Interview Questions – Part 1

Q1) What is a Spring Bean?
Ans: Any Java class written in Spring is called a Spring Bean

Q2) What are Bean Dependencies?
Ans: Fields/Variables declared in the Beans are called as Bean Dependencies.

Q3) What is called as injection?
Ans: Initialisation of properties in a bean is called as Injection.

Q4) What is Dependency Injection?
Ans:Process of injecting Bean Dependencies automatically in a Bean is called as Dependency Injection.

Q5) How many ways are there to Implement Dependency Injection?
Ans: 3 ways to inject the dependencies
1) Setter Injection
2) Constructor Injection
3) Field Injection.(using Annotations)

Q6) What is Setter injection? How can we Implement it?
Ans:

Q7) What is Constructor Injection? How can we Implement it?
Ans:

Q8) What is Field Injection? How can we Implement it?
Ans:

Q9) What is the use of Spring Configuration Class?
Ans: It is used to define Bean definitions

Q10) Where can we define our Beans?
Ans:In the Spring Configuration Class

Q11) What is the use of @Configuration Annotation?
Ans:It is used to mark the configuration class where beans are defined.

Q12) What happens if I dont use @Configuration?
Ans:

Q13) What is the use of @Bean Annotation?
Ans:It is used to write the Bean Definition.

Q14) What happens if I dont specify @Bean Annotation?
Ans: At Container startup it checks whether any method is marked with @Bean or not. If not marked, it will not consider it as a Bean definition and throws Exception since the Bean is not found.

Q15) What is the default name of a Bean?
Ans: It's Method Name

Q16) How to specify required name for a Bean?
Ans: @Bean("<BeanName>")

Q17) How to create ApplicationContext?
Ans: ApplicationContext ctx= new AnnotationConfigApplicationContext(<ConfigurationClassName.class>);

Q18) How can I access bean Instance from Spring Container?
Ans: Using getBean() method

Q19) What happens when specified Bean Name is not found in the container?
Ans: Throws NoSuchBeanFoundException at runtime.

Q20)C an I have two beans with same name?
Ans: No, Bean names should always be unique.

Q21) Can I have two Beans with same data types?
Ans:Yes

Q22) What are Bean Scopes?
Ans:

Q23) What is the default Bean Scope?
Ans:Singleton Scope

Q24) How to specify required Bean Scope?
Ans: Using @Scope Annotation

Q25) What happens when Bean Scope is Singleton?
Ans:By default, it will be aggressively loaded one time into the conatiner at container startup

Q26) What happens when Bean Scope is Prototype?
Ans:By default, it will be lazily loaded multiple times into the container whenever getBeans() method is called.

Q27) When Bean instance will be created if Bean scope is Singleton?
Ans: At Container startup

Q28) When Bean instance will be created if Bean scope is Prototype?
Ans: When getBean() method is called.

Q29) What are the Loading types available in spring?
Ans: Aggressive and Lazy Loading types

Q30) What is the default Loading Type?
Ans:It Depends on Bean Scope

Q31) How to specify the required Loading Type?
Ans:Using @Lazy Annotation

Q32) What happens when loading type is Aggressive?
Ans:

Q33) What happens when loading type is Lazy?
Ans:

Q34) When is the Bean Instance created if Loading Type is Aggressive?
Ans: At Spring Container startup

Q35) When is the Bean Instance created if Loading Type is Lazy?
Ans: Whenever getBean() method is invoked

Q36) What is the default loading type for Singleton beans?
Ans:Aggressive Loading

Q37) What is the default loading type for Prototype beans?
Ans:Lazy Loading

Q38) When Bean Instance is created if I specify Aggressive Loading for Singleton?
Ans:At Spring container startup.

Q39) When Bean Instance is created if I specify Lazy Loading for Singleton?
Ans:When we call getBean() Method

Q40) When Bean Instance is created if I specify Aggressive Loading for Prototype?
Ans:When we call getBean() Method

Q41) When Bean Instance is created if I specify Lazy Loading for Prototype?
Ans:When we call getBean() Method

Q42) How many times a single Bean is loaded?
Ans: only one time

Q43) How many times a Bean will be Instantiated?
Ans: It depends on the scope of the Bean.

