execute most of the spring apps (like standalone , distributed Apps) we just need JRE/JVM + spring libraries (jars)
=> Spring Boot kind env.. supplies built in /embedded Strvers like Tomcat, jetty and etc.. So there is no need of arranging them seperately.

Loosely coupled

=>If the degree of dependency b/w two comps is less ,then they are called loosely coupled comps otherwise they are called Tight coupled comps..

Spring is loosely coupled :

=> Spring modules are loosely coupled () o the dependency among the spring modules except with spring core module is very less)
=> We can develop spring App either using 1 module or 2 module or n modules or all modules of spring App..
=>Spring Libraries(jars)/source code/ documentation are given as module wise ,So we can to use only those modules in which we are intrested in . (this helps to reduces project sizes)

Aspect Oriented (refer previous classes)

=>Allows to seperate primary and secondary logics at the time development and mixes them dynamically at runtime..
(refere previous class)

Easily Integratable

=> Spring can be used to develop entire App by using spring itself.. or spring can be used along with Existing Java Technologies or Java frameworks to develope the spring Applications.

Dependency Management/ IOC (refer previous classes)

=> Spring Contaienrer capable arranging dependent class object for target calss object by taking of every process that is involved..

Non-Invasive (refer previous classes)

=>The classes and Interface we develop in spring appliations /projects are ordinary classes or Interfaces (i.e classes or Interfaces with out any specialities).. Spring Support POJO /POJI modal Progamming .. This makes spring as non-invasive .

Application framework

=>Since the spring framework can be used to develop all kinds of Applications (like standalone Apps, web applications, distributed Applications and etc ...) and also can be used to develope all kinds of logics like presentation logics, b.logics , peristence logics and etc.. So we can say spring is Application framework.

What is a Container?

=>Container is a software program that manages the life cycle of given comp birth to death (Object creation to object destruction)

=> It is like an aquirium managing the life cycle of fishes..

=>Servlet container manages the life cycle servlet comps

=>Jsp container manages the life cycle jsp comps

=> SpringContainers/IOC container manages the life cycle of spring beans.

|--->BeanFactory (Basic)

|-->ApplicationContext (advanced) (Best)

IOC ::Inversion of Control

Servlet ,Jsp containers are heavy are weight becoz they need the heavy weight webSErver or Application server softwares..
Spring/ IOC containers are not alternate for servlet,jsp containers

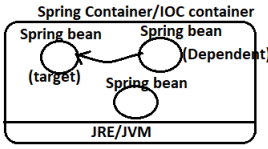
=>Spring Containers are light weight .. i.e by creating object for one pre-defined class we can create IOC container/spring container

To create BeanFactory container to create object for the class that implements org.springframework.beans.factory.BeanFactory(I)

- XmlBeanFactory (Impl class) (deprecated from spring 3.x)
- DefaultListableBeanFactory(impl class) (Recomanded to use)

Spring/IOC containers are given for spring bean life cycle management and for Dependency Management (Arranging Dependent class object to Taget class object)

=>Spring /IOC containers run on the top of JVM/JRE



=> The java class whose obj is created and managed by spring container is called spring bean.

4 approaches of spring App development
==========
a) Using xml driven configurations
b) Using Annotation driven configurations
c) 100% Code driven configurations
d) Using Spring Boot

=>While Developing spring application using Xml driven configurations we need to give inputs /instructions to SpringContainer/IOC container by spring bean cfg file (xml file)

=> any <file-name>.xml can be taken as spring bean cfg file.. but the recomanded name is applicationContext.xml

=> we need to pass spring bean cfg file as the input value while creating IOC containers /Spring Containers.

=> Generally Spring Bean cfg file contains the following details

a) configuring java classes as spring beans having bean ids (object names)
b) Mode of Dependency Management configurations
and etc..

making underlying container recognizing certain class/inteface/code or file with certain purpose is called confuguration...

eg:: By using <bean> tag in Spring bean file we try to make spring container to recognize given java class as spring bean .

eg:: By using <servlet>,<serlvet-mapping> tags web.xml we make servlet container recognizing certain java class servlet comp class.

applicationContex.xml (com/nt/cfgs)

```
<beans ....>
  <bean id="dt" class="java.util.Date"/>
</beans>
```

bean id (or) obj name

fully qualified java class to take as spring bean

=> when IOC container creates spring bean class object the bean id becomes object name (refer variable names pointing to spring bean class obj)

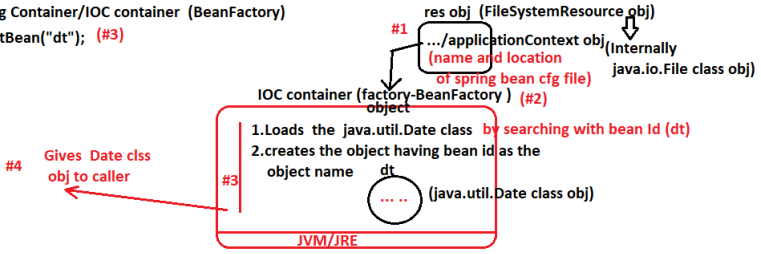
To create BeanFactory IOC container

```
//Locate and hold SpringBean cfg file
#1 FileSystemResource res=new FileSystemResource("...../applicationContext.xml");
```

Internally uses java.io.File to locate and hold given spring cfg file from the specified path of the computer drive (File system)

=>All files and folders of a computer belonging to different drives togather is called filesystem

```
//create BeanFactory IOC container
  XmlBeanFactory factory=new XmlBeanFactory(res);  (#2)

// get Spring Bean class from Spring Container/IOC container (BeanFactory)
#4 Date d1=(Date)  factory.getBean("dt");  (#3)
```



note1:: Bean ids with in the IOC container must be unique i.e Two spring bean can not have same bean ids.

note2:: BeanFactory Container perform lazy instantation of spring beans i.e until we call factory.getBean(-) method It will not attempt to create spring bean class objects.

Procedure to develop first Spring Application showing IOC container creation and spring bean management?

step1) Download and install Eclipse JEE Ide (either 2020-06 or 2020-09)

download as zip file from thios url ::https://www.eclipse.org/downloads/packages/release/2020-06/r and extract the zip file .

eclipse-jee-2020-06-R-win32-x86_64.zip

step2) keep the s/w setup ready in u r computer

jdk 1.8+ (java 13)
spring 5.3.x (5.3.1) (as zip file from https://repo.spring.io/release/org/springframework/spring/5.3.1/) (select ...dist.zip file)
Eclipse 4.x (4.17 - 2020-06) (as zip file from https://www.eclipse.org/downloads/packages/release/2020-06/r)

step3) Launch Eclipse IDE by choosing workspace folder

G:/workspaces/spriing/ntsp613 (the foder where eclipse proects will be saved)

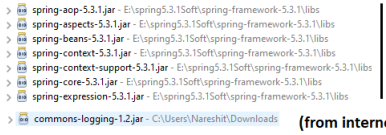
use eclipse.exe file from zip file extraction to start eclipse

step4) create Java Project in Eclipse IDE having name IOCProj1-SpringBeanBaiscs

File menu ---> new ---> project ---> java project ---> name:: IOCProj1-springBeanBasics ---> next --> say no to modules --> finish

step5) add spring core module libraries to the project

Right click project --> buildpath --> configure build path ---> Librbraries tab-->select classpath --> add external jars ---> select following jar files



from <spring_home>\libs folder

(from internet as supporting jar file doing logging operation)
collect from ::https://mvnrepository.com/artifact/commons-logging/commons-logging/1.2)

Bean class /Component class

=>The java class that is having both state and behaviour and state is used inside the behaviour is called Bean class /component class..

```
public class BankService{
  private String brachLocation;
  private String bankName;
  private String ifscCode;
  private long locationPin;

  public   String openAccount(CustomerDetails details){
      .... //ifscCode and branchLocation and bankName Details will be utilized
      ....
  }

  public  String trasnferMoney(int srcAcno, int destAcno , float amt){
      ... //transfermoney logic. here also ifsc code, bankName,brachLocation
      ...    details will be used.
  }
}
```

State is given

behaviour (methods)

Component/Bean class generally maintain lots of reusable logics.. in form of methods..

=>Component class can be a POJO class or non-POJO class

=>Compnent/Bean class can not be Java Bean class.

SpringBean calss

=>The Java class whose object is created and managed by spring container is called spring bean ..

=>Spring Bean class object life cycle should taken care by Springcontainer i.e right from Object creation to Object destruction the spring Cotnainer should take care of every thing.

=>Spring Bean can be a user-defined class or pre-defined class or third party supplied class.

=> Spring bean can be a Java bean class or Bean /component class.

=>Spring bean can be a POJO class or non-POJO class.

