Bean Definition Inheritance

This allows you to reuse various common properties of various beans.

Ex:

```
        Class hello{
        Int x;
        Int y;
        String str;
        String msg;
        // setter methods
        }
```

Without Bean Definition inheritance

```
Get Bean("h1");
<bean id="h1" class="com.jtc.hello">
        <property name="x" value="10"/>
        <property name="y" value="20"/>
        <property name="str" value="Hai Guys"/>
        <property name="msg" value="hello guys"/>
</bean>
< bean id="h2" class="com.jtc.Hello">
        <property name="x" value="10"/>
        <property name="y" value="30"/>
        <property name="str" value="Hai Guys"/>
        <property name="msg" value="hello guys"/>
</bean>
```

With Bean Definition inheritance

```
<bean id="baseHello" class="com.jtc.hello" abstract="true">
        <property name="x" value="10"/>
        <property name="str" value="Hai guys"/>
</bean>
<bean id="h1" parent="basehello">
        <property name="y" value="20"/>
        <property name="str" value="Hai guys"/>
</bean>
<bean id="h2" parent="basehello">
        <property name="y" value="30"/>
        <property name="str" value="Hai guys"/>
</bean>
```

Externalizing Bean Properties

This allows you to place the bean property values in a separate property file instead of hard-coding the values in spring configuration Document.

Steps

1. Write one or more property files and place them in project folder directly
   oracle.properties
   oracle.dc=some driove class
   oracle.url=some url
2. Refer the keys in xml as follows.
   <property name="dc" value="${oracle.dc}"/>
3. To Externalize Bean properties,you hava to register property Placeholder Configurer bean in spring configuration Document as follows.

```
<bean class="org.springframework.beans.factory.config.propertyPlaceholderConfigurer">
        <property name="location">
        <list>
        <value>oracle.properties</value>
        <value>mysql.properties</value>
        <value>common. properties </value>
        </list></property>
</bean>
```

Jtc22: Files required

Jtc22.java	TestBean.java
Data source.java	Common.properties
Jtcindia.xml	Mysql.properties
Oracle.properties	

Jtc 22.java

```
Package com.jtc india.spring;
Import org.springframework.context.ApplicationContext;
Import org.springfromework.Context.Support.ClassPathXml ApplicationContext;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit         : www.jtcindia.org
**/
Public class Jtc22{
Public static void main(String[] args) {
ApplicationContext ctx=new ClassPathXmlApplicationContext("jtcindia.xml");
TestBean tb=(TestBean)ctx.getBean("test");
tb.show();
}
}
```

TestBean.java	DataSource.java

Package com.jtcindia.spring;
Import javax.annotation.Resource;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit : www.jtcindia.org
**/
Public class test bean{
@Resource(name="oracleDS")
DataSource oracleDateSource;
@Resource(name="mysqlDS")
DateSource mysqlDataSource;
Public void show(){
System.out.println(oracleDataSource);
Syste.out.println(mysqlDataSource);
}
}
	Package com.jtcindia.spring;
	/*
	*@Author: Som Prakash Rai
	*@Company : java Training Center
	*@ visit : www.jtcindia.org
	**/
	Public class Data Source {
	Private String driver Class;
	Private String url;
	Private String username;
	Private String password;
	Private int min;
	Private int max;
	Private inttimeout;
	Private string getDrivetClass() {
	Return drivwrClass;
	}
	//Setters and getters
	Public string to string(){
	Return
	""+driverClass+"\n"+username="\n"+password+"\n"
	+min+"\n"+max+"\n"+timeout;
	} }

Common.properties	

Jtcindia.xml

```
<beans…>
<context:annotation-config/>
<bean class=”org.springframework.beans.factory.config.propertyPlaceholder Configurer”>
<property name=”location”>
<list>
<value>oracle.properties</value>
<value>mysql.propertis</value>
<value>commone.properties</value>
</list>
</property>
</bean>
<bean id=”baseDS” class=”com.jtcindia.spring.DateSource” abstract=”true”>
<property name=”password” value=”${jtc.pw}”/>
<property name=”min value”${jtc.min}”?>
<property name=”timeout”value=”${jtc.timeout}”/>
</bean>
<bean id=”oracleDS” parent=”baseDS”>
<property name=”driverClass” value=”${oracle.pw}”/>
<property name=”url value”${ oracle.min}”?>
<property name=”username”value=”${ oracle.un}”/>
</bean>
<bean id=”mysqlDS” parent=”baseDS”>
<property name=”driverClass” value=”${mysql.dc}”/>
<property name=”url value”${ mysql.url}”?>
<property name=”username”value=”${ mysql.un}”/>
</bean>
<bean id=”test” class=”com.jtcindia.spring.TestBean”/>
</beans>
```

Accessing Message bundles

- When develop any web application you will write various message bundles to support multiple languages.
- When you want to access properties from message Bundles with the help of spring, you need to do the following steps

  Write one or more message bundles to support multiple languages.

  Basename LanguageCode countryCode.properties
  Basename LanguageCode properties

Messages.properties(English)
Un.required=Username is Required(English).
Errors.required={0} is Required (English)
Erroes.range={0} length must be between {1} and {2} (English)

Messages hi.Properties (Hindi)
Un.required=Username is Required (hindi).
Errors.required={0} is Required (hindi)
Erroes.range={0} length must be between {1} and {2} (hindi)

Access the properties of message bundles using following methods of Application Context.

```
getMassage(String key,object [] args,Locale );
getMassage(String key,Object [] args, String defaultMassage, Locale);
```

Ex:

```
String m 1=ctx.getMessage(“un.required”,null,new locale(“h1”));
String m 2=ctx.getMessage(“errors.required”,new object [] {Email id”} new Locale(“en”))’
String m 3=ctx.getMessage(“errors.range”new object[] username”,”5”,”10”)new Locale(“hi”));
String m 4=ctx.getMessage(“pw.required”,null,new Locale(“hi”));
String m 5=ctx.getMessage(“pw.required”,null,”password is Required”,new locale(“hi”));
```

