Spring interview questions and answers

Spring interview questions - posted on Feb 04, 2014 at 11:05 AM by Nihal Singh

Q.1 What is Aspect oriented Programming (AOP)?

Basically Aspect oriented programming complements object oriented programming by providing another way of programming model structure.

In addition to classes, AOP gives you aspect, which enables modularization of concerns such as Transaction management or logging and can be separated out from the application business logic of the code (these kinds of concerns are termed as crosscutting concerns). AOP supports separating application business logic from System services.

What is IOC or Dependency Injection?

The basic concept of IOC (Dependency of Injection) is that you do not create your objects but describe how they should be created.

You don't directly connect your component and services together in code but describe which services are needed by which component in configuration file.

You just need to describe the dependency, the Spring container is then responsible for hooking it all up.

When to use Dependency Injections?

There are different scenarios where you Dependency Injections are useful.

- You need to inject configuration data into one or more component.
- You need to inject the same dependency into multiple components.
- You need to inject different implementation of the same dependency.
- You need to inject the same implementation in different configuration.
- You need some of the services provided by container.

When you should not use Dependency Injection?

There were scenarios where you don't need dependency injections e.g.

- You will never need a different implementation.
- You will never need different configurations.

What is Bean Factory in Spring?

A Bean Factory is like a factory class that contains collections of beans. The Bean Factory holds bean definition of multiple beans within itself and then instantiates the bean when asked by client.

Bean Factory is actual representation of the Spring IOC container that is responsible for containing and managing the configured beans.

Different Spring Bean Scope.

singleton: Return a single bean instance per Spring IOC container.

prototype: Return a new bean instance each time when requested.

request: Return a single bean instance per HTTP request.

session: Return a single bean instance per HTTP session.

global session: Return a single bean instance per global HTTP session and only valid when used in portlet context.

Q.7 How you will decide when to use prototype scope and when singleton scope bean?

You should use the prototype scope for all beans that are stateful and the singleton scope should be used for stateless beans.

Q.8 What are the different types of IOC?

There are three types of dependency Injection:

Constructor Injection: dependencies are provided as a constructor parameter.

Example: You want to inject some object say Foo through constructor in class HellowWorld like below.

```
public class HelloWorld{
public HelloWorld(Foo foo){
        this.foo = foo;
    }
}
```

In configuration file you need to do following entry.

```
<bean id="helloWorldBean" class="com.xyz.services.HelloWorld">

<constructor-arg ref="fooBean" />

</bean>

<bean id="fooBean" class="com.xyz.service.Foo">

</bean>
```

Setter Injection: Dependencies are assigned through setter method.

Example: same example as above.

```
public class HelloWorld{
private Foo fooBean;
public HelloWorld(){ }
```

```
public void setFooBean(Foo fooBean){
this.fooBean=fooBean;
}
```
And in configuration file you need to do following entry.

```
<bean id="helloWorldBean" class=" com.xyz.services.HelloWorld">

<property name=fooBean ref="fooBean" />

</bean>

<bean id="fooBean" class="com.xyz.service.Foo">

</bean>
```

Interface Injection: Injection is done through an interface and not supported in spring framework.

Q: 9 How to Call Stored procedure in Spring Framework?

To call a Stored procedure in Spring framework you need to create Class which will should extends StoredProcedure class. Take the example of getting Employee Details by Employee Id. package com.mytest.spring.storeproc

```
import java.sql.ResultSet;
import java.sql.Types;
import java.util.HashMap;
import java.util.Map;
import javax.sql.DataSource;
import org.springframework.jdbc.core.SqlOutParameter;
import org.springframework.jdbc.core.SqlParameter;
import org.springframework.jdbc.object.StoredProcedure;
public class EmployeeInfo extends StoredProcedure {

                private static final String EMP_ID = "EMP_ID";
                private static final String EMP_NAME = "EMP_NAME";
                private static final String JOIN_DATE = "JOIN_DATE";
                public SnapshotSearchStoredProcedure(DataSource dataSource, String procedureName)
                {
                     super(dataSource, procedureName);
                    declareParameter(new SqlParameter(EMP_ID, Types.NUMERIC));
                    declareParameter(new SqlOutParameter(EMP_NAME, Types.VARCHAR));
                    declareParameter(new SqlOutParameter(JOIN_DATE, Types.VARCHAR));
                    compile ();
                }

                 public Map execute(Integer empId)
                 {
                          Map<String, Object> inputs = new HashMap<String, Object>();
                          inputs.put(P_CLD_IDR, empId);
                          Map<String, Object> result = execute (inputs);
                          return result;
                 }
}
```

You just need to call the execute method from the DAO layer.