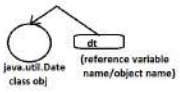
=>Spring bean can be a Invasive class or non-invasive class.

=> Every Java class should be cfg in spring bean cfg file (xml file) using <bean> tag.

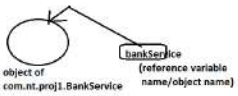
In xml env...

eg1:: `<bean id="dt" class="java.util.Date"/>` (id = object name, class = class name)

Spring container creates object for given Spring bean class having given bean id as the object name



eg2:: `<bean id="bankService" class="com.nt.proj1.BankService"/>` (id = bean id)



Annotation env..

```
@Component("bankService")
public class  BankService{
   ...
   ...
   ...
}
```

(bean id)

=> When spring container cretes the object for spring bean class the object name will be bean id. and spring bean is identified with its bean id.

=>every Java bean class is POJO class? (yes)

=>every bean /component class is POJO class? (may be or may not be)

=> Java bean class can be a spring bean ?(yes)

=> Component class can be a spring bean ?(yes)

=> Abstract class/interface can be taken as spring bean?(Directly not possible)

=> Java class with settters,gotters and b.methods is called what ?

a) java bean class b) Bean /component class (Answer is (b))

=>POJO class should have getters and setters? (no)

=>Spring bean must have setters && getters ? (no)

=> Any class can be taken as spring bean ? (yes)

=>By default every java class is compoment class ?(no)

=>POJO class is java class with out any specialaties

=>POJI is a interface with out any specialaties

=>Java Bean is a java class with standards (mainly getters ,setter methods)

=>Component/bean class is java class that is having both state and behaviour and state is used in the b.logic of the behaviour (methods)

=>Any class java class whose object is created and managed by Spring container is called spring bean.

Spring Core

=>Base module for other modules

=> It is standard module in spring application development.. (This module is required as minimum module)

=> If this module is used alone to develop spring Apps.. we can develop only standalone Apps.

=> This module is given mainly for two operations

a) To supply spring containers /IOC containers

i)BeanFactory Container (basic)

ii) ApplicationContext Container (advanced)

IOC: Inversion Of Control

b) spring Bean life cycle management and Dependency Management (IOC)

|--> Assinging Dependent spring bean class object to Target Spring bean class object.

DependencyManagement is also called as IOC (Inversion of Control) . Since SpringContainer are capable of performing the Dependency Management ,So they are also called IOC containers..

Taking care of all activities on spring bean class from birth to death (Object creation to object destrucion)

Target class :: The Main class that uses other class services

dependent class :: The helper class used by main class..

There will be Compositon(HAS-A relation) between target class and dependent i.e Target class will maintain the object dependent class.

Target class	Dependent class
Student <-------->	Course Material
Flipkart	DTDC
Employee	Department
Vehicle	Engine
Student	Facutly
Student	LibraryMembership
PErson	Passport

The target/main class needes Dependent class So The target class obj maintais dependent class objet.

Two types Dependency Management

a) Dependency Lookup

b) Dependency Injection

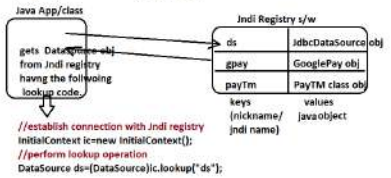
Depedency Lookup

Here Target /main class writes logics and spends times to search and get Dependent/helper class object.

=>Here Target class pulls dependent class object from different resources..

eg1:: Student getting coursematerial by asking/requesting for it from a organization..
(target) (dependent)

eg2:: Java App /Class getting jdbc DataSource object from Jndi registry through jndi lookup operation
(target) (Dependent)



Jndi registry provides global visiblity objects having nicknames/jndi names..

JNDI :: Java Naming and Directory Interface

```
//establish connection with Jndi registry
InitialContext ic=new InitialContext();
//perform lookup operation
DataSource ds=(DataSource)ic.lookup("ds");
```

Limitation of DL (Dependency lookup):::

->Here targert class should spend some time and logics to search and get dependent class object.

advantage:::

-> Here target class will get only required Dependent class object.

b) Dependency Injection

HEre the underlying server /container/framework/ JVM/ any Runtime env.. dynamically assigns/injects Dependent class object target class object.. In this process it only creates both target and Dependent class objects if needed.

eg1:: Student getting course meterial the moment he joins for the course

eg2:: Servlet container assisnging ServletConfig object to Servlet class object the moment it creates the Servlet class object

eg3:: JVM assinging default values to object once the object is created..

Limitation::

-> Here the underlying server/container/JVM/... etc.. may inject both neccessary and unneccessary values

advantage:

->Target class can use dependent directly with out searching for it.

-> It is faster to get and use Dependents.

Spring support both Dependendency Lookup and dependency injection.
(spring uses more of this)

Dec 5th and 6th::
Junit 5, mockito3 and HttpUnit
dec 5th :: 5pm
dec 6th :: 10 am

POJI (Plain Old Java Interface)
====
=>An interface with out specialaties is called POJI.
=> It is the interface not extending from technology/framework specific interfaces..
=> Most of the times POJI can be compiled by using jdk libraries. i.e there is no need of technology /framework Specific libraries (jars) in classpath.

```
interface Test{
  ....
  ....
}
"Test" is POJI
```

```
interface Test1 extends Demo1{
  ....
  ....
}
interface Demo1{
  ....
}
"Test1","Demo1" interfaces are POJIs
```

```
interface Test2 extends java.io.Serializable{
  ...
  ...
}
Test2 is POJI.
```
It is part of language api

```
interface Test
 extends java.rmi.Remote{
  ....  part of RMI Technology
  ....
}
```
not "Test" is POJI

```
interface Test3 extends Test4{
  ...
  ...
}
interface Test4 extends java.sql.Connection{
  ...
  ...
}
```
Test3,Test4 are not POJIS

```
@Remote (part of EJB API)
interface Test6 {
  ...
  ...
}
```
"Test6" is POJI. but we can not compile Test6 class only using jdk libraries .. We must have EJB libraries. (Exceptional case)

note:: spring framework is non-invasive (loosely coupled with spring aPI)becoz it suppors POJO-POJI model programming)
if the degree of dependency is less b/w two comps then they are called loosely coupled comps
eg:: TV and Remote , pen and paper and etc..
if the degree of dependency is more b/w two comps then they are called tightly coupled comps
eg:: CPU Box and console , bike and rider , Engine and fuel and etc..

JAva Bean class
========
=> The java class that is developed by following some standards is called Java Bean.
=> We do not use java bean classes as main classes of the Project they will be used as hepler classes in project to pass multiple values from 1 main class to another main class.

Flipkart ---Java Bean class obj---> DTDC
(shopping logic) (delivery logic)

E-Commerce App <---Card Details as Java Bean class obj---> Google pay

=> The standards required to develop the java class as java bean class are
a) class must be taken as public class. (To make visible across the multiple packages)
b) Recomanded to implement java.io.Serializable(I) [To send its object data over network]
c) All member variables must be taken as prviate and non-static. (for Encapsulation)
d) Every Member variable(java bean class property) must have 1 setter method and 1 getter method
note: setter methods are useful to set or modify data to bean properties and getter methods are useful to read data from bean properties.
e) must have one 0-param constructor directly(given by pogrammer) or indirectly (given by java compiler as default constructor) (For easy Object creation)

=>We can not send normal object's data over the network..
We can send only Serializable object's over the network..

EmployeeBean.java
---
a
```
public class EmployeeBean implements java.io.Serializable{
  //Bean properites
  private int eno;
  private String ename;
  private String eadd;
  private float salary;
    //setters && getters
  public  void setEno(int eno){
    this.eno=eno;
  }
  public  int getEno(){
    return  eno;
  }
  ....
  ...
  ...
  ...
  ..
  ..
}
```
b
c
(d)

=>Every Java Bean class is POJO class but every POJO class need not be a java bean class.

Need of Java Bean in real practices

```
//business class /service class having b.method with b.logic
public calss StudentExamsService{

 public String gnerateRank(int sno,String sname,Strng sadd, int m1,int m2,int m3){
   //calc total
     .....
   // generate avg
     .....
   // generate grade/rank
     .....
   //return rank/grade
   return ".....";
 }
}//class
```
6 params

We should not desgin Java method having more than 3 params becoz of the following limitations
a) Caller should remeber complex signature of the method.
b) Caller should rember the order of parameters while calling the method
c) Caller should rember the index of parameters while calling the method
d) we can not ignore to pass one or two argument values.. still we need to pass meaningful default values there..