Register the bean called Resource BundleM essageSource with spring container by specifying the following in spring configuration Document.

```
<bean id=”messageSource” class=”org.sprinngframework.context.support.Resource Bundle
Messagesource”>
        <property name=”basename” value=”message”/>
        </bean>
```

Jtc23: Files required

Jtc23.java	Hello.java
Messages.properties	Messages hi.properties
Jtcindia.xml	

Jtc23.java

```
Package com.jtc india.spring;
Import org.springframework.context.ApplicationContext;
Import org.springfromework.Context.Support.ClassPathXml ApplicationContext;
Public class Jtc23{
Public static void main(String[] args) {
ApplicationContext ctx=new ClassPathXmlApplicationContext(“jtcindia.xml”);
TestBean tb=(TestBean)ctx.getBean(“TestBean”);
tb.show English();
tb.show Hindi();
}
}
```

Test Bean.java

```
Package com.jtc india.spring;
Import java.util.Locale;
Import org.springframework.beans.factory.annotation.autowired;
Import org.springframework.context.ApplicationContext;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
Public class Test Bean{
@Autowired
ApplicationContext ctx=null;

Void showEnglish(){
String m1=ctx.getMessage("username.required",null,null);
System.out.println(m1);
String m2=ctx.getMessage("password.required"new object[]{password},null};
System.out.println(m2);
String m3=ctx.getMessage("errors.min",new object[]{"username","5"),null)"
System.out.println(m3);
String m4=ctx.getMessage("errors.min",new object[]{"username","10"),null)"
System.out.println(m4);
String m5=ctx.getMessage("errors.min",new object[]{"username","6","12"),null)"
System.out.println(m5);
String m6=ctx.getMessage("errors.min",new object[]{"username","6","12"),null)"username range is not
ok.",null);
System.out.println(m6);
}

Void show Hindi(){
String m1=ctx.getMessage("username.required",null,null Locale ("hi");
System.out.println(m1);
String m2=ctx.getMessage("password.required"new object[]{password},null Locale ("hi");
System.out.println(m2);
```

```
String m3=ctx.getMessage("errors.min",new object[]{"username","5"),null Locale ("hi");
System.out.println(m3);
String m4=ctx.getMessage("errors.min",new object[]{"username","10"),null Locale ("hi");
System.out.println(m4);
String m5=ctx.getMessage("errors.min",new object[]{"username","6","12"),null Locale ("hi");
System.out.println(m5);
String m6=ctx.getMessage("errors.min",new object[]{"username","6","12"),null)"username range is not
ok.",null Locale ("hi");
System.out.println(m6);
}
```

Messages.properties	**Jtcindia.xml**

Username.required=Username is Required.
Password.requierd={0} is Requierd.
Error.min={0} must be min : {1}
Error.max={0}must be max : {1}
Errors.range={0} must be between : {1} and {2}
 	
Messages hi properties
Username.required=username is Required. {IN HINDI}
Error.min={0} must be min : {1} {IN HINDI}
Error.max={0}must be max : {1} {IN HINDI}
Errors.range={0} must be between : {1} and {2} {IN HINDI}
	<beans…>
	<context:annotation-config/>
	
	<bean id="messageSource"class="org.springframework.
	context.support.Resource B undleMessage Source">
	<property name="basename" value="message"/>
	</bean>
	
	<bean id="test Bean" class="com.jtcindia.spring.
	TestBean"/>
	</beans>

Using property Editors

- Property Editors are used to edit the property values.
- You can do the following using property Editors
  Convert the value from one data type to another data type
  Convert the value form one format to another format.

Ex

```
Class Hello {
Int x;
Double y;
String str;
….
}

<bean id="h" class="….Hello">
        <property name="x"value="99"/>
        <property name="y"value="99.99"/>
        <property name="str"value="hello guys"/>
</bean>
```

For simple data types, there are many built-in property editors avaiLable.
For your custom data types, you have to develop custom property editors

Steps to develop and Register custom property editors
Write your own editor class by extending property Editor Support class avaiLable in java. Beans package.

Override the following method in custom Editor class
Public void setAsText(String txt)

Write the required conversion logic inside setAsText() method.

Register the Custom property editors with spring container as follows.

```
<bean class="org.springframework.beans.factory.config.customEditorConfigurer">
<property name="costomEditors">
        <msp>
                <entry key ="java.util.List" value="com.jtcindia.spring.listEditor"/>
                <entry key=" com.jtcindia.spring.Fee" value="com.jtcindia.spring.FeeEditor"/>
        </map>
</property>
</bean>
```

Jtc24: Files required

Jtc24.java	student.java
Student Id.java	Student IdEditor.java
Fee.java	FeeEditor.java
ListEditor.java	Jtcindia.xml

Jtc24.java

```
Package com.jtcindia.spring;
Import org.springframework. context..application.Context;
Import org.springframework.context.support.*;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit        : www.jtcindia.org
**/
Public class Jtc24 {
Public static void main (string [] args} {
Application Context ctx=new classPathXmlApplicationContext("jtcindia.xml");
Student stu=(student) ctx.getBean("Student");
System.out.println(stu.getsid().getBid());
System.out.println(stu.getsid().getsid());
System.out.println(stu.getSname());
System.out.println(stu.getPhone());
System.out.println(stu.getFee().getFeeBal());
System.out.println(stu.getFee().getFeepaid());
System.out.println(stu.getFee().getTotalFee());
System.out.println(stu.getEmails());
System.out.println(stu.getPhones());
}
}
```

Student.java

```
Package com.jtcindia.spring;
Import java.util.list;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit       : www.jtcindia.org
**/
Public class student {
StudentId sid;
String sname;
Long phone;
Fee fee;
List<string> emails;
List<Long> phones;
// Seters and Getters
}
```

