Generic Questions

Project details

What is your role in project? Which modules have you worked on?

Were you involved in any design phase? Explain what things you put in design document?

Were you involved in Database design as well?

How to prepare design document and what are the things are to be put in them?

How Report are generated from Java ? (Related to report generation

Discussion regarding agile methodology and sprint , sum master etc.

Any experience on IBM servers?

Which build tools are used?

PROD/UAT support knowledge.

Idea on JIRA ?

Java Questions

What are the Java boundary or Transaction boundary?

<https://dzone.com/articles/spring-transaction-management>

)

Exception handling and types.

Hash map and uses.

Try catch finally concepts

Multithreading

Exception handling mechanism and how it can be implemented in an application?

How to store data in database using java? (Complete flow with exception handling)

What is JMS ?

What are EJBs ?

**Ques:** Testing Questions

**Ques:** How unit testing is done in project? Did you used Junit?

**Ques:** What is Junit ?

**Ques:** DB Questions

**Ques: How can we increase performance in Employee table to fetch particular data? (Performance tuning)**

Indexing is an effective way to tune your SQL database that is often neglected during development.

**Ques:** General idea about Pl/sql , oracle, proc

**Ques: Pl/SQL /Mongo /Angular if any knowledge.**

**Ques: Can you write basic sql queries? DDL,DML statements ?**

**Ques: Can you write stored procedures?**

A stored procedure is a set of SQL statements with an assigned name, which are stored in a [relational database management system](http://searchsqlserver.techtarget.com/definition/relational-database-management-system) as a group, so it can be reused and shared by multiple programs

To help you build powerful database applications, stored procedures provide several advantages including better performance, higher productivity, ease of use, and increased scalability.

**How to execute stored procedure from Java?**

# How to Call a Stored Procedure by JDBC Java Class?

A Stored procedure can return result sets, you can use getResultSet method in the CallableStatement class to retrieve return result sets. When a procedure has return value for an OUT parameter, you must tell the JDBC driver what SQL type the value will be, with the registerOutParameter method. To call stored procedures, you invoke methods in the CallableStatement class. The basic steps are:

* Creating a CallableStatement object by calling the Connection.prepareCall method.
* Using the CallableStatement.setXXX methods to pass values to the input (IN) parameters.
* Using the CallableStatement.registerOutParameter method to indicate which parameters are output-only (OUT) parameters, or input and output (INOUT) parameters.
* Invoke one of the following methods to call the stored procedure:
  + CallableStatement.executeUpdate method, if the stored procedure does not return result sets.
  + CallableStatement.executeQuery method, if the stored procedure returns one result set.
  + CallableStatement.execute method, if the stored procedure returns multiple result sets.
* Calling the CallableStatement.getResultSet method to obtain the result set (which is in a ResultSet object), if the stored procedure returns one result set. But if the stored procedure returns result sets, retrieve the result sets by combining CallableStatement.getResultSet and CallableStatement.getMoreResults methods.
* Using the CallableStatement.getXXX methods to retrieve values from the OUT parameters or INOUT parameters.
* Calling the CallableStatement.close method to close the CallableStatement object when you have finished using that object.

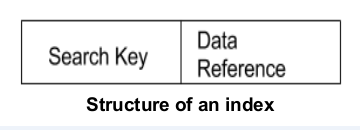
Connection con = getConnection();  
 CallableStatement cs = null;  
 try {  
 **cs = con.prepareCall("{call WhoAreThey(?,?)}");  
 cs.registerOutParameter(1, Types.VARCHAR);  
 cs.setString(2, office);  
 cs.execute();  
 String str = cs.getString(1);**  
 if (str != null) {  
 System.out.println(str);  
 }  
 else {  
 ResultSet rs = cs.getResultSet();  
 while (rs.next()) {  
 System.out.println("Name : " + rs.getString(2));  
 }  
 }

**Ques: Indexing**

<https://www.geeksforgeeks.org/indexing-in-databases-set-1/>

Indexing is a way to optimize performance of a database by minimizing the number of disk accesses required when a query is processed.

An index or database index is a data structure which is used to quickly locate and access the data in a database table



Indexes are special look up table that the database search engine can use to speed the data retrieval. Simple way we can define it as Index is pointer to data in a table.

**When should indexes be avoided?**

Although indexes are intended to enhance a database's performance, there are times when they should be avoided. The following guidelines indicate when the use of an index should be reconsidered.

* Indexes should not be used on small tables.
* Tables that have frequent, large batch updates or insert operations.
* Indexes should not be used on columns that contain a high number of NULL values.
* Columns that are frequently manipulated should not be indexed

Syntax:

CREATE INDEX index\_name ON table\_name (COLUMN1, COLUMN2…)

**Ques: From 1000 records how to fetch 200 records using hibernate?**

**Questions on other topics**

**Ques: Spring core IOC**

**Ques: Spring Boot and its advantage**

In Simple Terminology, What Spring Boot means

What Is Spring Boot, Spring Boot Tutorial

**Why Spring Boot?**

* To ease the Java-based applications Development, Unit Test and Integration Test Process.
* To reduce Development, Unit Test and Integration Test time by providing some defaults.
* To increase Productivity

**Advantages of Spring Boot:**

* It is very easy to develop Spring Based applications with Java or Groovy.
* It reduces lots of development time and increases productivity.
* It **avoids writing lots of boilerplate Code, Annotations and XML Configuration**.
* It is very easy to integrate Spring Boot Application with its Spring Ecosystem like Spring JDBC, Spring ORM, Spring Data, Spring Security etc.
* It follows “Opinionated Defaults Configuration” Approach to reduce Developer effort
* It provides **Embedded HTTP servers like Tomcat, Jetty** etc. to develop and test our web applications very easily.
* It provides **CLI (Command Line Interface) tool to develop** and test Spring Boot(Java or Groovy) Applications from command prompt very easily and quickly.
* It provides **lots of plugins to develop and test** Spring Boot Applications very easily using Build Tools like Maven and Gradle
* It provides lots of plugins to work with embedded and in-memory Databases very easily.