Client App

```
public class ClientApp{
  p s  v main(String args[]){
     StudentExamService service=new StudentExamService();
     String rank=seevice.generateRank(101,"raja","hyd",78,89,56);
     S.o.p(rank);
   }
}
```

Solution (Design Java method having java bean as the parameter and pass data from Client App as Java bean class object)
=======

```
//java bean
public class StudentBean implements Serializable{
  //bean properties
  private int sno;
  private String sname,sadd;
  private int m1,m2,m3;
  //setters && getters
   .....  (6+6)
    ....
   .....
}
```
Optional

```
//Service class/Business class

public class StudentExamService{

public String generateRank(StudentBean bean){
  //calc total
    .....
  // generate avg
    .....
  // generate grade/rank
    .....
  //return rank/grade
  return ".....";
}
}
```
1

//use getter method on bean calss obj to get data..(sno,sname,sadd,m1,m2,m3 values)

Client App
==========
```
public class ClientApp{
  p s v main(String args[]){
     StudentBean st=new StudentBean();
       st.setSno(101);
       st.setSadd("hyd");
       st.setM3(89); st.setM2(56);
     //service class obj
      StudentExamService service=new StudentExamService();
       String rank=service.generateRank(st);
         S.o.p(rank);
     }
}
```

Advantages of passing data as java Bean class obj
==========
a) Method signature looks very simple to rember
b)No need of remembering order while setting data to JAva bean class obj
c) No need of remembering any indexes
d) we can ignore to set the values to java bean class object, if we do not want to set and no need of passing any default value..
e) It is indurstry standard

```
        return "....";
    }
}
```

note:: To overcome the limitations of oop style programming.. use AOP style programming.

Solution using AOP (ASpect Orientted Programming)

note: Do not mix primary and secondary logics at the time of development , always develop them seperately .
mix them dynamically at run time with the support AOP enabled softwares like spring AOP,AspectjAOP, JbossAOP and etc..

```
public class BankService{

 public String withdraw(long acno,float amt){
  //primary logic
  balance=balance-amt;
  return "....";
 }
 public String deposite(long acno,float amt){
  //primary logic
  balance=balance+amt;
  return "....";
 }
}
```

```
public class Security{
  ...
  ... security logic
  ...
}

public class Logging{
  ...
  ... Logging logic
  ...
  ...
}

public class Auditing{
  ...
  ... auditing logic
  ...
  ...
}
```

AOP Enabled software (Spring AOP/AspectjAOP) collects the instructions from xml configurations (or) Annotation cfgs

Generates class at runtime as dynamic class /proxy class in the Memory of JVM (This class is called InMemory Proxy class)

this class is called InMemoryProxy class becoz it is created in the memory JVM memory of RAM where the application executes.

```
public class BankService$Proxy extends BankService{

 public withdraw(long acno, float amt){
   ....
   .... //contains both primary and secondary logics
   ....
   ....
 }
 public deposite(long acno, float amt){
   ....
   .... //contains both primary and secondary logics
   ....
   ....
 }
}
```

RAM
JVM Memory of the Application

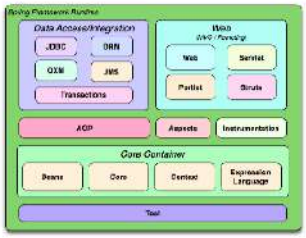
```
public class BankService$Proxy extends BankService{
  ...
  ...
}
```

Advantages of AOP style programming

a) Secondary logics are reusable across the multiple b.methods of multiple business classes
b) Code does not become clumzy becoz secondary logics seperated from primary logics
c) The Enabling or Disabling of secondary logics on primary logics can be done with out touching the source code
d) It is industry standard process..

note:: seperating one primary logic from another primary logic does not come under AOP style programming
for example seperating b.logic from persistence logic is not AOP..
note: As of now spring f/w supporting two implementations of AOP style programming
they are a) Spring AOP b) AspectjAOP.

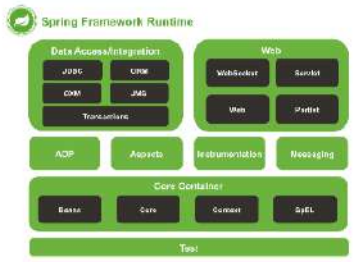
Spring 3.x overview diagram



In spring 3.x/4.x/5.x 20 + modules are given by doing
a) They divied 2.x big modules in to small modules
b) Extra modules are added like "Test","portlet","SPEL" and etc..

Spring 5.x compitable with jdk 1.8 to jdk 15 features..
Soring Boot latest version is 2.4.x is built on top of spring 5.3.x giving support for jdk 1.8 to jdk 15 features..

Spring 4.x and 5.x overview diagram



What is difference among the POJO class, POJI , Java Bean class, Bean class/Component class and Spring Bean class?

POJO class (Plain Old Java Object class)

=>The Java class with out any specialaties is called POJO class
=>The Java class that is not implementing /extending from Certain Technology/framework API Specific interfaces/classes is called POJO class.
eg: The class developed in Spring App .. not implementing/extending from Spring API interfaces / classes is called POJO class.
=>Most of times we can compile POJO class by just having jdk libraries (jars) i.e not adding any Technonology/framework specific jars/libraries.

```
public class Demo{
  ...
  ..
}
```

"Demo" is POJO class

```
public class Demo1 extends HttpServlet{
  ...                          part of servlet api
  ...
}
```

"Demo1" is not a POJO class

```
public class Demo2 extends Thread{
  ...                          It is part of java api
  ...
}
```

"Demo2" is pojo class

```
public class Demo3 implements java.rmi.Remote{
  ...                          part of RMI Technology
  ..
}
```

"Demo3" is not a POJO class

=>POJO class is not against of exceding from class or implementing interfaces..
It is against of extending from framework/Technology specific classes and against of implementing framework/technology specific interfaces..

```
public class Demo4 extends Demo5{
  ...
  ...
}
public class Demo5 implements java.io.Serializable{
  ...                          part of java api
  ...
}
```

"Demo4,Demo5" classes are POJO classes

```
public class Demo6 extends Demo7{
  .....
  ...
}
public class Demo7 implements java.sql.Connection{
  ...                          part of jdbc api
  ...
}
```

"Demo6,Demo7" are not pojo classes

```
public class Demo8{

public void m1(){
  .... //logic using spring api
  ...
 }
}
```

=>It is a POJO class but can not compiled only having jdk libraries (Exceptional case)

to

```
@Entity --> belongs hibernate api
public class Demo9{
  .....
  ....
}
```

"Demo9" is POJO class But can not be compiled using Jdk Libraries .. (Exceptional case)

=> Lang api, utility api , IO streams , Collections, ReflectionAPI , networking api, multi threading , streaming api ,awt, swings are part of java language..
=> JDBC,JNDI,RMI, Servlet, Jsp, JTA,Java mail , JAAS and etc.. are Java Technologies
=> Spring ,hibernate, Struts, JSF, webServices, and etc.. are java java frameworks..

POJI (Plain Old Java Interface)

Java FullStack Developer (/3r/5r/5r ...) (12 to 16 lars package)

a) Java radio

Core Java, adv.Java, spring with boot, webServices/microServices, Design Patterns,Hibernate (Java developer)
maven/gradle, Jenkins, putty, Kubernetes, Ansible and etc...
Docker, log4j, junit, mockito, agile, jira and etc...

b) UI Technologies (UI Developer)

html, css, java script, jquery/angularJs/angular/reactJs (Node)

c) DB s/w (SQL developer)

=>Oracle/mysql/postgreSQL (SQL DB s/w), MongoDB (No SQL DB s/w)
note:: Good knowledge of PL/SQL programming required

d) DevOPs+ AWS (In Build and Deployment) (DevOps and Cloud Engineer)

DevOps tools :: maven/gradle, jenkins, putty, kubernetes, Ansible and etc...
Docker, Git

=>Hiring java coder for experience peopled ---> 4 to 10 lacs
=> Java trainer :: core java + adv java + spring and oracle, html ,css ,java script (3 to 5 lacs)

Spring JEE module

=>This module provides abstraction on multiple JEE, EE module technologies and simplifies their work
=>This module having lots of sub modules.. in that most of them are outdated..

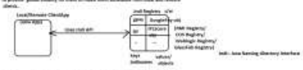
a) Spring mail/Spring Email

=> Spring mail provides abstraction on java mail api and simplifies mailing operations..

Java App --JDBC API--> DB s/w (oracle, mysql, postgreSQL and etc..)
Java App --Java Mail API--> Mailserver s/w (James mail Server, Microsoft Exchange server,SMTP Gse)
These are responsible to manage email accounts and email messages..

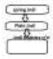
b) Spring Jndi

To provide global visibility and accessibility to java objects we need to place them in Jndi registry s/w.. All Distributed service related objects like GooglePay object, PayTM object, IPLScore Comp obj ,Weather Report obj will be placed jndi registry to provide global visibility for them to make them accessible from local and remote clients..