Student IDEditor.java

```
Package com.jtcindia.spring;
Import java.beans.propertyEditorSupport;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit       : www.jtcindia.org
**/
Public class student IDEditor extends
Property EditorSupport{
Public void setAsText(String txt){
String p1=txt.substring(o,3);
String p2=txt.substring94);
StudentID sid=new
studentID(integer.parseInt(p2),p1);
This.setValue(sid);
}
}
```

Student ID.Java

```
Package com.jtcindia.springng.
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit       : www.jtcindia.org
**/
Public class studentID
String bid;
Int sid;
```

Fee.java

```
Package com.jtcindia.springng.
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit       : www.jtcindia.org
**/
Public class fee{
Double total fee;
Double feepaid;
```

Public student ID (int sid, string bid) {
This.sid=sid;
This.bid=bid;
}
//setters and getters
}
	Double feepaid;
	Double feebal;
	Public Fee(double totalFee, double feepaid, double feeBal){
	Super();
	This.totalFee=totalFee;
	This.feepaid=feepaid;
	This.feeBal=feeBal;
	}
	//Setters and getters
	}

FeeEditor.java	List Editer.java

Package com.jtcindia.spring
Import java.beans.propertyEditorSupport;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit : www.jtcindia.org
**/
Public class feeEditor extends property Editor Suport{
Public void set As Text (String txt){
String str[]=txt.split(",")'
Fee fee=new Fee(Double.parseDouble(str[0]).Double.parseDouble (str[1]), double parse Double (str[2]);
This.set Value(fee);
}}
	Package com.jtcindia.spring;
	Import java.beans.property EditorSupport;
	Import java.util.*;
	/*
	*@Author: Som Prakash Rai
	*@Company : java Training Center
	*@ visit : www.jtcindia.org
	**/
	Public class feeEditor extends property Editor Suport{
	Public void set As Text (String txt){
	String str[]=txt.split(",")'
	List list=Arrays.asList(str);
	This.setValue(list);
	}
	}

Jtcindia.xml

```
<beans…>
<context:annotation-config/>
<bean class="org.springframework.beans.factory.config.customEditorConfigurer">
<property name="customEditors">
<map>
<entery key="com.jtcindia.spring.studentID">
<value>com.jtcindia.spring.StudentIDEditor</value>
```

```
</entry>
<entry key=”com.jtcindia.spring.fee”>
<value>com.jtcindia.spring.FeeEditor</value>
</entery>
<entery key=”java.util.list>
<value>com.jtcindia.spring.listEditor</value>
</entry>
</map>
</property>
</bean>
<bean id=”student” class=”com.jtcindia.spring.student”>
<property name=”sid” valu=”B1-101”/>
<property name=”sname” value=”somprakash”/>
<property name=”phone”value=”9999”/>
<property name=”fee” value=”100000,50000,50000/>
<property name=”emails” value=aa@jtc.com,bb@jtc.com,cc@jtc,com.dd@jtc.com/>
<property name=”phones”>
<list>
<value>1111</value>      <value>2222</value>        <value>3333</value>
</list>
</property>      </bean>        </beans>
```

Events and Listeners

- Event Handling mechanism is provided with spring container called application Context.
- Application Context container handles two types of Events:
  1. Standard or system events
  2. Custom or application Events.

Standard Events

- Spring provides the following standard events:
  1. Context Started Event
  2. Context RefreshedEvent
  3. Context Stopped Event
  4. ContextClosedEvent
- Context started Event will be published when Application Context is started using start() method on Configurable Application Context interface.
- ContextRefreashed Event will be published when application Context is initialized or refreshed using refresh() method on configurable ApplicationContext interface.
- Context Stopped Event will be published when Application Context is stopped using stop() method on Configurable Application context interface.
- Context Closed Event will be published when application Context is closed using close() method on configurable Application Context interface.

Application Events

You can also create and publish you own application events.
Ex
FeePaidEvent,
NewBatchEvent
Etc.
Application Event handling is provided through
Application Event class
Application Listener interface.

Publishing Application Events:

Writing and Publishing Application Events
Writing Application Events:

- Write your own Event class by extending Application Event class.
- Declare the variables to hold the information about the Event.
- Your event class should contain two arguments constructor.
  First argument is information about the event.
  Second argument is source of the event.

Publishing Application Events:
```
Batch Bat=new Batch(…….);
BatchEvent be=new BatchEvent (bat,this);
Ctx.publishEvent(be);
```

Writing and Registering Application Listeners
Writing Application Listener

- Write you own listener class by implementing Application Listerer interface.
- Override the following method in Customer listener class
  Public void onApplication Event (Application Event event)

Registering Application Listener:
Configure that Listener class in spring context xml.
```
<bean class=”com.jtc.JTCListener1”/>
<bean class=”com.jtc.jtcListener2”/>
```

Using Event specific Listeners

JTClistener 1.java

```
        Package com.jtcindia.spring;
        Import org.springframework.context.ApplicationListener;
        Public class JTCListener 1 implements Application listener<BatchEvent> {
                Public void on ApplicationEvent (BatchEvent be) {{
                        System.out.println("in onApplicationEvent()");
                        System.out.println("Batch event is raised");
                        System.out.println("be.getBatch());
                }
        }
```

JTClistener2.java

```
        Package com.jtcindia.spring;
        Import org.springframework.context.ApplicationListener;
        Public class JTCListener 2 implements Application listener<WorkshopEvent> {
                Public void on ApplicationEvent (WorkshopEvent we) {{
                        System.out.println("in onApplicationEvent()");
                        System.out.println("WorkshopEvent is raised");
                        System.out.println("be.get Workshop());
                }
        }
```

JTCindia.xml

```
        <bean class="com.jtcindia.spring.JTCListener1"/>
        <bean class="com.jtcindia.spring.JTCListener2"/>
```