Q44) Can i restrict the maximum number of instances to create for a Bean?
Ans: NO

Q45a) How can I make only 5 Singletons?
Ans define 5 singleton Beans

Q45b) I want to talk to MySQL and Oracle. I need only Two Data Source objects. How to get exactly Two DataSource Objects?
Ans: define Two DataSource as Singletons.

Q46) What is the order of loading the Beans into Spring Container?
Ans: All Beans will be loaded in the order we define in the configuration class. The loading order may change depending on the Bean dependencies.

---Answer the following questions based on Lab1----

Q47) Beans are configured in the order A,B,Hello. What is the Loading Order?
Ans:Loading Order A, B , Hello

Q48) Beans are configured in the order B,A,Hello. What is the Loading Order?
Ans:Loading Order A, B , Hello

Q49) Beans are configured in the order A,B,Hello & createHello(A ao, B bo). What is the Loading Order?
Ans:Loading Order A, B , Hello

Q50) Beans are configured in the order A,B,Hello & createHello(B bo, A ao). What is the Loading Order?
Ans:Loading Order B, A , Hello

Q51) How can I use Existing Singletons?
Ans:

Q52) Can I write multiple Configuration classes?
Ans: Yes

Q53) What are the ways available to use multiple Configuration classes?
Ans:

Q54) What is the use of @Import Annottaion?
Ans:

Q55) What is the use of @ImportResource Annottaion?
Ans:

Q56) What are the Various Types of Bean Properties which can be Injected?
Ans:

Q57) How to configure List?
Ans:

Q58) What happens when Required List is not found?
Ans: Empty List will be injected.

Q59) What sub-type of List will be used by-default?
Ans:ArrayList

Q60) How to configure Set?
Ans:

Q61) What happens when Required Set is not found?
Ans:

Q62) What sub-type of Set will be used by-default?
Ans:

Q63) How to configure Map?
Ans:

Q64) What happens when Required Map is not found?
Ans:

Q65) What sub-type of Map will be used by-default?
Ans:

Q66) What is Wiring and What are types of Wiring?
Ans:

Q67) What are the Wiring Processes available?
(Or)What Models can be used to detect the beans?
Ans: There 3 Wiring Models.
1)byName wiring process
2)byType wiring process

Q68) What is Explicit Wiring?
Ans:

Q69) How beans will be detected in the case of Explicit Wiring?
Ans:
In the case Explicit Wiring , Beans will be detected with byType first.
If matching bean found with ByType then that will be injected.
If matching bean not found then that Beans will be detected with byName next.
If matching bean found with ByName then that will be injected.

Answer the following based on Explicit Wiring.(70,71,72)

Q70) What happens when 0 beans found?
Ans:

Q71) What happens when exactly 1 bean found?
Ans:

Q72) What happens when Multiple beans found?
Ans:

Q73) What is AutoWiring?
Ans:

Q74) How can I implement AutoWiring?
Ans:Two ways
1)Using autowire attribute (deprecated).
2)Annotation Based Autowiring.(***)

Q75) What happens when I specify byName AutoWire process?
Ans: Beans will be detected based on Bean name and then Injects the matching bean with Setter method.

Answer the following based on byName AutoWire process(76,77)

Q76) What happens when 0 beans found?
Ans:

Q77) What happens when exactly 1 bean found?
Ans:

Q78) What happens when I specify byType AutoWire process?
Ans: Beans will be detected based on Data Type and then Injects the matching bean with Setter method.