Spring interview questions - posted on June 27, 2013 at 17:05 PM by Kshipra Singh

1. Differentiate between BeanFactory and ApplicationContext in spring.

- With ApplicationContext more than one config files are possible while only one config file or .xml file is possible with BeanFactory.
- ApplicationContext publishes events to beans that are registered as listeners while BeanFactory doesn't support this
- ApplicationContext support internationalization messages, application life-cycle events, validation and many enterprise services like JNDI access, EJB integration, remoting etc. while BeanFactory doesn't support any of these.

2. What is the difference between singleton and prototype bean?

Mainly it is the scope of a beans which defines their existence on the application
- Singleton: It means single bean definition to a single object instance per Spring IOC container.
- Prototype: It means a single bean definition to any number of object instances.

3. How do beans become 'singleton' or prototype?

- There exists an attribute in bean tag, called 'singleton'.
- If it is marked 'true', the bean becomes singleton.
- If it is marked 'false', the bean becomes prototype.

4. What type of transaction Management Spring support?

Spring supports two types of transaction management:

1. Programmatic transaction management
2. Declarative transaction management.

5. When do you use programmatic and declarative transaction management ?

- Programmatic transaction management is used preferably when you have a small number of transactional operations.
- Incase of large number of transactional operations it is better to use declarative transaction management.

6. What is IOC?

- IOC stands for Inversion of Control pattern.
- It is also called as dependency injection.
- This concept says that you do not create your objects but describe how they should be created.
- Similarly, you do not directly connect your components and services together in code but describe which services are needed by which components in a configuration file.
- A container then hooks them all up.

7. What are the different types of IoC (dependency injection) ?

There are three types of dependency injection:

a.) Constructor Injection : Here dependencies are provided as constructor parameters.
b.) Setter Injection : Dependencies are assigned through JavaBeans properties.
c.) Interface Injection : Injection is performed through an interface.

Spring supports only first two categories of Injection.

8. What are the benefits of IOC?

The main benefits of IOC or dependency injection are:

a.) It minimizes the amount of code in your application.
b.) It makes your application easy to test as it doesn't require any singletons or JNDI lookup mechanisms in your unit test cases.
c.) Loose coupling is promoted with minimal effort and least intrusive mechanism.
d.) IOC containers support eager instantiation and lazy loading of services.

9. What is Bean Wiring ?

Bean wiring means creating associations between application components i.e. beans within the spring container.

10. How do you access Hibernate using Spring ?

There are two ways to Spring's Hibernate integration:
a.) By Inversion of Control with a HibernateTemplate and Callback
b.) By extending HibernateDaoSupport and Applying an AOP Interceptor

11. How would you integrate Spring and Hibernate using HibernateDaoSupport?

This can be done through Spring's SessionFactory called LocalSessionFactory. The steps in integration process are:

a.) Configure the Hibernate SessionFactory
b.) Extend your DAO Implementation from HibernateDaoSupport
c.) Wire in Transaction Support with AOP

12.What are the various transaction manager implementations in Spring ?

1. DataSourceTransactionManager : PlatformTransactionManager implementation for single JDBC data sources.

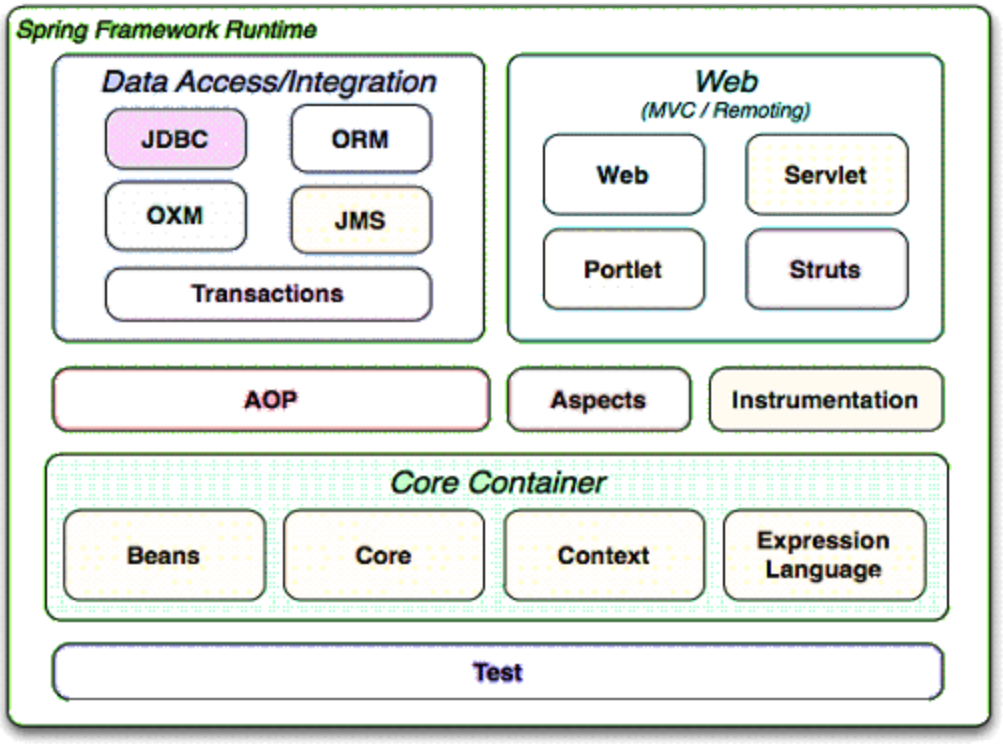
2. HibernateTransactionManager: PlatformTransactionManager implementation for single Hibernate session factories.