Jndi:: Java Naming directory interface

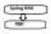
Spring Jndi provides abstraction on plain Jndi and simplifies jndi work/operations.



Bind :: Putting object to Jndi registry
Rebind :: replacing existing object from Jndi registry
Unbind :: removing object from jndi registry
lookup :: searching and getting object from jndi registry..

c) Spring RMI (outdated)

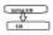
Spring RMI provides abstraction on plain RMI and simplifies Distributed App development
Since RMI is outdated , Spring RMI is also outdated..



RMI: Remote Method Invocation

d) Spring EJB (outdated)

Spring EJB provides abstraction on plain EJB and simplifies Distributed App development..



note:: Nowadays Industry is using webservices to develop Distributed Applications..
Spring Rest (Separate extension module) is given to WebServices in spring (Restfull webServices)

Spring Rest pre-requisites
corejava, spring core, spring MVC

=> more sub modules are in spring JEE like Spring JMS, Spring JTA and etc..

Spring AOP module (AOP :: Aspect Oriented Programming)

=>AOP is new methodology of programming that is complementing OOP style programming i.e AOP is not replacement for OOP.. it is enhancement to OOP.

=> The logics that are minimum and mandatory to complete the task are called Primary logics or main logics
eg:: getting balance, opening account, withdraw logic, deposite logic , transfer money logic
=> The logics that are additional, optional and configurable are called secondary logics.. i.e with out logics also our App runs.. But to run the App effectively these logics are required..
eg:: logging (keeps track code flow) , Auditting (keeps user activities), security , performance monitoring and etc..

Example Oop style programming (Here we mixup both primary and secondary logics) in b.methods

```
public class BankService{

 public String withdraw(long acno,float amt){
   //security logic     |
   ...                  |  Secondary logics
   // logging logic     |
   ...                  |
   // auditing logic    |
   ....
   //primary logic(B.logic)
   balance=balance-amt;
   return "....";
 }

  public String deposite(long acno,float amt){
   //security logic     |
   ...                  |  Secondary logics
   // Logging logic     |
   ...                  |
   // auditing logic    |
   ....
   //primary logic(B.logic)
   balance=balance+amt;
   return "....";
  }
}
```

Limitation with OOP style programming

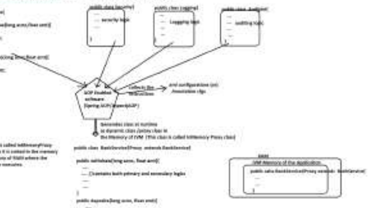
a) Secondary logics are not reusable across the multiple b.methods
b) Code becomes clumsy becoz we mixup both primary and secondary logics
c) To Enable or disable secondary logics on primary logics we need to modify the source code of classes..
d) It is not recommended process and not the industry standard process.

note:: To overcome the limitations of oop style programming.. use AOP style programming..

Solution using AOP (Aspect Oriented Programming)

note: Do not mix primary and secondary logics at the time of development , always develop them separately.. mix them dynamically at run time with the support AOP enabled softwares like spring AOP,AspectJAOP, JbossAOP and etc..

```
public class BankService{

 public String withdraw(long acno,float amt){
  //primary logic
  balance=balance-amt;
  return "....";
 }
 public String deposite(long acno,float amt){
  //primary logic
  balance=balance+amt;
  return "....";
 }
}
```



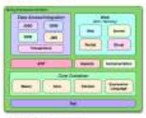
this class is called InMemoryProxy class becoz it is created in the memory (JVM memory of RAM) where the application executes..

Advantages of AOP style programming

a) Secondary logics are reusable across the multiple b.methods of multiple business classes
b) Code does not become clumsy becoz secondary logics separated from primary logics
c) The Enabling or Disabling of secondary logics on primary logics can be done with out touching the source code
d) It is industry standard process..

note:: separating one primary logic from another primary logic does not come under AOP style programming
For example separating b.logic from persistence logic is not AOP..
note:: As of now spring f/w supporting two implementations of AOP style programming
They are a) Spring AOP b) AspectJAOP

Spring 3.x overview diagram



In spring 3.x/4.x/5.x 20 + modules are given by doing
a) Then divided 2.x big modules in to small modules
b) Some modules are added like "Test", "portlet", "XML" and etc..

Spring 5.x compatible with jdk 1.8 to jdk 15 features..
Spring Boot latest version 2.4.x is built on top of spring 5.3.x
giving support for jdk 1.8 to jdk 15 features.

Spring 4.x and 5.x overview diagram

What is difference among the POJO class, POJI , Java Bean class, Bean class/Component class and Spring Bean class?

POJO class (Plain Old Java Object class)

=>The Java class with out any specialities is called POJO class
=>The Java class that is not implementing /extending from certain Technology/framework API Specific Interfaces/classes is called POJO class.
eg: The class developed in Spring App... not implementing/extending from Spring API interfaces / classes is called POJO class.
=>Most of times we can compile POJO class by just having jdk libraries (jars) i.e not adding any Technology/framework specific jars/libraries.

```
public class Demo1{
 ...
 ..
}
```
"Demo" is POJO class

```
public class Demo2 extends HttpServlet{
 ...
 ...
}
```
(HttpServlet: part of servlet api)
"Demo2" is not a POJO class

```
public class Demo3 extends Thread{
 ...
 ...
}
```
(Thread: it is part of java api)
"Demo3" is pojo class

```
public class Demo4 implements java.rmi.Remote{
 ...
 ...
}
```
(java.rmi.Remote: part of RMI Technology)
"Demo4" is not a POJO class

```
public class Demo5{

public void m1(){
  ... //logic using spring api
  ....
 }
}
```
=>It is a POJO class but can not compiled only having jdk libraries (Exceptional case)

=>POJO class is not against of extending from class or implementing interfaces.. It is against of extending from framework/Technology specific classes and against of implementing Framework/Technology specific interfaces..

```
public class Demo6 extends Demo5{
 ...
 ...
}
public class Demo6 implements java.io.Serializable{
 ...
 ...
}
```
(java.io.Serializable: part of java api)
"Demo6,Demo6" classes are POJO classes

```
public class Demo6 extends Demo5{
 ...
 ...
}
public class Demo7 implements java.sql.Connection{
 ...
 ...
}
```
(java.sql.Connection: part of jdbc api)
"Demo6,Demo7" are not pojo classes

```
@Entity --> belongs hibernate api
public class Demo8{
 ...
 ...
}
```
"Demo8" is POJO class But can not be compiled using Jdk libraries... (Exceptional case)

=> Lang api, utility api , IO streams , Collections, Reflection API , networking API, multi threading , Annotations api ,awt, swings are part of Java language..
=> JDBC,JNDI,RMI, Servlet, Jsp, JTA,Java mail , JAAS and etc... are Java Technologies
=> Spring ,hibernate, Struts, JSF, webServices and etc... are Java Java frameworks..

POJI (Plain Old Java Interface)

Java FullStack Developer (/3+/4+/5+...) (12 to 16 lacs package)

a) Java suite

Core Java , adv.java, spring with boot, webServices/microServices, Design Patterns,hibernate (java developer)
maven/gradle, jenkins, putty, kubernetees ,Ansible and etc..
Docker, log4j, junit, mockito ,agile, jira ane etc..

b) Ui Technologies (UI Developer)

html, css, java script , jquery/angular/angularJs/ReactJs
(best)

c) DB s/ws (SQL developer)

=>Oracle/mysql/postgreSQL (SQL DB s/ws) , MongoDB (No SQL DB s/w)
note:: Good knowledge of PL/SQL programming required

d) DevOPs+ AWS (In Build and Deployment) (DevOps and Cloud Engineer)

DevOps Tools :: maven/gradle, jenkins, putty, kubernetees ,Ansible and etc..
Docker ,GIT

=>Only Java suite for experience peopled ---> 4 to 10 lacs
=> Java Fresher :: core java + adv java + spring and oracle , html ,css ,java script (2 to 5 lacs)

Spring JEE module

=>This module provides abstraction on multiple JSE, JEE module technologies and simplifies their work
=>This module having lots of sub modules.. in that most of them are outdated..

a) Spring mail/Spring Email

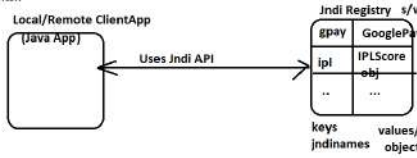
=> Spring mail provides abstraction on java mail api and simplifies mailing operations..

JDBC API
Java App ----------> DB s/w (oracle ,mysql, postgreSQL and etc..)