Jtc25: Files required

Jtc25.java	Batch.java
Batch Event.java	workshop.java
workshop.java	JTCListener1.java
JTCListener2.java	JTCListener3.java
JTCManager.java	Jtcindia.xml

Jtc25.java

```
Package com.jtcindia.spring;
Import org.springframework.context.ApplicationContext;
Import org.springfromework.Context.Support.ClassPathXml ApplicationContext;
Public class Jtc25{
Public static void main(String[] args) {
ConfigurableApplicationContext ctx=new ClassPathXmlApplicationContext("jtcindia.xml");
Ctx.start();
JTCManager jtc=(JTCManager)ctx.getBean("jtc")
Batch b=new Batch(B-16”,”28-apr-2010” 6.30-8.30”.150);
Jtc.addBatch(b);
Workshop Ws=new workshop(“web services”50000,99);
Jtc.addworkshop(ws);
Ctx.stop():
Ctx.close();
}
}
```

Batch.java

```
Package com.jtcindia.spring;
Public class Batch{
String bid;                string sp;
String timings;            int nos;
Public Batch(String bid, string sp, string timings, int nos)
{
This.bid=bid;      this.sp=sp;
This.timings=timings;    this.nos=nos;
}
Public string toString(){
String msg=”JTC anno… New Batch”’;
Msg=msg+”\n batch id”+bid;
Msg=msg+”\n start Data”+bid;
Msg=msg+”\n Timings”+timings;
Msg=msg+”\n No of seats”+nos;
Return msg;
}
}
```

Batch Event.java

```
Package com.jtcindia.spring
Import org.springframework.context.ApplicationEvent;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
Public class BatchEvent extends Application Event{
Batch batch;
Public BatchEvent(Object source , Batch batch) {
Super(source);
This.batch=batch;
}
Public batch getBatch() {
Return batch;
}
}
```

Workshop.java

```
Package com.jtcindia.spring
Import org.springframework.context.ApplicationEvent;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
Public class workshop {
String topic;                        double fee;
Int nos;
Public workshop(string topic, double fee, int nos) {
This.topic=topic;           this.fee=fee;
This.nos=nos;
}
Public string to String() {
String msg=”jtc Anno…new workshop”;
```

WorkshopEvent.java

```
Package com.jtcindia.spring
Import org.springframework.context.ApplicationEvent;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
Public class BatchEvent extends Application Event{
Workshop ws;
Public workshopEvent(object source, workshoup ws) {
Super (source);
This.ws=ws;
}
Public workshop get Workshop() {
Return ws;
}
Msg=msg+"\n topic"+topic;
Msg=msg+"\n fee"+fee;
Msg=msg+"\n No of seats"+nos;
Return msg;
}
```

```
}
```

JTCListener1.java

```
Package com.jtcindia.spring
Import org.springframework.context.*;
Import org.springframework.context.*;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit        : www.jtcindia.org
**/
Public class JTCListener 1 implements
Application Listener {
Public void onApplication Event (Application Event event)
{
If(event instance of Batch Event) {
Batch Event Batch Event=(Batch Event)event;
Batch bat=batchEvent.getBatch();
System.out.println(bat);
}
Else if(event instanceof WorkshopEvent) {
workshopEvent weEvent=Workshop Event)event;
workshop ws=ws Event.get Workshop();
system.out.println(ws);
}else{
System.out.println(event);
}}}
```

JTCListener1.java

```
Package com.jtcindia.spring
Import org.springframework.context.*;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit        : www.jtcindia.org
**/
Public class JTCListener 2 implements
ApplicationListener<Batchevent> {
Public void onApplication Event(BatchEvent event) {
System.out.println("JTCListener2 onApplication Event()");
System.out.printin("Batch event is raised");
BatchEvent be=(BatchEvent)event;
System.out.println(be.getbatch();
}
}
```

JTCListener3.java

```
Package com.jtcindia.spring
Import org.springframework.context.*;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit        : www.jtcindia.org
**/
Public class JTCListener3 implements
ApplicationListener<WorkshopEvent> {
Public void onApplication Event(WorkshopEvent event) {
System.out.println("JTCListener3 onApplication Event()");
System.out.printin("WorkshopEvent is raised");
BatchEvent be=( WorkshopEvent)event;
System.out.println(be.getWorkshop();
```

JTCManager.java

```
Package com.jtcindia.spring
Import org.springframework.context.*;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit        : www.jtcindia.org
**/
Public class JTCManager implements
Application ContextAware{
ApplicationContext ctx=null;
Public void add Batch(Batch bat) {
Ctx.publishEvent(new BatchEvent(this.bat));
}
Public void addWorkshop (workshop ws){
```

}
}
	Ctx.publishEvent(new WorkshopEvent(this,ws));
	}
	Public vaid setApplication Context(ApplicationContext ctx){
	This.ctx=ctx;
	} }

Jtcindia.xml

```
<?xml version =”1.0” encoding=”UTF-8”?>
<beans xmlns=http://www.springgramework.org/schema/beans
        Xmlns:p=http://www.springframework.org/schema/p
        Xmlns:context=http://www.springframewok.org/schema/context
        Xsi:schemaLocation=”http://www.springframework.org/schema/beans
        Xmlns:xsi=”http://www.w3.org/2001/xmlschema-instance:
http://www.springframework.org/schema/beans/spring-beans-3.0.xsp
http://www.springframework.org./schema/context http://www.springframework.org/schema/context/
spring context-3.0.xsp”>

                <bean id=”jtc” class=”com.jtcindia.spring.jtcManager”/>
                <bean class=”com.jtcindia.spring.Jtclistener1”/>
                <bean class=”com.jtcindia.spring.Jtclistener2”/>
                <bean class=”com.jtcindia.spring.Jtclistener3”/>
</beans>
```