3. JdoTransactionManager : PlatformTransactionManager implementation for single JDO persistence manager factories.

4. JtaTransactionManager : PlatformTransactionManager implementation for JTA, i.e. J2EE container transactions.

Q: 13 What are the different modules in spring framework?

The Spring features or organized into about 20 modules. These modules are grouped into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation and Test, as depicted below.



Q: 14 What is Auto Wiring in Spring?

The Auto-wiring in spring framework can be performed by configuring in xml and Spring Auto-Wiring with Annotation @Autowired.

Auto-wiring beans with xml configuration: In Spring framework, you can wire beans automatically with auto-wiring feature. To enable auto-wiring just define the “autowire” attribute in <bean> tag.

<bean id="customer" class="com.test.autowire.Customer" autowire="byName" />

There are 5 modes of Auto-wiring supported.

no – Default, no auto wiring, set it manually via “ref” attribute

<bean id="customer" class="com.test.autowire.Customer">
<property name="person" ref="person" />
</bean>

<bean id="person" class="com.test.autowire.Person" />

byName – Auto wiring by property name. If the name of a bean is same as the name of other bean property, auto wire it.

In below example the name of the person bean is same as name of “customer” bean’s property Person object. So spring will auto-wire it via setter method.

<bean id="customer" class="com.test.autowire.Customer" autowire="byName"/>

<bean id="person" class="com.test.autowire.Person" />

byType – Auto wiring by property data type. If data type of a bean is compatible with the data type of other bean property, auto wire it.

In below example the data type of the person bean is same as name of “customer” bean’s property Person object. So spring will auto-wire it via setter method.

<bean id="customer" class="com.test.autowire.Customer" autowire="byType"/>

<bean id="person" class="com.test.autowire.Person" />

constructor – byType mode in constructor argument.

Here the data type of “person” bean is same as the constructor argument data type in “customer” bean’s property (Person object), so, Spring auto wired it via constructor method – “public Customer(Person person)”

<bean id="customer" class="com.test.autowire.Customer" autowire="constructor"/>

<bean id="person" class="com.test.autowire.Person" />

autodetect – If a default constructor is found, use “autowired by constructor”; Otherwise, use “autowire by type”.

If a default constructor is found, uses “constructor”; Otherwise, uses “byType”. In this case, since there is a default constructor in “Customer” class, so, Spring auto wired it via constructor method – “public Customer(Person person)”

<bean id="customer" class="com.test.autowire.Customer" autowire="autodetect"/>

<bean id="person" class="com.test.autowire.Person" />

Q:15 What is JdbcTemplate in Spring? And how to use it?

The JdbcTemplate class is the main class of the JDBC Core package. The JdbcTemplate (The class internally use JDBC API) helps to eliminate lot of code you write with simple JDBC API (Creating connection, closing connection, releasing resources, handling JDB Exceptions, handle transaction etc.). The JdbcTemplate handles the creation and release of resources, which helps you to avoid common error like forgetting to close connection.

Examples:

1. Getting row count from database.

int rowCount = this.jdbcTemplate.queryForObject("select count(*) from t_employee", int.class);

2. Querying for a String.

String lastName = this.jdbcTemplate.queryForObject("select last_name from t_employee where Emp_Id = ?", new Object[]{377604L}, String.class);

3. Querying for Object

```
Employee employee = this.jdbcTemplate.queryForObject( "select first_name, last_name from t_employee where
Emp_Id = ?", new Object[]{3778604L}, new RowMapper()
{
public Employee mapRow(ResultSet rs, int rowNum) throws SQLException {
Employee employee = new Employee();
employee.setFirstName(rs.getString("first_name"));
employee.setLastName(rs.getString("last_name"));
return employee;
}
});
```

4. Querying for N number of objects.

```
List<Employee> employeeList = this.jdbcTemplate.query("select first_name, last_name from t_employee", new
RowMapper<Employee>() {
public Employee mapRow(ResultSet rs, int rowNum) throws SQLException {
Employee employee = new Employee();
employee.setFirstName(rs.getString("first_name"));
employee.setLastName(rs.getString("last_name"));
return employee;
}
});
```

Q:16 What NamedParameterJdbcTemplate in Spring?