Answer the following based on byType AutoWire process(79,80,81)

Q79) What happens when 0 beans found?
Ans:

Q80) What happens when exactly 1 bean found?
Ans:

Q81) What happens when Multiple beans found?
Ans:

Q82) Can I Inject sub type objects into super type variables?
Ans: Yes

Q83) Can I Implement Cyclic Dependency in Spring?
Ans: Yes

Q84) What is Field Injection? How to implement this?
Ans: Injecting Bean Dependecies directly without setter method and constructor is called Field Injection.
Field Injection can be implemented using Annotaions.

Q85) How to implement Annotation based AutoWiring?
Ans: 3 ways
@Autowired
@Resource
@Inject

Q86) Where can I use @Autowired annotation?
Ans: You can @Autowired annotation in 3 places.
1)Fields
2)setter methods
3)Constructor (in XML)

Q87) What happens when I specify @Autowired?
Ans: @Autowired detects the beans with byType process and then injects them directly without setter method and constructor

Q88) Can I use @Autowired for ByName Process?
Ans: Yes , You can use @Autowired along with @Qualifier for ByName Process

Q89) When to use required=true for @Autowired?
Ans:

Q90) When to use required=false for @Autowired?
Ans:

Q91)What is the default value of required attribute in @Autowired?
Ans: true

Answer the following based on @Autowired(required=true) (92,93,94)

Q92) What happens when 0 beans found?
Ans:

Q93) What happens when exactly 1 bean found?
Ans:

Q94) What happens when Multiple beans found?
Ans:

Answer the following based on @Autowired(required=false) (95,96,97)

Q95) What happens when 0 beans found?
Ans:

Q96) What happens when exactly 1 bean found?
Ans:

Q97) What happens when Multiple beans found?
Ans:

Answer the following based on @Autowired(required=true) and ByName Process (@Qualifier) (98,99)

Q98) What happens when 0 beans found?
Ans:

Q99) What happens when exactly 1 bean found?
Ans:

Answer the following based on @Autowired(required=false) and ByName Process (@Qualifier) (100,101)

Q100) What happens when 0 beans found?
Ans:

Q101) What happens when exactly 1 bean found?
Ans:

Q102) What happens when I specify @Resource?
Ans: @Resource detects the beans with byType process and then injects them directly without setter method and constructor

Q103) Can I use @Resource for ByName Process?
Ans: Yes , You can use @Resource with name attribute for ByName Process

Answer the following based on @Resource with ByType(104,104,106)

Q104) What happens when 0 beans found?
Ans:

Q105) What happens when exactly 1 bean found?
Ans:

Q106) What happens when Multiple beans found?
Ans:

Answer the following based on @Resource with ByName(107,108)

Q107) What happens when 0 beans found?
Ans:

Q108) What happens when exactly 1 bean found?
Ans:

Q109) What happens when I specify @Inject?
Ans: @Inject detects the beans with byType process and then injects them directly without setter method and constructor

Q110) Can I use @Inject for ByName Process?
Ans: Yes , You can use @Inject along with @Qualifier for ByName Process

Answer the following based on @Inject with ByType(111,112,113)

Q111) What happens when 0 beans found?
Ans:

Q112) What happens when exactly 1 bean found?
Ans:

Q113) What happens when Multiple beans found?
Ans:

Answer the following based on @Inject with ByName(114,115)

Q114) What happens when 0 beans found?
Ans:

Q115) What happens when exactly 1 bean found?
Ans:

Q116) What are the Intialization callbacks provided with BeanFactory container?
Ans:

Q117) What are the Intialization callbacks provided with ApplicationContext container?
Ans:

Q118) What is the use of Initialization Callbacks?
Ans:

Q119) What is the order of invoking Initialization Callbacks?
Ans:

Q120) Can I mark the multiple methods with @PostConstruct
Ans:Yes

Q121) You can Inject the required resources with D.I ?What is the use of Intialization callbacks?
Ans: Mainly to check whether resources are initialized by the Spring Container or not.

```
class Hello{
@Autowired
Hai hai;

@PostConstruct
public void init(){
if(hai==null){
hai=.....;
} else {
throw Some Exception;
}
}
void show(){
hai.m1();
}
}
```

Q122) What are the Disposable callbacks provided with BeanFactory container?
Ans: Refer the Notes.