@inject

Jtc26: Files required

Jtc26.java	A.java
B.java	Hello.java
Jtcindia.xml	

Jtc26: Files required

```
Package com.jtcindia.spring;
Import org.springframework.context.ApplicationContext;
Import org.springfromework.Context.Support.ClassPathXml ApplicationContext;
/*
*@Author: Som Prakash Rai
```

```
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
Public class Jtc26{
Public static void main(String[] args) {
ConfigurableApplicationContext ctx=new ClassPathXmlApplicationContext("jtcindia.xml");
Hello h=(Hello)ctx.getBean("hello");
h.show();
}
}
```

A.java	B.java

Package com.jtcindia.spring;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit   : www.jtcindia.org
**/
Public class A {
Private int a;   //s.i
Private string msg;   //s.i
Public void seta(int a) {
This.a=a;
}
Public void set mag(string msg){
This.msg=msg;
}
Public string to string() {
Return ""+a+"\t"+msg;
}
}
	Package com.jtcindia.spring;
	/*
	*@Author: Som Prakash Rai
	*@Company : java Training Center
	*@ visit : www.jtcindia.org
	**/
	Public class B {
	Private int b; //c.i
	Private string str; //c.i
	Public B(int b, string str) {
	This.b=b;
	This.str=str;
	}
	Public string to string(){
	Return""+b+"\t"+str;
	}
	}

Hello.java.

```
Package com.jtcindia.spring;
Import javax.inject.inject;
Import org.springfromework.beans.factory.annotation.Qualifier;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
Public class Hello{
@Inject
```

```
Private A aobj;

@Inject
@@ualifier("bo@")
Private B bobj;
Public void show(0{
System.out.println(aobj);
System.out.println(bobj);
} }
```

Jtcindia.xml

```
<?xml version ="1.0" encoding="UTF-8"?>
<beans xmlns=http://www.springframework.org/schema/beans
       Xmlns:p=http://www.springframework.org/schema/p
       Xmlns:context=http://www.springframewok.org/schema/context
       Xsi:schemaLocation="http://www.springframework.org/schema/beans
       Xmlns:xsi="http://www.w3.org/2001/xmlschema-instance:
http://www.springframework.org/schema/beans/spring-beans-3.1.xsp
http://www.springframework.org./schema/context http://www.springframework.org/schema/context/
spring context-3.1.xsp">
<context:annotation-config/>
<bean id="ao"
Class="com.jtcindia.spring.A"
P:a="99"
P:msg="A99"/>
<bean id="bo1" class="com.jtcindia.spring.B">
<constructor-are value ="77"/>
<constructor-are value="B77"/>
</boan>
<bean id="bo2" class="com.JTCindia.spring.B">
<constructor-arg value="66"/>
<constructor-arg value=B66"/>
</bean>
<bean id="hello" class="com.jtcindia.spring.Hello"/>
</beans>
```

Jtc27: Files required

Jtc27.java	Hai.java
Hello.java	jtcconfig.xml

Jtc27.java.

```
Package com.jtcindia.spring;
Import org.springfromework.context.annotation.AnnotationConfigApplicationContext;
/*
```

```
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit        : www.jtcindia.org
**/
Public class Jtc27 {
Public static void main (string [] args) {
AnnotationConfigApplicationContext ctx=new AnnotationConfigApplicationContext(JTCConfig.class);
System.out.println("Now Spring container is Ready");
Hello hello1=(Hello) ctx.getBean (Hello.class);
Hello1.show();
Hello hello2=(Hello)ctx.getBean("hello");
Hello2.show();
System.out.println(hello1==hello2);
Hai hai1=(Hai)ctx.getBean(Hai.class);
Hai1.show();
Hai hai2=(Hai)ctx.getBean("hai");
Hai2.show();
System.out.println(hai1==hai2);
}
}
```

Hai.java.

```
Package com.jtcindia.spring;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit        : www.jtcindia.org
**/
Public class Hai {
Public Hai (){
System.out.println("Hai-D.C");
}
Public void show(){
System.out.println("Hai-show()");
}
}
```

Hello.java.

```
Package com.jtcindia.spring;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit        : www.jtcindia.org
**/
Public class Hello {
Public Hello (){
System.out.println("Hello -D.C");
}
Public void show(){
System.out.println("Hello -show()");
}
}
```

JTCConfig.java.

```
Package com.jtcindia.spring;
//IMPORT HERE
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit        : www.jtcindia.org
```

```
**/
@configuration
Public class JTCConfig {
             Public JTCConfig () {
                     System.out.println("spring Container is now getting Ready");
           }
@Bean (name="hello")
Public Hello getHello(){
         System.out.println("getHello()");
          Return new Hello();
}
@bean(name="hai")
@scope("prototype")
Public Hai getHai(){
         System.out.println("getHai()");
         Return new Hai();
} }
```

Jtc28: Files required

Jtc28.java	Hai.java
Hello.java	jtcconfig.xml

JTCConfig.java.

```
Package com.jtcindia.spring;
//IMPORT HERE
@configuration
Public class JTCConfig {
             Public JTCConfig () {
                     System.out.println("spring Container is now getting Ready");
           }
@Bean (name="hello")
Public Hello getHello(){
         System.out.println("getHello()");
          Return new Hello();
}
@bean(name="hai")
@Lazy
Public Hai getHai(){
         System.out.println("getHai()");
         Return new Hai();
} }
```

Jtc29: Files required

Jtc29A.java	Jtc29B.java
Jtc29C.java	Hai.java
HaiConfig.java	Hello.java
helloConfig.java	

Jtc29A.java.

```
Package com.jtcindia.spring;
Import org.springfromework.context.annotation.AnnotationConfigApplicationContext;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
Public class Jtc29A {
Public static void main (string [] args) {
AnnotationConfigApplicationContext ctx=new
AnnotationConfigApplicationContext(HelloConfig.class,HaiConfig.class);
System.out.println("Now Spring container is Ready");
Hello hello=(Hello) ctx.getBean (Hello.class);
Hello.show();
Hai hai =( hai)ctx.getBean("hai.class");
hai.show();
}
}
```