Java Mail API
Java App ----------> MailServer s/w (James mail Server , Microsoft Exchange server,SMTP One)
These are responsible to manage email accounts and email messages..

b) Spring Jndi

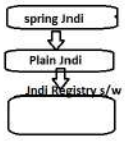
To provide global visibility and accessibility to java objects we need to place them in Jndi registry s/w.. All Distributed service related objects like GooglePay object, PayTM object, IPLScore Comp obj ,Weather Report obj will be placed jndi registry to provide global visibilty for them to make them accessible from local and remote clients..



(RMI Registry/
COS Registry/
Weblogic Registry/
GlassFish Registry)

Jndi:: Java Naming directory Interface

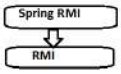
Spring Jndi provides abstraction on plain Jndi and simplties jndi work/operations..



Bind :: Putting object in Jndi registry
Rebind :: replacing existing object from Jndi registry
unbind :: removing object from jndi registry
lookup :: searching and getting object from jndi regitry..

c) Spring RMI (outdated)

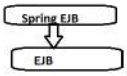
Spring RMI provides abstraction on plain RMI and simplifies Distributed App developement
Since RMI is outdated . Spring RMI is also outDated..



RMI :Remote Method Invocation

d) Spring EJB (outdated)

Spring EJB provides abstraction on plain EJB and simplifies Distributed App devleopment..



note:: Nowadays industry is using webservices
to develop Distrobuted Applications..
Spring Rest (Seperate extenstion module) is given to
WebSErvices in spring (Restfull webServices)

Spring Rest pre-requisites:
servlet, spring core, spring MVC

=> many sub modules are in spring JEE like
Spring JMS, Spring JTA and etc..

Spring AOP module (AOP :: Aspect Oriented Programming)

=>AOP is new methodology of programming that is complementing OOP style programming i.e
AOP is not replacement for OOP .. It is enhacement to OOP.

=> The logics that are minimum and mandatory to complete the task are called Primary logics
or main logics
eg:: getting balance, opening account, withdraw logic, deposite logic , transfer money logic
=> The logics that are addtional ,optional and configurable are called secondary logics.. i,e with out logics
also our App runs.. But to run the App effectively these logics are required..
eg:: logging (keeps track code flow) , Auditing (keeps user activities), security , performence monitorin and etc..

Example Oop style programming (HEre we mixup both primary and secondary logics)
in B.methods

```
public class BankService{

 public  String withdraw(long acno,float amt){
  //security logic
     ...
  // Logging  logic      | Secondary logics
     ..
  //  auditing logic
      .....
  //primary logic(b.logic)
  balance=balance-amt;
  return ".....";
 }

    public  String  deposite(long acno,float amt){
      //security logic
          ...
      // Logging  logic      | Secondary logics
          ..
      //  auditing logic
          .....
      //primary logic(b.logic)
      balance=balance+amt;
      return ".....";
    }
}
```

Limitations with OOP style programming

a) Secondary logics are not reusable.
b)Code becomes clumzy becoz we mixup both primary and secondary logics
c) To Enable or disable secondary logics on primary logics
we need to modify the soruce code of classes..
d) It is not recomanded process and not the industry standard process.

note:: To overcome the limitations of oop style programming.. use AOP style progamming..

and managed by spring container)

=>Servletcontainer manages the life cycle of Servlet comps
=>Jsp Container manages the life cycle of jsp comps .
note : Spring containers are no way servlet,jsp containers...

Dependency Management

=> It is the process of assigning Dependent class object to target class object dymically at runtime..
=>In Regular java application programmer needs to do this work manually.. where as in spring App The spring container takes care of whole process nothing dependency management..

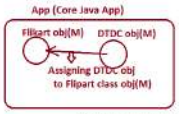
=>The class that uses other class in called target class /main class..
=> The class that acts helper class to other class is called Dependent/Helper class.

Target class/main class	Dependent class/Helper class	
Flipkart	DTDC	HAS--Relationshop (Composition)
Student	Course	
Employee	Department	
Student	Library	
Flipkart	PaymentBroker/Gateway	
Vehile	Engine	

Core JAva App (Basic App) (Flipkart using DTDC)

-> Programmer should following activies (manual process)
a) Load both target and dependent classes
b) create objs for both target and depedent classes
c) Assign dependent class obj to traget class obj
d) use Target class obj logics with along dependent class object.

(Makig coffe/tea manually)



M => indicates manual operation

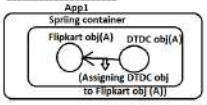
Spring Core module App (Framework App)

-> Create spring Container
->Supply input file(xml) to Spring container
|--->Declare Flipkar as target class
|--->Declare DTDC as depedent class
->Get Target class object having dependent class from Spring container and use it.

Here Spring container loads both target, dependent classes--> creates both objs --> assigns dependent class obj to target class obj internally.. (This is called Dependencyt managment)

(Automated process done by Spring container)
(getting tea/coffe from coffe/tea maker)

Spring Core module App
App1



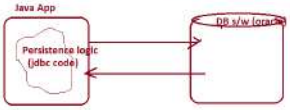
A =>Automated operation

Spring DAO module (DAO -->Data Access object)

=> Oracle, mysql,postgreSQL and etc..are called Db s/ws.. but they are also called Data Storage technologies
=> JDBC, Hiberate, springJDBC/DAO, Sprign ORM, spring Data and etc. are called Data Access Technologies/ frameworks becoz they are capable interacting DB s/ws.. can manipulate DB s/ws data (CURD operations) by developing persitence logic.

CRUD/CURD operations

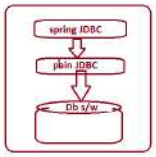
C--->create	The logic that performs CURD Operations is called PErsistence logic eg: jdbc code , hiberanate code, spring jdbc code spring orm code, spring data code..
R --->Read	
U-->Update	
D--->Delete	



Here
oracle is Data store technology/software
jdbc is Data Access technology/Persistence technology
jdbc code is called persistnece logic.

=>Spring DAO/JDBC provides abstraction (hiding details) on plain JDBC Technology and simplifies JDBC style SQL queries based persistnece logic development.
=> Spring DAO/JDBC Persitence logic is Db s/w dependent persitence logic becoz it is using the DB s/w dependent SQL queries.

plain jdbc App	DAO spring JDBC App
20 lines of code	5 lines of code
	(Here most of code will be geneated by spring DAO/JDBC internally by taking the support of plain JDBC technology)



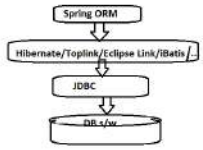
Both plain JDBC and spring JDBC Persitence logics are DB s/w depnedent logics becoz of the SQL Queries that they are using

Spring ORM (ORM :: Object relational mapping)

=> ORM frameworks or sofwares allows to us to develope objects based Objects based DB s/w independent Persitence logic with out any SQL Quries.. These ORM frameworks internally uses dyamically generated JDBC code as required for the underlying DB s/w.. but programmers never knows /works with that JDBC code.. he always works with java objects to develop and execute persitence logic.

ORM frameworks are :: hibernate, iBatis , JDO, OJB, Toplink ,Eclipse link and etc..

Spring ORM is not a ORM framework. IT is a spring module providing abstraction on multiple ORM frameworks as listed above..



=>spring ORM, ORM frameworks persistence logic is DB s/w independent persistence logic becoz it is based Object Persistence logic with out any SQL queries.

ORM = JPA
JPA =>Java PErsistence API
ORM =Object relational mapping

Spring web module (2.x)

spring web module 2.x = spring 1.x web module + spring 1.x mvc module + other concepets

spring 2.x Two parts is having two parts

part1) (bit old and outdated)

=> provides plugins(addtional code) make spring apps communicatable from other web framework Apps like Struts Apps, JSF apps and etc.. (out dated)

note:: Plugin is a additional /patch code that provides extra funcationalities to existing code or software

Part2) (Very very important)

=> It is part of spring framework or we can say sub framework.. nothing spring web MVC framework or spring MVC framework as alternate to STruts,JSF framwworks to develop MVC architecture based java web application..

=> spring mvc or spring web mvc is part of spring framework (sub framework in spring) providing abstraction on servlet ,jsp technologies to simplify mvc archiecture based web applicaltons..