Q123) What are the Disposable callbacks provided with ApplicationContext container?
Ans: Refer the Notes.

Q124) What is the use of Disposable Callbacks?
Ans:

Q125) What is the order of invoking Disposable Callbacks?
Ans:

Q126) Can I mark the multiple methods with @PreDestroy?
Ans:Yes

Q127) When Initialization Callbacks will be called by the container in the Bean Life?
Ans:

Q128) When Disposable Callbacks will be called by the container in the Bean Life?
Ans:

Q129) What are Aware interfaces?
Ans:

Q130) How Can I access the Bean Name inside the Bean Class?
Ans:

Q131) How Can I access the BeanFactory inside the Bean Class?
Ans: Two Ways.

Q132) How Can I access the ApplicationContext inside the Bean Class?
Ans: Two Ways.

Q133) How Can I Extentend Container Functioanlity?
Ans:

Q134)How to use BeanPostProcessor?
Ans: BeanPostProcessor allows to extend the functionality of ApplicationContext container.
Consider the following case:

You want to make jlc() method as lifecycle method for every bean.

jlc() method has to be called when bean class

- is implementing JLC interface.
- is containing the method which is marked with @JLC.

A) Bean class Implementing JLC interface.

```
class Hello imp JLC, InitializingBean{
public void jlc(){
// do some thing.
}
}
public class MyBeanPostProcessor implements BeanPostProcessor {
public Object postProcessBeforeInitialization(Object object, String bname){
 //1. get Object of the Bean
 //2. get the List of interfaces implemeted by the Bean class using Reflection.
// 3.Check the JLC interface is in that list.
 // 4.If JLC interface is in the List
        then call  jlc() method
          }
...
}
```

B) Method in the Bean class marked with @JLC

```
class Hello imp InitializingBean{
@JLC
```

```
public void jlc(){
// do some thing.
}
}
public class MyBeanPostProcessor implements BeanPostProcessor {
public Object postProcessBeforeInitialization(Object object, String bname){
 //1. get Object of the Bean
 //2. get the List of methods available in Bean class using Reflection.
// 3.Check whether any method is marked with @JLC.
 // 4.If @JLC is found for any method
        then call  that method
          }
...
}
```

Q135) When exactly postProcessBeforeInitialization() method will be called?
Ans:

Q136) When exactly postProcessAfterInitialization() method will be called?
Ans:

Q137) What are the parameters of postProcessBeforeInitialization() method and what is the importance of them?
Ans:

Q138) What are Spring Containers avaialble?
Ans:

Q139) What are the sub-classes of BeanFatory interface?
Ans:

Q140) How to create the BeanFactory object?
Ans:

Q141) What are the Lifecycle activities of Bean running with BeanFactory Container?
Ans: 8-Steps

Q142) What are the sub-classes of ApplicationContext interface?
Ans:

Q143) How to create the ApplicationContext object?
Ans:

Q144) What are the Lifecycle activities of Bean running with ApplicationContext Container?
Ans: 12-steps

Q145) What are the difference between BeanFactory and ApplicationContext ?
Ans:

Q146) What happens at Spring Container shut-down time?
Ans:

Q147) What are the common taks between BeanFactory and ApplicationContext ?
Ans:

Q148) What are the attributes of @Bean annotation?
Ans:

Q149) What Annotaions can be used for Bean Definition?
Ans:

Q150) What are the difference among @Autowired ,@Resource,@Inject?
Ans:

Q151) What is the default loading type with BeanFactory container?
Ans:

Q152) What is the default loading type with ApplicationContext container?
Ans:

Q153) How to decide to the type of Dependency Injection?
Ans:

Setter Injection or Filed Injection -> Make the beans loosely coupled
Constructor Injection -> Make the beans tightly coupled

Ex:

```
class A{ }
class Hai{
  A ao;
  ...
  Hai(){ }
  public void setAo(A ao){
    this.ao=ao;
  }
}
```

Without A bean instance,
Container will create Hai bean instance
Container cannot inject A bean into Hai.
i.e. Without A bean, Hai bean is instantiated. So these two beans are loosely coupled.

```
class Hello{
A ao;
Hello(A ao){
  this.ao=ao;
}
}
```

Without A bean instance,
Container cannot create Hello bean instance.
i.e Without A bean, Hello bean will not be instantiated. So these two beans are tightly coupled.