Jtc29B.java.

```
Package com.jtcindia.spring;
Import org.springfromework.context.annotation.AnnotationConfigApplicationContext;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
Public class Jtc29B {
Public static void main (string [] args) {
AnnotationConfigApplicationContext ctx=new AnnotationConfigApplicationContext();
Ctx.register(helloConfig.class);
Ctx.register(HaiConfig.class);
Ctx.refresh();
System.out.println("Now Spring container is Ready");
Hello hello=(Hello) ctx.getBean (Hello.class);
Hello.show();
Hai hai =( hai)ctx.getBean("hai.class");
hai.show();
}
}
```

Jtc29B.java.

```
Package com.jtcindia.spring;
Import org.springfromework.context.annotation.AnnotationConfigApplicationContext;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit         : www.jtcindia.org
**/
Public class Jtc29B {
Public static void main (string [] args) {
AnnotationConfigApplicationContext ctx=new AnnotationConfigApplicationContext();
Ctx.register(com.jtcindia.spring");
System.out.println("Now Spring container is Ready");
Hello hello=(Hello) ctx.getBean (Hello.class);
Hello.show();
Hai hai =( hai)ctx.getBean("hai.class");
hai.show();
}}
```

Hai.java.

```
Package com.jtcindia.spring;
Public class Hai {
Public Hai (){
System.out.println("Hai-D.C");
}
Public void show(){
System.out.println("Hai-show()");
}
}
```

HaiConfig.java

```
Package com.jtcindia.spring;
Import org.springframework.context.annotation.Bean;
Import
org.springframework.context.annotation.Configuration;
@Configuration
Public class HaiConfig {
@Bean(name="hai"
Public Hai getHai(){
        System.out.println("getHai()");
        Return new Hai();
} }
```

Hello.java.

```
Package com.jtcindia.spring;
Public class Hello {
Public Hello (){
System.out.println("Hello -D.C");
}
Public void show(){
System.out.println("Hello -show()");
}
}
```

Hello Config.java

```
Package com.jtcindia.spring;
Import org.springframework.context.annotation.Bean;
Import
org.springframework.context.annotation.Configuration;
@Configuration
Public class Hello Config {
@Bean(name=" Hello"
Public Hai get Hello (){
        System.out.println("get Hello ()");
        Return new Hello ();
} }
```

Jtc30: Files required

Jtc30.java	Hai.java
HaiConfig.java	Hello.java
helloConfig.java	

Jtc30.java.

```
Package com.jtcindia.spring;
Import org.springfromework.context.annotation.AnnotationConfigApplicationContext;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
Public class Jtc30 {
Public static void main (string [] args) {
AnnotationConfigApplicationContext ctx=new AnnotationConfigApplicationContext(HelloConfig.class);
System.out.println("Now Spring container is Ready");
Hello hello=(Hello) ctx.getBean (Hello.class);
Hello.show();
Hai hai =( hai)ctx.getBean("hai.class");
hai.show();
}
}
```

HelloConfig.java

```
Package com.jtcindia.spring;
Import org.springframework.context.annotation.Bean;
Import org.springframework.context.annotation.configuration;
Import org.springframework.context.annotation.Import;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
@Configuration
@import(Haiconfig.class)
Public class HelloConfig {

@Bean(name="hello")
Public Hello getHello(){
        System.out.println("getHello()");
        Return new Hello();                          } }
```

Jtc31: Files required

Jtc31.java	A.java
B.java	Hello.java
JTCConfig.java	Jtcindia.xml

Jtc31.java.

```
Package com.jtcindia.spring;
Import org.springfromework.context.annotation.AnnotationConfigApplicationContext;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit         : www.jtcindia.org
**/
Public class Jtc31 {
Public static void main (string [] args) {
AnnotationConfigApplicationContext ctx=new AnnotationConfigApplicationContext(JTCConfig.class);
Hello h=(Hello) ctx.getBean (Hello.class);
H.show();
}
}
```

A.java

```
Package com.jtcindia.spring;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit         : www.jtcindia.org
**/
Public class A {
Private int a;
Private string msg;
Public void seta(int a) {
This.a=a;
}
Public void set mag(string msg){
This.msg=msg;
}
Public string to string() {
Return “”+a+”\t”+msg;
}
}
```

B.java

```
Package com.jtcindia.spring;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit         : www.jtcindia.org
**/
Public class B {
Private int b;
Private string str;
Public B(int b, string str) {
This.b=b;
This.str=str;
}
Public string to string(){
Return””+b+”\t”+str;
}
}
```

Hello.java

```
Package com.jtcindia.spring;
Import javax.inject.inject;
Import
org.springfromework.beans.factory.annotation.Q
ualifier;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit         : www.jtcindia.org
**/
Public class Hello {
@inject
Private A aobj;

@inject
@Qualifier("bo2")
Private B bobj;

Public void show(){
System.out.println(aobj);
System.out.println(bobj);
}}
```

JTCConfig.java

```
Package com.jtcindia.spring;
Import org.springframework.context.Application
Bean;
Import
org.springfromework.Context.annotation.Configura
tion;
Import
org.springframework.context.annotation.ImportRe
source;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit         : www.jtcindia.org
**/
@Configuration
@importResource("jtcindia.xml")
Public class JTCConfig {
Public JTCConfig(){
System.out.println("spring Container is new getting
ready");
}
@Bean(name="hello");
Public HellogetHello(){
System.out.println("getHello()");
Return new Hello();
}}
```