Spring trendeing modules:: spring MVC, spring data, spring security , spring boot
(for both interview and job survival)

spring core ,springAOP, spring MvC, spring boot, spring data
(very imp for interview)

Modules of our spring course

- spring core
- spring JDBC/DAO
- spring Data
- spring ORM
- spring AOP
- spring TxMgmt
- spring MVC
- spring security
- spring Batch
- spring social
- spring oauth
- spring boot

+ 10 mini proejcts
+ 8 to 1 java realtime tools
+ 4 Servers
+ 3 Db s/w.
+ java8/9/10/.. features..

we implement spring Apps in 4 approaches
a) Using xml driven configrurations
b) Using annotation driven configrurations
c) Using 100% code configrurations
d) Using Spring boot

Spring videos url ::
https://www.youtube.com/watch?v=Bw3v1b3WjDM&list=PLVIQHNRLflP-wfUj1MAuLwIMekHpP-yQu

---

spring
======
type :: Java Application framework
version :: 5.3.1 (compitable with jdk 1.8+)
vendor :: Interface21 (pivotal team)
Open source
creator :: Mr.Rod Jhonson
To download spring framework :: download as zip file from
Go to https://repo.spring.io/release/org/springframework/spring
and select version (5.3.1) ---> spring-5.3.1-dist.zip file

To read spring docs :: https://docs.spring.io/spring-framework/docs/current/reference/html/

Books :: Spring LIVE ✕ (not required)

=>Attened classes
=> Collect notes and example apps
=> read Spring docs
=> use internet for gathering info

(Better than any book in the market)

Spring installation :: extract the zip file..

<Spring_home>\docs folder gives spring api docs , koltin spring api docs,
spring reference docs about different modules
<Spring_home>\libs folder gives jar files representing spring libraries of different modules
we get 3 flavours of jar files for every section/module
->jar file having byte code (.class files) (eg: spring-aop-<ver>.jar)
-> jar file having documentation (html files) (eg: spring-aop-<ver>-javadocs.jar)
-> jar file having soruce code (.java files) (eg: spring-aop-<ver>-soruces.jar)

<spring_home>\schemas folder glves .xsd files containing rules to develop xml files in spring Apps as
Spring bean confluration files
XSD --:: Xml Schema Definitation.

XSD file is a document that contains rules and guidline to construct xml file .. The rules are like
=>Structure to be followed (order of tags)
=>Tags to be used
=>attributes to used
and etc..

DTD is older and outdated
XSD is altenate for DTD
DTD ::Document type Definitation

All spring developers develops their xml files (spring bean cfg file) bssed on these xsd documents based rules..

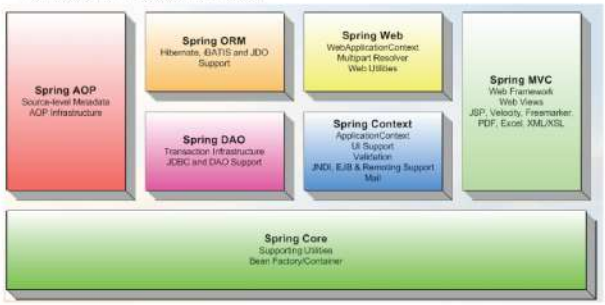
---

framework
Why spring is named as spring?
Ans) He is named after spring season that the rod jhonson enjoyed in the himalays..

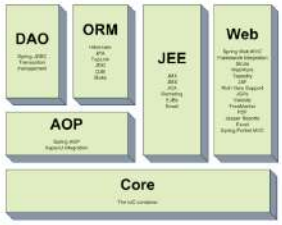
=>Spring versions 1.x /2.x/3.x/4.x/5.x (latest version 5.3.1)
=>Spring framework is given as modules (sub parts) to make programmers to uses their no.of modules as need in the spring based application devielopement..

spring 1.x :: 7 modules
spring 2.x ::6 modules
spring 3.x/4.x /5.x :: 20+ modules

Spring 1.x overview diagram (spring 1.x modules)
==========



Spring 2.x modules (spring 2.x overview diagram)
==========



=>Since there are 20+ modules in spring 3.x/4.x/5.x .. and we can not rember those many modules names.. So the industry uses spring 2.x moduels names as the standard modules ..

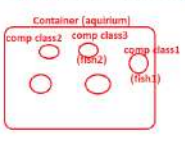
=>In Spring 3.x/4.x/5.x versions more modules are there becoz they divided big modules spring 2.x into multiple small modules and also added new modules..

=>The following modules are spring extensions i.e they are not part spring framework , So they must be arranged seperately..
=>spring security
=> Spring Batch
=> Spring Social
=>Spring Oauth
=>Spring Boot
=>Spring data
and etc..

=> Sping 2.x web module = spring 1.x web module +spring 1.x mvc module + other concepts
=>Spring 2.x JEE module =Spring 1.x context module + new concepts

Spring Core Module
==========
=>It is base module for other modules ..
=> While developing any module(s) based spring App .. base will be utilized directly or indirectly.
=>It is standard module to use in all spring Apps development .. This module is minimum in any spring App development.
=>If this module is alone .. we can develop only standalone Apps.
=>If this module is used along with other modules we can develop various types apps like web applications, distributed Apps.. But this module work in those Applications is provding
a) Spring Containers
b) Performing Dependency Management.
=>This module is mainly given for
a) Spring Containers
i)BeanFactory Container (base)
ii) Application Container (advanced)
b) Dependendency Management.

=>Container is a software prg that manages whole life cycle of given component (reusable java class) from brith to death (from object creation to object destruciton) i.e container loads the class , creates the object , container manages the object and container only destroy the object..
=>Spring Container manages the life cycle of spring beans (the Classes whose objects are created and managed by spring container)
=>Servletcontainer manages the life cycle of Servlet comps
=>Jsp Container manages the life cycle of jsp comps .
note : Spring containers are no way servlet,jsp containers...



Mr.Rambabu (admin)
9866545966-whatsapp number
Naresh IT Online Team
support@nareshit.com

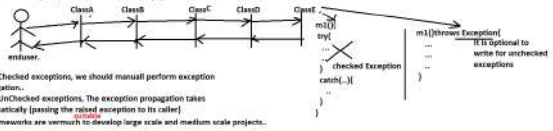
nataraajavaarena@gmail.com
FB group :: nataraajavaarena
url:: https://www.facebook.com/groups/388095825162910
(Faculty info)

Spring 5.x with spring 2.x

=>Course fee: 3500/-
=>Duration : 120 sessions (1 slot) (6pm)
70-75 sessions (2 slots) (8pm and 7am)
=>6/7 classes per week (daily 1:35 to 1:45mins)
=>Free realtime java tools
=>pre-requisite:: Core JAva oops/exception handling

Advantages of working with frameworks

=> Improves the productivity (Doing More work in less time)
=>Avoids the boilterplate code problem
=> Frameworks APIs (packages having classes,interfaces,enums, annotations) are designed based realtime scenarions.
=>Most the Framework API methods are designed to throw unchecked exceptions , So exception handling is optional and exception propagation is possible.



=>For Checked exceptions, we should manuall perform exception propagation..
=>For UnChecked exceptions, The exception propagation takes automatically (passing the raised exception to its caller)
=> Frameworks are vermuch to develop large scale and medium scale projects..
and etc..

Based on the kind of applications we develop there are 4 types of JAva frameworks

a) Web Application frameworks
b) ORM frameworks (ORM :: Object relational mapping)
c) Application Frameworks
d) WebService/Distributed Application frameworks

a)Web application frameworks

=>Developed based on servlet,jsp technologies to simplify MVC architecture java web application Development..
MVC (Model,View,Controller) is industry standard Architecture to develop web applications as layered Apps (having multiple classes /files interacting with each other) by using multiple technologies

eg::
Struts ----------> from apache (old)
JSF ----------> from sunMs/Oracle corp (old)
Webwork ----------> from OpenSymphony (old)
Spring MVC (part of spring framework) ---> from Interface21/pivotol (best)
ADF ----------> from oracle corp (commercial)
and etc..

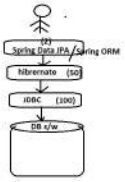
b) ORM frameworks

=>Provides abstaction on JDBC tehcnology and allows to develop objects based DB s/w independent Persistnece logic(DB opreations like insert,updste,delete and select) with out taking support of SQL Queries.

=> JDBC persistence logic is DB s/w dependent becoz of SQL Queries that it uses
=> ORM framworks O-R Mapping persistnce logic is DB s/w independent becoz it is not using any SQL queries

Hibernate ----------> from SoftTree/RedHat (best)
IBatis ----------> from Apache (third best)
OJB ----------> from apache
EclipseLink ----> from Eclipse org (second best)
TopLink ----------> from oracle corp
Open JPA ------> from sun Ms /oracle corp
Spring DATA JPA ---> from interface21/pivotal team | These two Internally uses
Spring ORM ---------> from interface21 /pivotal team | other ORM frameworks like hibernate and etc..