Q154) Is there any relationship between BeanFactory and ApplicationContext container?
Ans: ApplicationContext is the decorator of Bean Factory.

ApplicationContext = BeanFactory + Extra code
ApplicationContext creates the instance of BeanFcatory internally.

Q155) How Can I decide to use type of Spring container?
Ans : Use always ApplicationContext only.

Q156) Is there any way to extend the BeanFactory container functionality?
Ans : No.

Q157) Can I clone Application Context object?
Ans: No No No

Q158) Can I inject the Bean with One scope into another Bean with different scope?
Ans: Yes .

Q159) What is the difference between @Autowired(required=true) and @Autowired(required=false)?
Ans:

Q160) What is the use of @Qualifier Annotation? Where Can I use this?
Ans:

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 src folder.

oracle.properties
```
oracle.driver=oracle.jdbc.driver.OracleDriver
oracle.url=jdbc:oracle:thin:@localhost:1521:XE
```

mysql.properties
```
mysql.driver=com.mysql.jdbc.Driver
mysql.url=jdbc:mysql://localhost:3306/jlcdb
```

2) To Externalize Bean Properties, you have to register bean called PropertySourcesPlaceholderConfigurer in Spring Configuration Class as follows.

```
@Bean
public  static PropertySourcesPlaceholderConfigurer getProps() {
        return new PropertySourcesPlaceholderConfigurer();
}
```

3) Now You can access the values of required Keys in two ways.

A)Inject the Environment Bean in other Required Bean

```
@Autowired
private Environment env;
```

Get the Value of Property as follows

```
String mydc=env.getProperty("mysql.driver")
String oradc=env.getProperty("oracle.driver")
```

B) Declare the Variable to hold the required Property value and mark that variable with @Value annotation by specifying the key.

```
@Value("${mysql.driver}")
private String mydc;

@Value("${oracle.driver}")
private String oradc;
```

Lab43: Files required

1. Lab43.java	**2. TestDAO.java**
3. DataSource.java	**4. JLCAppConfig.java**
5. oracle.properties	**6. mysql.properties**
7. commons.properties	

1. Lab43.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab43 {
public static void main(String[] args) {
ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

TestDAO testDAO=ctx.getBean("testDAO",TestDAO.class);
testDAO.show();
}
}
```

2. TestDAO.java

```
package com.coursecube.spring;

import javax.annotation.Resource;
import org.springframework.beans.factory.annotation.*;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class TestDAO {

	@Autowired
	@Qualifier("oracleds")
	private DataSource orads;

	@Resource(name="mysqlds")
	private DataSource myds;

	void show() {
		System.out.println(orads);
		System.out.println(myds);
	}
}
```

3. DataSource.java

```
package com.coursecube.spring;
/*
```

```
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class DataSource {
private String driverClass;
private String url;
private String username;
private String password;
private int min;
private int max;
private int timeout;