<http://www.springboottutorial.com/spring-boot-vs-spring-mvc-vs-spring>

1. Spring based applications have a lot of configuration.

When we use Spring MVC, we need to configure component scan, dispatcher servlet, a view resolver, web jars(for delivering static content) among other things

<bean

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix">

<value>/WEB-INF/views/</value>

</property>

<property name="suffix">

<value>.jsp</value>

</property>

</bean>

<mvc:resources mapping="/webjars/\*\*" location="/webjars/"/>

Below code snippet shows typical configuration of a dispatcher servlet in a web application.

<servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>

org.springframework.web.servlet.DispatcherServlet

</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/todo-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

When we use Hibernate/JPA, we would need to configure a datasource, an entity manager factory, a transaction manager among a host of other things.

<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"

destroy-method="close">

<property name="driverClass" value="${db.driver}" />

<property name="jdbcUrl" value="${db.url}" />

<property name="user" value="${db.username}" />

<property name="password" value="${db.password}" />

</bean>

<jdbc:initialize-database data-source="dataSource">

<jdbc:script location="classpath:config/schema.sql" />

<jdbc:script location="classpath:config/data.sql" />

</jdbc:initialize-database>

<bean

class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"

id="entityManagerFactory">

<property name="persistenceUnitName" value="hsql\_pu" />

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

</bean>

<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">

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

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

</bean>

<tx:annotation-driven transaction-manager="transactionManager"/>

Transaction management in spring? What is Transaction boundary?

Did you used Hibernate in your project?

**Ques: Spring features, Spring MVC flow. Why we use Spring framework?**

**Advantages of Spring Framework**

**1) Predefined Templates**

Spring framework provides templates for JDBC, Hibernate, JPA etc. technologies. So there is no need to write too much code. It hides the basic steps of these technologies.

Let's take the example of JdbcTemplate, you don't need to write the code for exception handling, creating connection, creating statement, committing transaction, closing connection etc. You need to write the code of executing query only. Thus, it save a lot of JDBC code.

2**) Loose Coupling**

The Spring applications are loosely coupled because of dependency injection.

**3) Easy to test**

The Dependency Injection makes easier to test the application. The EJB or Struts application require server to run the application but Spring framework doesn't require server.

**4) Lightweight**

Spring framework is lightweight because of its POJO implementation. The Spring Framework doesn't force the programmer to inherit any class or implement any interface. That is why it is said non-invasive.

**5) Fast Development**

The Dependency Injection feature of Spring Framework and it support to various frameworks makes the easy development of JavaEE application.

**6) Powerful abstraction**

It provides powerful abstraction to JavaEE specifications such as JMS, JDBC, JPA and JTA.

**7) Declarative support**

It provides declarative support for caching, validation, transactions and formatting.

<http://www.springboottutorial.com/spring-boot-vs-spring-mvc-vs-spring>

1. **What is the core problem that Spring MVC Framework solves?**

Spring MVC Framework provides decoupled way of developing web applications. With simple concepts like Dispatcher Servlet, ModelAndView and View Resolver, it makes it easy to develop web applications.

1. **What is the core problem that Spring Framework solves?**

Most important feature of Spring Framework is Dependency Injection. At the core of all Spring Modules is Dependency Injection or IOC Inversion of Control.

Why is this important? Because, when DI or IOC is used properly, we can develop loosely coupled applications. And loosely coupled applications can be easily unit tested.

1. **Duplication/Plumbing Code**

Does Spring Framework stop with Dependency Injection? No. It builds on the core concept of Dependeny Injection with a number of Spring Modules

· Spring JDBC

· Spring MVC

· Spring AOP

· Spring ORM

· Spring JMS

· Spring Test

Consider Spring JMS and Spring JDBC for a moment.

Do these modules bring in any new functionality? No. We can do all this with J2EE or JEE. So, what do these bring in? They bring in simple abstractions. Aim of these abstractions is to

· Reduce Boilerplate Code/ Reduce Duplication

· Promote Decoupling/ Increase Unit Testablity

For example, you need much less code to use a JDBCTemplate or a JMSTemplate compared to traditional JDBC or JMS.

1. **Good Integration with Other Frameworks.**

Great thing about Spring Framework is that it does not try to solve problems which are already solved. All that it does is to provide a great integration with frameworks which provide great solutions.

· Hibernate for ORM

· iBatis for Object Mapping

· JUnit & Mockito for Unit Testing

**Experience in Angular js, junits.**

**Did you used any tools to create Spring controllers?**

**Hibernate/ How to map tables with classes?**

**Hibernate entity class design**

**Angular JS**