JPA :: JAva PErsistence API.



c) application frameworks

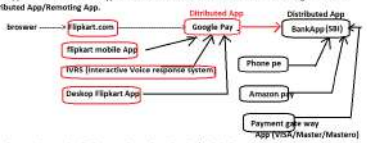
=>Provides abstraction on multiple java,jee technologies and simplifies all kinds of logics development(like presentation logic , b.logic, persistneceLogic and etc...) and also simplifies all kinds of application devleosopment (stand alone Apps, web applicaitons, Distributed Apps and etc..).

eg:: spring ----------> from Interface21 (pivotal team)
=>It is built on top of multiple java,jee technologies like JDBC, JMS,JndI,Servlet,jsp, EJB, RMI, JTA,Java mail and etc., and also on top of the frameworks like hibernate, IBaits and etc..

JMS:: JAva Messaging SErvice
Jndi :: Java Naming And Directoryinterface
jsp :: java server pages
EJB :: Enterprise Java Beans
RMI :: Remote Method Invocation
JTA :: Java TRansaction API
JPA: Java Persistnece API

d) WebSErvice /Distributed App Framework

=>The App that allows different types of Local or remote clients to the access the logics is called Distributed App/Remoting App.



=>browser interaction flipkart.com is web application(website)
=> flipkart.com talking to GooglePay/PhonePay/AmazonPay/PayTM/PaymentGateway and etc... comes under Distriobuted App..
=> GooglePay/PhonePe/and etc... talking to Bank App comes under distributed Applicaiton..

To develop Distributed App we need Distributed Technologies /frameworks.

=>RMI (Remote method invocation) (Technology)
=>EJB (Enterprise Java Beans) (Technology)
=>CORBA (Common Object Request Broker Architeture) (Technology) | (outdated)

RMI::Remote method invocation
EJB :Enterprise Java Beans
SOAP : Simple Object Access Protocol

=>WebServices (doing well)
--> SOAP based webservices (very old)
|-->jax-rpc , jax-ws , axis and etc.. (Frameworks to implement SOAP based webservices)
->Restful webSErvices (doing good)
(REpresentation STate)
|---> jax-RS (jersy) ,spring Rest, apache CFX and etc.. (Frameworks to implement Restfull webSErvices)

what is the difference b/w web applicaiton and distributed Application?

web application (website)	Ditributed App (Remoting Applicaiton)
(a) Generally the client browser s/w	a) Allows different types of client like web applications, Desktop Apps, mobile Apps, IOT devices, IVRS devices and etc..
(b) Here communicaiton model is request-response model	b) Here the communication model is method calls model. Here Client App calls method inorder to invoke methods of the server App (Distributed App)
c) Client is non-programmable Browser s/w.	c) Here Client App programmable Apps
d) these are Thin-Client -FatSErver Apps	d) These are Fat Client-Fat Server Application
e) Runs based on http/https protocols	e) Here different protocols will be used like SOAP, SOAP over Http, Iiop and etc..
f) Technologies and frameworks in java to develop web applications Servlet,jsp (technologies) struts, spring mvc,jsf, webwork(frameworks)	f) Technologies and frameworks in java to develop Distributed applications RMI,EJB (technologies) WebServices (Both SOAP and Restful) (frameworks)
g) eaxmples flipkark.com , amazon.in , nareshit.com and etc..	g) examples UPI Payment comps (gpay, phone pe,...) Payment gateways (VISA/Master/Mastreo,...) Payment Brokers (Paypal ,payuMoney,RazerPay ....) (these acts bride b/w e-commerce Apps and payment gateways) =>IPLScoreComp =>ICCScoreComp =>Wheather Forecast Comp =>BSE/NSE comp and etc..

Based on the mode of application devleopment we do there are two types frameworks::
a) Invasive Frameworks (Tightly coupled Frameworks)
b) Non-Invasite frameworks (Loosely Coupled Frameworks)

a| Invasive Frameworks (Tightly coupled Frameworks)

Applicaitons
=> while devleoping in these frameworks.. The applicaiton classes developed programmers having application logics should implement/extend Framework API interfaces/classes i.e our Application classes tighly coupled with underlying framework API , So we can not move our Applicaitons classes another frameworks for execution though the b.logics are reusable..
This behaviour is called invasive behaviour.

eg:: Struts 1.x

Struts API

```
public class BankService implements org.apache.struts.action.Action{
  ...
  ... //code for withdraw(),deposite() and etc..
  ...
}
```

b| Non-Invasive frameworks (Loosely Coupled Frameworks)

=> The Classes of Applicaiton development need not to implement framework API interfaces and classes.. i.e Application classes loosely coupled with framework API. So these Application classes can be moved other frameworks for execution - very easily.. This Behaviour is called Non-Invasive behaviour

eg:: spring , hibernate, webServices ,jsf , struts 2.x and etc..

```
public class BankSErvice{
  ...
  ... withdraw ,deposite and etc.. logics..
  ...
  ..
```

Mr.Rambabu (admin)
9866545966-whatsapp number
Naresh IT Online Team
support@nareshit.com

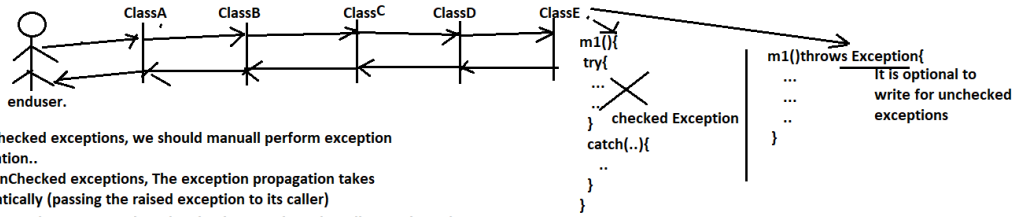
natarazjavaarena@gmail.com
FB group :: natarazjavaarena
url:: https://www.facebook.com/groups/388095825162910
(Faculty Info)

Spring 5.x with spring 2.x
==========

=>Course fee: 3500/-
=>Duration : 120 sessions (1 slot) (6pm)
70-75 sessions (2 slots) (6pm and 7am)
=>6/7 classes per week (daily 1:35 to 1:45mins)
=>Free realtime java tools
=>pre-requisite:: Core JAva oops/exception handling

Advantages of working with frameworks
==========

=> improves the productivity (Doing More work in less time)
=>Avoids the boilterplate code problem
=> Frameworks APIs (packages having classes,interfaces,enums, annotations) are designed based realtime scenarios.
=>Most the Framework API methods are designed to throw unchecked exceptions , So exception handling is optional and exception propagation is possible.



->For Checked exceptions, we should manuall perform exception propagation..
->For UnChecked exceptions, The exception propagation takes automatically (passing the raised exception to its caller)
=> Frameworks are vermuch to develop large scale and medium scale projects..

and etc..

Based on the kind of applications we develop there are 4 types of JAva frameworks
==========

a) Web Application frameworks
b) ORM frameworks (ORM :: Object relational mapping)
c) Application Frameworks
d) WebService/Distributed Application frameworks

a)Web application frameworks
==========

=>Developed based on servlet,jsp technologies to simplify MVC architecture java web application Development..
MVC (Model,View,Controller) is industry standard Architecture to develop web applications as layered Apps (having multiple classes /files interacting with each other) by using multiple technologies

eg::
Struts ----------> from apache (old)
JSF ----------> from sunMs/Oracle corp (old)
Webwork ----------> from OpenSymphony (old)
Spring MVC (part of spring framework) ---> from interface21/pivotol (best)
ADF ----------> from oracle corp (commercial)
and etc..

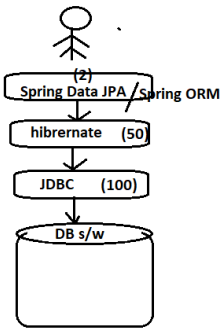
b) ORM frameworks
==========

=>Provides abstaction on JDBC tehcnology and allows to develop objects based DB s/w independent Persistnece logic(DB opreations like insert,update,delete and select) with out taking support of SQL Queries.

=> JDBC persistence logic is DB s/w dependent becoz of SQL Queries that it uses
=> ORM framworks O-R Mapping persistnece logic is DB s/w independent becoz it is not using any SQL queries

Hibernate ----------> from SoftTree/RedHat (best)
iBatis ----------> from Apache
OJB ----------> from apache
EclipseLink ------> from Eclipse org
Toplink ----------> from oracle corp
Open JPA -------> from sun Ms /oracle corp
Spring DATA JPA ---> from interface21/pivotal team | These two internally uses
Spring ORM --------> from interface21 /pivotal team | other ORM frameworks like hibernate and etc..