public void setDriverClass(String driverClass) {
        this.driverClass = driverClass;
}
public void setUrl(String url) {
        this.url = url;
}
public void setUsername(String username) {
        this.username = username;
}
public void setPassword(String password) {
        this.password = password;
}
public void setMin(int min) {
        this.min = min;
}
public void setMax(int max) {
        this.max = max;
}
public void setTimeout(int timeout) {
        this.timeout = timeout;
}
@Override
public String toString() {
        return  driverClass + "\n " + url + "\n " + username + "\n " + password + "\n " + min + "\n " +
max + "\n " + timeout ;
}
}
```

4. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
```

```
import org.springframework.context.annotation.PropertySource;
import org.springframework.context.support.PropertySourcesPlaceholderConfigurer;
import org.springframework.core.env.Environment;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
@PropertySource({"classpath:oracle.properties","classpath:mysql.properties","classpath:commons.
properties"})
public class JLCAppConfig {

	@Autowired
	private Environment env;

	@Value("${db.password}")
	private String password;

	@Value("${db.min}")
	private int minimum;

	@Value("${db.max}")
	private int maximum;

	@Value("${db.timeout}")
	private int timeout;


	@Bean(name="oracleds")
	public DataSource getOracleDS() {
		DataSource orads=new DataSource();

		String driverClass=env.getProperty("oracle.driver");
		String url=env.getProperty("oracle.url");
		String username=env.getProperty("oracle.username");

		orads.setDriverClass(driverClass);
		orads.setUrl(url);
		orads.setUsername(username);

		orads.setPassword(password);
		orads.setMin(minimum);
		orads.setMax(maximum);
		orads.setTimeout(timeout);

		return orads;
	}
```

```
	@Bean(name="mysqlds")
	public DataSource getMySQLDS() {
		DataSource myds=new DataSource();
		String driverClass=env.getProperty("mysql.driver");
		String url=env.getProperty("mysql.url");
		String username=env.getProperty("mysql.username");

		myds.setDriverClass(driverClass);
		myds.setUrl(url);
		myds.setUsername(username);

		myds.setPassword(password);
		myds.setMin(minimum);
		myds.setMax(maximum);
		myds.setTimeout(timeout);

		return myds;
	}
	@Bean(name="testDAO")
	public TestDAO getTestDAO() {
		return new TestDAO();
	}
	@Bean
	public static PropertySourcesPlaceholderConfigurer getProps() {
		return new PropertySourcesPlaceholderConfigurer();
	}
}
```

5. oracle.properties

```
oracle.driver=oracle.jdbc.driver.OracleDriver
oracle.url=jdbc:oracle:thin:@localhost:1521:XE
oracle.username=system
```

6. mysql.properties

```
mysql.driver=com.mysql.jdbc.Driver
mysql.url=jdbc:mysql://localhost:3306/jlcdb
mysql.username=root
```

7. commons.properties

```
db.password=srinivas
db.min=50
db.max=500
db.timeout=1
```

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.

1) Write one or more message bundles to support multiple languages.

basename messages.properties

Syntax:

basename_LanguageCode_CountryCode.properties
basename_LanguageCode.properties

messages.properties (English)

un.required=Username is Required (English).
errors.required={0} is Required (English).

messages_hi.properties (Hindi)

un.required=Username is Required (Hindi).
errors.required={0} is Required (Hindi).

2) Register the bean called ResourceBundleMessageSource with Spring Container by specifying the following in Spring Configuration Document.

```
@Bean
  public MessageSource messageSource() {
    ReloadableResourceBundleMessageSource messageSource = new
ReloadableResourceBundleMessageSource();
    messageSource.setBasename("messages");
    messageSource.setDefaultEncoding("UTF-8");
    return messageSource;
  }
```

3) Access the properties from message bundles using following methods of ApplicationContext or MessageSource

getMessage(String key,Object []args,Locale);
getMessage(String key,Object []args,String defaultMessage , Locale);

Ex:

```
ms.getMessage("un.required",null,new Locale("hi"));
ctx.getMessage("un.required",null,new Locale("en"));

ms.getMessage("errors.required",new Object[]{"Phone"},new Locale("hi"));
ctx.getMessage("errors.required",new Object[]{"Phone"},new Locale("en"));
```

Lab44: Files required

1. Lab44.java	**2. TestBean.java**
3. JLCAppConfig.java	**4. messages.properties**
5. messages_hi.properties	

1. Lab44.java

```
package com.coursecube.spring;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class Lab44 {
public static void main(String[] args) {

ApplicationContext ctx=new AnnotationConfigApplicationContext(JLCAppConfig.class);
System.out.println("---------Now Spring Container is Ready-----");

TestBean testBean=ctx.getBean("testBean",TestBean.class);
testBean.showEnglish();
System.out.println("--------------------");
testBean.showHindi();
System.out.println("--------------------");
testBean.show();
}
}
```