Jtcindia.xml

```
<?xml version ="1.0" encoding="UTF-8"?>
<?xml version ="1.0" encoding="UTF-8"?>
<beans xmlns=http://www.springframework.org/schema/beans
        Xmlns:p=http://www.springframework.org/schema/p
        Xmlns:context=http://www.springframewok.org/schema/context
        Xsi:schemaLocation="http://www.springframework.org/schema/beans
        Xmlns:xsi="http://www.w3.org/2001/xmlschema-instance:
http://www.springframework.org/schema/beans/spring-beans-3.1.xsp
http://www.springframework.org./schema/context http://www.springframework.org/schema/context/
spring context-3.1.xsp">
<context:annotation-config/>
<bean id="ao" Class="com.jtcindia.spring.A"
             P:a="99"  P:msg="A99"/>
```

```
<bean id="bo" class="com.jtcindia.spring.B">
            <constructor-are value ="88"/>
            <constructor-are value="B88"/>
</boan>
</boans>
```

Jtc32: Files required

Jtc32.java	A.java
B.java	Hello.java
JTCConfig.java	

JTCConfig.java

```
Package com.jtcindia.spring;
Import org.springframework.context.annotation.Bean;
Import org.springframework.context.annotation.configuration;
Import org.springframework.context.annotation.ImportResource;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
@Configuration
        Public class JTCConfig {
              System.out.println("spring Container is new getting Ready");
        }
@Bean(name="hello")
Public Hello getHello(){
       System.out.println("getHello()");
       Return new Hello();
}
@Bean(name="ao")
Public A getA(){
            A obj=new A();
            Obj.setA(99);
            Obj.setMsg("manish");
            Return obj;
}
@Bean(name="bo")
Public B getB(){
             B obj=new B(88,"Som");
             Return obj;
}
}
```

Jtc33: Files required

Jtc33.java	Hello.java
JTCConfig.java	

Hello.java	JTCConfig.java

Package com.jtcindia.spring;
 	
Import javax.annotation.postConstruct;
Import javax.annotaion.PreDeestroy;
Import
org.springframework.beans.factory.Disposable
Bean;
Import
org.springframework.beans.factory.InitializingB
ean;
 	
Public class Hello implements
InitializingBean,DisposableBean{
static{
system.out.println("Hello-S.B");
}
Public Hello(){
System.out.println("Hello-D.C");
}
@postConstruct
Public void init(){
Slystem.ot.println("Hello-init()");
}
Public void afterPropertiesSet()");
System.out.println("Hello-
afterPropertiesSet()");
}
Public void myInit(){
System.out.println("Hello-myInit(0");
}
@preDestroy
Public void cleanup(){
System.out.println("Hello-cleanup()");
}
Public void destroy(){
System.out.println("Hello-destroy()");
}
	Package com.jtcindia.spring;
	Import
	org.springframework.context.annotation.Bean;
	Import
	org.springframework.context.annotation.Configuratio
	n;
	@Configuration
	Public class JTCConfig {
	
	Public JTCConfig(){
	System.out.println("spring Container is new getting
	ready");
	}
	
	@Bean(name="hello",initMethod="myInit",destroyM
	et hod="myCleanup")
	Public Hello getHello(){
	System.out.println("getHello()");
	Return new Hello();
	}
	}
	
	Jtc33.java
	Package com.jtcindia.spring;
	Import
	org.springframework.context.annotation.AnnotationC
	onfig Application Context;
	/*
	*@Author: Som Prakash Rai
	*@Company : java Training Center
	*@ visit : www.jtcindia.org
	**/
	Public class Jtc33 {
	Public static void main (string[] args) {
	AnnotationConfigApplicationContext ctx=new
	AnnotationConfigApplicationContext ctx ctx=new
	AnnotationConfigApplicationContext(jtcconfig.class);

Public void myCleanup(){
System.out.println("Hello-mycleanup()");
}
Public void show(){
System.out.println("hello-show()");
}
}
	System.out.println("New Spring container is Reday");
	Hello h=(Hello) ctx.getBean (Hello.class);
	h.show();
	ctx.registerShutdownHook();
	}
	}

Q72) How can I implement annotation Based Autowiring?
Ans: using the following 3 Annotations

@Autowired
@Resource.
@inject

Q73) what is difference among @Autowired, @Resource and @inject?
Ans Refer Notes

Q74) Annotaions are replacement for XML.but still we are writing Beans in xml only. Can I use Only annotations without xml?

Ans: you, you can configure your Beans with java Based configuration without XML configuration.
Use the following Annotaions for java Based Configuration

@ configuration
@Bean

Refer the Jtcs form Jtc27 to Jtc33.

Q75) what is the use of @ Import Annotation?
Ans: Refer Notes

Q76) What is the use of @ Import Resource Annotation?
Ans: Refer Notes

Q77) What is the use of @scope Annotation?
Ans: Refer notes

Q78)d what is the use of @Lazy Annotation?
Ans: Refer notes

Q79) What are the Intialization callbacks provided with BeanFactory container?
Ans: Refer the Notes.

Q80) what are the Intialization callbacks provided with ApplicationContext container?
Ans: Refer the Notes.

Q81) You can Inject the required resources with D.I? what is the use of initialization callbacks?
Ans: Mainly to check whether resources are initialized by the spring Container or not.

```
classHello{
@Autowired
Hai hai;

@Post Construct
Public void init(){
If(hai==null){
Hai=.....,
}else{
Throw some Exception;
}
}
Void show(){
Hai.m1();
}
}
```

Q82) what are the Disposable callbacks provided with BeanFactory container?
Ans: Refer the Notes.

Q83) what are the Disposable callbacks provided with ApplicationContext container?
Ans: Refer the Notes.

Q84) How can I get the reference of Bean Factyory container into the bean?
Ans: There are two ways to initialize your bean with Bean Factory reference.l

By implementing Bean FactoryAware interface and overriding setBeanFactory() method.
By using @Autowired (Use this)

Q85) How can I get the reference of ApplicationContext Container into the bean?
Ans: There are two ways to initialize your bean with Application Context reference.