JPA :: JAva PErsistence API.



c) application frameworks

=>Provides abstraction on multiple java,jee technologies and simplifies all kinds of logics development(like presentation logic , b.logic, persistnecelogic and etc..) and also simplifies all kinds of application devleoopment (stand alone Apps, web applicaitons, Distributed Apps and etc..).

eg:: spring ----------> from interface21

=>It is built on top of multiple java,jee technologies like JDBC, JMS,Jndi,Servlet,jsp, EJB, RMI, JTA,Java mail and etc.. and also on top of the frameworks like hibernate, iBaits and etc..

JMS:: JAva Messaging SErvice
Jndi :: Java Naming And DirectoryInterface
jsp :: java server pages
EJB :: Enteprise Java Beans
RMI :: Remote Method Invocation
JTA :: Java TRansaction API
JPA: Java Persistnece API

Mr.Rambabu (admin)
9866545966-whatsapp number
Naresh IT Online Team
support@nareshit.com

natarazjavaarena@gmail.com
FB group :: natarazjavaarena
url:: https://www.facebook.com/groups/388095825162910
(Faculty Info)

Framework
==========

house
|--->option1 :: (purchase open land
start the construction of the house)
(Take care both common and specific activities of house constuction)
|--->option2 :: (purchase readymade house /flat from
builder)
(Take care only specific activities of house constuction becoz the builder
will take care of common activites .. Here builder is like framework)

Def1::
======
It is special installable software that built on the top of 1 or more technologies having ability to generate common logics of the Application dynamically by making programmer to supply only application specific logics..

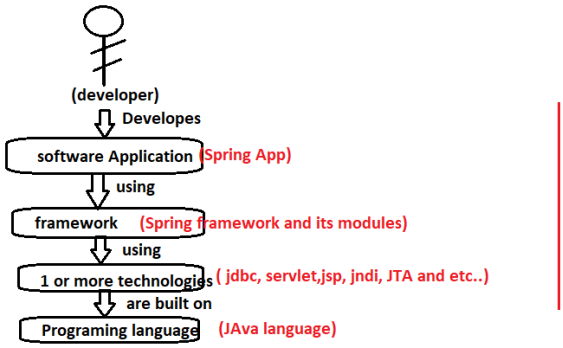
note:: Every framework is designed based technologies and design patterns (Best Practices)

=>Servlet,jsp web application is technology basaed web application development..
=>Spring MVC(part of spring) web application is framework based web application development.

Def2::
=====
It is special installable s/w that that provides abstraction(hiding details) on one or more technologies and simplities the applicaiton development..

note:: While working with languages and technologies we should take of care both common logics and application specific logics.. this improves burden on the programmer and also kills the productivity.

note:: While working with frameworks we should take care of only application specific specific logics.. This reduces burden on the programmer and also improves the productivity.



ADP :: Application Development Framework (from oracle corp)

JSF:: Java server faces

Example for java frameworks
==========
struts (old) ,jsf (old) , spring , hibernate , webServices , Webwork (not doing good) , ADP (costly) and etc..

JAva script toolkits/frameworks
==========
jquery, angularjs, angular, reactjs and etc..

Plain JDBC App (Technology Based Application)
==========
1) Load jdbc driver class to register JDBC driver s/w with DriverManager service (JDbc Driver s/w Actication) (Bridge/mediator b/w JAva App and Db s/w)
2)Establish the connection b/w Java App and DB s/w (Road b/w Java App Db s/w)
3) create JdbcStatement object (Vechile b/w Java App and DB s/w)

Common logics (same in all apps)

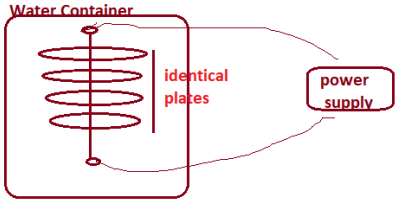
4) Send and execute SQL query in DB s/w using JDC Statement object (passing inputs)
5) Gather SQL query results from DB s/w and Process the results (gathering outputs)

Application speciifc logics (changes app to app)

6) Perform Exception Handling
7) Close jdbc objects including jdbc connection. (closing the road b/w java app and DB s/w)

Common logics (same in all apps)

=> The Technology based Application devleopment is having boilerplate code problem/Code Redundency Problem.
note:: the code that repeates across the multiple parts of Project/App either with no changes or with minor chanages is called boilerplate code



for videos::
https://www.youtube.com/watch?v=Bw3v1b3WjDM&list=PLVlQHNRLflP-wlUj1MAuLwiMekHpP-yQu

Spring JDBC App (Spring Application) (framework App)
==========
spring JDBC is part Spring framework which internally uses plain JDBC Technology and generates common logics of jdbc code

1) create JdbcTemplate (given by spring api) class obj (It takes care of commong logics of jdbc code)
2) Send and exceuteSQL query in DB s/w
3) gather resutls and process results

Applicaiton spcific logics

While working with frameworks , there is no boilerplate code problem. becoz framework itself generates common logics internally by taking support one or another technology.
the

Welcome to the world of Spring Programming

Who needs spring?

=>Every java devloper needs the knowledge spring .
=> No spring -->No job for java developer.

What is spring boot?

=>spring boot extension of spring
=> It is a methodology of spring programming

Title :: Spring with Boot

Are we going to learn spring boot ?
Ans) Yes

3-4 years of experience
can be claimed. using course
knowledge..

Spring framework modules

=>Spring core
=>Spring DAO/JDBC
=>Spring ORM
=>Spring Data
=>Spring Data JPA
=>Spring DATa MongoDB
=>Spring Tx
=> Spring MVC ,Spring AOP
=> Spring security
=>Spring Batch
=>Spring Social w ith Oauth
=>SpringMail
and etc..

Approaches of spring programming

(a) Using Xml Driven Configurations
(b) Using Annotation driven Configurations
(c) Using Java Config/100% Code configurations
(d) Using Spring Boot

10+ mini Projects
(As Indurstry standard Layered Apps)

Tools

Mave,Gradle, log4j, slf4j , junit, mockito, SVN, GIT, JasperReports/IReports , Agile-JIRA , Docker, Jenkins, and etc..

Pre-requisites:: Strong Core JAva (oops, Exception handling)

FB group name :: natarazjavaarena
https://www.facebook.com/groups/388095825162910
email Id: natarazjavaarena @gmail.com

Admin details

Mr.Rambabu
9866545966
Naresh IT online Team =support@nareshit.com

Course Fee:: 3500/- 6pm
Duration :: 120 Sessions (Single slot)
75 sessions (two slots)
(6pm ,7am)
weekly :: 6 sessions /7 sessions

Spring is Framework

JAva Learning is all about
a) Language :: Core JAva
b) Technologies :: Adv.java (jdbc,jndi, servlet,jsp and etc..)
c) frameworks :: spring,hibernate,JSF , WebSErvices

What is the diffrence b/w programming language , software technology and framework?

Programming language (It is lik raw material -->ricc granuals, wheat granuals and etc..)

=> It is directly installable s/w acting as raw material by providing basic features that are required to develop software applications

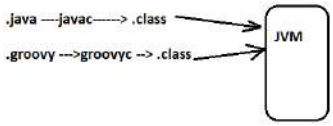
=> It defines the syntaxes(rules) ,semantics(structure) of the programming by supplying compilers and interpeters.

=>Programming languages are base to create other technologies, frameworks, Operating systems , Tools , Db s/ws and etc..

eg:: c,c++, java , c# , html, vb, vc++ and etc..

JVM based languages

java
groovy
Go
Scala
swift
kotlin
and etc..



Software Technology

=> It is a software specification providing set of rules and guidelines in the for api to develop implementation s/ws.
=>Software tehcnology is not installable.. but Software technology based implementation softwares of installable or arrangable.. working with these implementation sofwares nothing but working with software technologies.

JdbcTechnology (Sun Ms)
[Contains rules and guideliens
in the form jdbc api/pakcages)
|----->Vendor1 (oracle cop) --> jdbc driver s/w for oracle
|----->Vendor2 (Devx) ---> jdbc driver s/w for mysql
|-----> Vendor3 (Enterprise DB) ----> jdbc driver s/w for postgresql

(Implemetation s/w for jdbc technology as jdbc driver s/ws)

=>JDBC Technology is not installble.. but the jdbc technology based jdbc driver s/ws are installable or arrangable..
=>Working with JDBC driver s/w .. is nothing but working JDBC Technology..

Two types of S/w Technologies

Open Technologies

HEre the techology rules and guidelines are open to all software vendor companies to create implementation sofwares..

eg:: JDBC, Servlet, Jsp and etc..
note:: All java ,JEE technologies are open technologies

propiertery Technologies

HEre the technology rules and guidlines are specific to one vendor and only that vendor allowed to create implementation softwares.. and other vendors are not allowed..

eg:: Microsoft technologies like asp.net