2. TestBean.java

```
package com.coursecube.spring;

import java.util.Locale;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
public class TestBean {

@Autowired
ApplicationContext ctx;
```

```
void show() {

String msg1=ctx.getMessage("hello.required",new Object[] {},"Hello Guys",new Locale("en"));
System.out.println(msg1);

String msg2=ctx.getMessage("hai.required",new Object[] {},new Locale("en"));
System.out.println(msg2);

}

void showEnglish() {

String msg1=ctx.getMessage("uname.required",null,null); //Default Lan
System.out.println(msg1);

String msg2=ctx.getMessage("uname.required",null,new Locale("en"));
System.out.println(msg2);

String msg3=ctx.getMessage("errors.required",new Object[] {"Email Id"},new Locale("en"));
System.out.println(msg3);

String msg4=ctx.getMessage("errors.required",new Object[] {" Phone "},new Locale("en"));
System.out.println(msg4);

String msg5=ctx.getMessage("errors.minlength",new Object[] {"Username ","7"},new Locale("en"));
System.out.println(msg5);

String msg6=ctx.getMessage("errors.maxlength",new Object[] {"Username ","15"},new
Locale("en"));
System.out.println(msg6);

String msg7=ctx.getMessage("errors.range",new Object[] {"Age ","20","40"},new Locale("en"));
System.out.println(msg7);

String msg8=ctx.getMessage("errors.range",new Object[] {"Course Fee ","10000","25000"},new
Locale("en"));
System.out.println(msg8);
}

void showHindi() {

String msg3=ctx.getMessage("errors.required",new Object[] {"Email Id"},new Locale("hi"));
System.out.println(msg3);

String msg4=ctx.getMessage("errors.required",new Object[] {" Phone "},new Locale("hi"));
System.out.println(msg4);
```

```
String msg5=ctx.getMessage("errors.minlength",new Object[] {"Username ","7"},new Locale("hi"));
System.out.println(msg5);

String msg6=ctx.getMessage("errors.maxlength",new Object[] {"Username ","15"},new Locale("hi"));
System.out.println(msg6);

String msg7=ctx.getMessage("errors.range",new Object[] {"Age ","20","40"},new Locale("hi"));
System.out.println(msg7);

String msg8=ctx.getMessage("errors.range",new Object[] {"Course Fee ","10000","25000"},new
Locale("hi"));
System.out.println(msg8);

}
}
```

3. JLCAppConfig.java

```
package com.coursecube.spring;

import org.springframework.context.MessageSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.ReloadableResourceBundleMessageSource;
/*
* @Author : Srinivas Dande
* @Company : CourseCube
* @Website : www.coursecube.com
* */
@Configuration
public class JLCAppConfig {

        @Bean(name="testBean")
        public TestBean getTest() {
                return new TestBean();
        }

        @Bean
      public MessageSource messageSource() {
          ReloadableResourceBundleMessageSource messageSource = new
                                        ReloadableResourceBundleMessageSource();
           messageSource.setBasename("messages");
          messageSource.setDefaultEncoding("UTF-8");
         return messageSource;
   }
}
```

4. messages.properties

```
uname.required=Username is Required (English).
errors.required={0} is Required (English).
errors.minlength={0} minlength must be {1} chars (English).
errors.maxlength={0} maxlength must be {1} chars (English).
errors.range={0} must be between  {1} to {2}(English).
```

5. messages_hi.properties

```
uname.required=Username is Required (Hindi).
errors.required={0} is Required (Hindi).
errors.minlength={0} minlength must be {1} chars (Hindi).
errors.maxlength={0} maxlength must be {1} chars (Hindi).
errors.range={0} must be between  {1} to {2}(Hindi).
```