By implementing ApplicationContextAware interface and overriding setApplicationContext() method.
By using @Autowired (use this)

Q86) what is the use of bean Post processor?
Ans: Bean Post Processor allows to extend the Functionality of ApplicationContextContainer.
Consider the following case;

You want to make JTC() method as lifecycle method for every bean.
Jtc() method has to be called when bean class

Is implementing JTC interface.
Is Containing the method which is marked with @JTC.

A)Bean class implementing JTC interface.

```
Class hello imp jtc, initinalizing Bean{
Public void jtc(){
//dosome thins.
}
}
Public class MyBeanPostProcessor implements Bean Post Processor{
Public object post Process Before Initialization (Objectd object, String bname){
//1.get object of the Bean
//2.get the List of interfaces implemented by the Bean class using Reflection .
//3.Check the JTCinterface is in the list
            Then call jtc() method
                        }
            .....
}
```

Method in the Bean class marked with @jtc

```
Class Hello imkp initializingBean{
@JTC
Public void jtc(){
// do some thing.
}
}
```

```
Public class MyBeanPostProcessor implements Bean Post Processor{
Public object post Process Before Initialization (Object object, String bname){
//1.get object of the Bean
//2.get the List of interfaces implemented by the Bean class using Reflection .
//3.Check whether any method is marked with @JTC.
//4.If @JTC is found for any method
            Then call that method
                        }
            .....
}
```

Q87) what are the Spring Containers Supported?
Ans: Refer Notes.

Q88) What are the difference between BeanFactory and Application Context Container?

Ans: Refer Notes.

Q89) we are creating Spring container object in main method. After scope over still container objects exits. How to justify (Every object should be inside scope).
Ans: if container is shutdown then it has to invoke disposable callbacks.

Q90) Is there any relationship between Bean Factory and ApplicationContext Container?
Ans: Application Context is the Decorator of Bean Factory.
ApplicationContext = BeanFactory + Extra code
ApplicationContext creates the instance of Bean Factory internally.

Q91) Why we are using Abstract ApplicationContext ?
Ans: only to invoke the method registerShutdownHook()

Q92) why we are using AnnotaionConfiguration ApplicationContext ?
Ans: To use java Based Configuration
Q93) Can I Write multiple spring configuration files?
Ans: Spring Application can have mutilple spring Configuration files
You can use multiple spring configuration files in two ways.
Assume that you have written JTC.xml. jtc1.xml. jtc2.xml
A) in jtc.xml

```
<import resource=”jtc1.xml”/>
<import resource=”jtc2.xml”/>
Ctx=new classPathXmlApplication Context(“jtc.xml”);
```

B) String str [] ={“jtc.xml”,”jtc1.xml”,”jtc2.xml” };

```
Ctx=new classPathXmlApplicationContext (str);
```

Q94) can I write multiple java Based Configuration clasess?
Ans: You You can

Q95) whether Spring Container instance Creation is sigletorn be multi-threaded.
Ans: Depending on You.

Q96) How can I decide to use type of spring container?
Ans: Use always Application context only.

Q97) There are two classes A(Lazy) and B(Agresive) and are in Inheritance relationship. When spring Container is loading sub class what will happen to super class?
Ans: one instance of B will be created. Loads both classes and allocates the memory for both the class variables and invokes the both the class constructors.

Q98) Is there any way to extend the Bean Factory container functionality?
Ans: No.

Q99) Can I clone Application Context object?
Ans: No No No

Q100) Can I inject the Bean with one scope into another Bean with different scope?
Ans: Yos.

Q101) Can I mark two methods with @ post Construct Annotation ?
Ans: Yos, you can

Q102) Can I mark two methods with @ Pre Destroy annotation?
Ans: you you can

Q103) Can I inject the Bean defined one xml into another bean defined in another xml?
Ans: You.

Q104) What are Inner Beans?
Ans: when you specify the Bean definition inside another definition then it is called as Inner Bean

```
<bean id="hai" class="…..Hai"/>
<bean id="hello" class="…..Hello">
<property name="hai" ref="hai"/>
</bean>
```

With inner Beans:

```
<bean id="hello"class="….Hello">
<property name="hai">

<bean class="….Hai"/>
</broperty>
</bean>
```

Q105) while you are configuring the beans in xml, we are hard-coding the values? Con I make dynamic?

Ans: You, you can use property files. (using Externalizing Bean Properties)

Q106) what is the use of Application Events?
Ans: Refer Notes

Q107) How can I implement Application Events?
Ans: Refer Notes

Q108) How can I publish Application Events?
Ans: Refer Notes

Q109) How can I implement Application Listeners?
Ans: Refer Notes

Q110) How can I Register Application Listeners?
Ans: Refer Notes

Q111) Can I register multiple Application Listeners with the spring Container ?
Ans: Yes

Q112) what is the use of Property Editors?
Ans: Refer Notes

Q113)How can I develop Custom property Editors?
Ans: Refer Notes

Q114) How can I Register Custom Property Editors?
Ans: Refer Notes

Q115) How can I Register Message Bundles?
Ans: Refer Notes
Q116) How can I Access the properties from Message Bundles?
Ans: Refer Notes

Q117) Why I need to inject Application Context into the Bean?
Ans: You can use the following methods of ApplicationContext.

getMessage ()
PublishEvent()
Etc

Q118) what is the difference between classPathResource and File System Resource?
Ans: Refer Notes

Q119) What is the difference between class Path Xml Application Context and Files System Xml Application Context?
Ans: Refer Notes

Q120) what is the sub class of Bean Factory interface?
Ans: Refer Notes

Q121) what are the sub classes of Application Context interface?
Ans: Refer Notes

Q122) what is the use of @Required Annotation?
Ans: Refer Notes

Q123) What is the Difference between @Autowired (required=true) and @ Autowired (required=false)?
Ans: Refer Notes

Q124) what is the use of @Qualifier Annotation? Where Can I use this?
Ans: Refer Notes

Q125) What are the GOF Patterns discussed in spring IOC?
Ans: a) Singleton Pattern
b) Fatory pattern
c) Decorator Pattern