The NamedParameterJdbcTemplate allow basic set of JDBC operations, it allows named parameter in the query instead of traditional (?) placeholder, the functionality is similar to JdbcTemplate class.

Example:

```
NamedParameterJdbcTemplate namedParameterJdbcTemplate;

String empInsrtQuery = "INSERT INTO Employee (name, age, salary) VALUES (:name, :age, :salary)";
Map namedParameters = new HashMap();
namedParameters.put("name", name);
namedParameters.put("age", age);
namedParameters.put("salary", salary);
namedParameterJdbcTemplate.update(empInsrtQuery, namedParameters);
```

Q: 17 What are Advice, Aspect, Join-point and point cut in spring?

Advice: An advice is an action taken by the aspect at particular join-point is called Advice.

Aspect: An aspect is a subprogram which is associated with specific property of a program (Example separating logging code from the main program).

An aspect is functionality or feature that cross cuts over object. AOP increase modularity of a program.

Join-Point: A join point is a point used in spring AOP framework to represent a method execution. It always point during execution of program, method or exception. A join point is basically an opportunity within the code to apply

aspect.

Point Cut: In AOP a point cut is a set of many join points where an advice can execute. A chunk of code (known as Advice) associated with join point get executed.

Q:18 What are the different types of Advice?

There are different types of Advice.

Before Advice: The advice which executed before a join point called before advice. The before advice does not have the ability to prevent the execution flow proceeding at the join point (unless it throws an exception).

After Return Advice: The advice which executed after a join point completed normally without any exception.

Around Advice: It is responsible for choosing whether to proceeds to the join point or shortcut the advised method execution by returning its own return value or throwing an exception. This is most powerful kind of advice. With Around advice you can perform custom behavior before and after method execution.

After throwing advice: The advice executed when a method throws an exception.

After (finally) advice: The advice is executed when program exits the join points either normally or by throwing an exception.

Q: 19 What is Weaving in Spring?

Weaving is the process of linking aspect with other application types or object to create an advised object. This can be performed at compile time, runtime and load time. In spring framework weaving is performed at runtime.

Q:20 What is AOP Proxy?

AOP proxy is an object to implement the aspect contracts (advice method executions and so on). The AOP proxy is object is created by the AOP framework. In spring framework AOP proxy is JDK dynamic proxy or CGLIB proxy.

Q: 21 What is front controller in Spring MVC?

The Front Controller is basically a type of Design pattern which are being implemented in different framework (e.g. Struts and Spring MVC etc.). In Spring MVC DispatcherServlet act as a Front Controller for the framework and responsible for intercepting every request and then dispatches/forwards request to appropriate controller. Configure the DispatcherServlet in the web.xml file of web application and request which we want to be handled by DispatcherServlet should be mapped using URL mapping. For example all requests ending with *.do will be handled by the DispatcherServlet.

```
<web-app>
<servlet>
<servlet-name>example</servlet-name>
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>example</servlet-name>
<url-pattern>*.do</url-pattern>
</servlet-mapping>
```

</web-app>

Q:22 Difference between FileSystemResource and ClassPathResource?

In FileSystemResource you need to give the configuration file (i.e. spring-config.xml) relative to your project or the absolute location of the file.

In ClassPathResource spring looks for the file in the ClassPath so configuration (i.e. spring-config.xml) file should be included in the classpath. If spring-config.xml is in classpath, you can simply give the name of the file.

For Example: If your configuration file is at src/main/java/com/test/loadresource then your FileSystemResource would be:

FileSystemResource resource = new FileSystemResource("src/main/java/com/test/loadresource/spring-config.xml");

And ClassPathResource would be:

ClassPathResource resource = new ClassPathResource("com/test/loadresource /spring-config.xml");

Q: 23 What is inner Bean Definition?

A bean definition added inside the property or constructor-arg elements are called inner bean.

Example:

```
<bean id="outerbean" class="...">
<!-- instead of using a reference to a target bean, simply define the target bean inline -->
<property name="targetbean">
<bean class="com.example.Person"> <!-- this is the inner bean -->
<property name="name" value="XYZ"/>
<property name="age" value="35"/>
</bean>
</property>
</bean>
```