Hexaware Interview

1. How to convert JSON object to String?

JSON- Java script object notation . It is an suitable data format to exchange data from servers. It is suitable for cross domain applications. It is an lightweight data interchange format. It is easy for machines to parse and generate. It is language independent

Structures:

It supports two structures.

1. An ordered key – value pairs is known as object
2. Sequence of values is known as arrays

JSON Objects:

Employee {

“name”:”Vasantha”,

“Age” : 34

}

* Key should be in string
* Value can be string , number, boolean null , array , object.
* The value can be access by Employee.name or Employee[‘name’]

JSON Array:

Employee[

{

“id” : 1,

“name”: “Vasantha”

},

{

“id”:2,

“Name”:”selvi”

}

]

The array values can be access by Employee[1].id

JSON Object Creation:

1. Empty object:

var jsonObject={};

1. New Object:

var jsonObject=new Object();

length attribute is used to find the length of the array.

Methods in JSON:

JSON.parse(jsonObject) :- convert the JSON object to java script object or java script array

JSON.stringnify(javascriptObject) :- convert the java script object to JSON object

* Date object is not allowed in string

1. What is purpose of JSONP ?

JSONP(JSON with Padding) is a method for sending the JSON data without worrying about the cross domain. It doesn’t use the XMLHttpRequest to send data , istead it use <script> tag.

1. How should map the form params in restful web services?

Using @FormParam

1. How should read the entire quers params in restful webservices?

Using the

1. What would the @SpringBootApplication replace for?

@SpringBootApplication is used to mark the class as configuration class that declares one or more @Bean definition and triggers the auto-configuration and component scanning. It replace the following annotations such as @Configuration , @ComponentPackage and @EnableAutoConfiguration.

Spring Boot Spring Application class is used to bootstrap and launch the application. It s suitable for Spring MVC and Spring –Rest webservices . it automatically creates the application context from the classpath , load the configuration classes and launch the application.

1. How spring security works ?

**Barclays Interview**

1. **What are transaction isolation level?**

Transaction management ensures that the integrity and consistency of the data in the relational management system. To ensure consistency and integrity data base must satisfy the ACID properties.

Atomicity :

Consistency :

Isolation :

Durability :

The isolation determines that ,how transaction integrity is visible to other transactions. Every transaction in database should take place in such a way that is the only transaction accessing the resources in that database.

The isolation level can be determined based on the following situations.

1. Dirty read: This occurs when transaction one read the data that is changed and not committed by transation2 . Later transaction 2 commits the data but the transacion1 holds the data that are not exist in database.
2. Non Repeatable Read: This occurs when transaction 1 read some data . Transaction 2 change the same data and commit it. Now transaction 1 read the same data , it gets different results.
3. Phantom read: This occurs when transaction 1 execute some queries based on certain condition and get the result. Then the transaction 2 inserts some data which satisfy the transaction 1 condition. Now transaction 1 executes the same query , which gets different set of results.

**Different Isolation Level:**

Based on the above criteria, there are four isolation level defined.

1. Read uncommitted :- It is the lowest isolation level . It allows dirty read . This allows one transaction can read the data that are not committed by other transaction
2. Read committed :- It doesn’t allow dirty read. It ensures that the transaction always read the committed data at the time of read. The transaction holds read or write lock on particular row. So it doenst allow any other transation to update , read or delete that row.
3. Repeatable read :- This avoids non repeatable read. This is the most restrictive isolation level. The transaction holds the read lock on the row it refere and write lock on the row it updates, inserts and deletes. So the other transaction cannot read , update or deleted these rows consequently.
4. Serializable :- This is the highest isolation level . It ensures that the concurrent transactions to be executed serially.

**Spring with transaction management:**

* Spring provides good support on handling the transactions and help developers more on concentrating on the business logic rather than handling the transation.
* Spring provides two types of transaction support.
* Spring support various transaction management API’s . JDBC ,Hibernate and JPA. We need to implement the proper implementation classes.
* When we create database connection , it runs in auto-commit mode

Ex: JDBC -> Driver Manager Data Source

Hibernate ->HiberateTransactionManager

1. Programmatic Transaction : This is supported by using the TransactionTemplate or PlatFormTransaction Manager
2. Declarative transaction : This is achieved by using spring AOP. Since spring AOP provides loose coupling between business logic and transaction management
3. Spring-tx depenedency provides the spring declarative transaction support
4. Always the transaction should be handled in Service layer by annotating the method or class as @Transactional

**import org.springframework.transaction.annotation.Transactional;**

import com.journaldev.spring.jdbc.dao.CustomerDAO;

import com.journaldev.spring.jdbc.model.Customer;

public class CustomerManagerImpl implements CustomerManager {

private CustomerDAO customerDAO;

public void setCustomerDAO(CustomerDAO customerDAO) {

this.customerDAO = customerDAO;

}

@Override

**@Transactional**

public void createCustomer(Customer cust)

{

customerDAO.create(cust);

}

}

Spring configuration file

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"

xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd">

**<!-- Enable Annotation based Declarative Transaction Management -->**

<tx:annotation-driven proxy-target-class="true"

transaction-manager="transactionManager" />

**<!-- Creating TransactionManager Bean, since JDBC we are creating of type**

**DataSourceTransactionManager -->**

<bean id="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

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

</bean>

**<!-- MySQL DB DataSource -->**

<bean id="dataSource"

class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName" value="com.mysql.jdbc.Driver" />

<property name="url" value="jdbc:mysql://localhost:3306/TestDB" />

<property name="username" value="pankaj" />

<property name="password" value="pankaj123" />

</bean>

<bean id="customerDAO" class="com.journaldev.spring.jdbc.dao.CustomerDAOImpl">

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

</bean>

<bean id="customerManager" class="com.journaldev.spring.jdbc.service.CustomerManagerImpl">

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

</bean>

</beans>

**<tx:annotation-driven> -** tell to the spring container that we are using annotation based transaction management configuration.

**transaction-manager –** attribute default value is transactionManager.

**Proxy-target-class –** attribute tells the spring container to use the class based proxies.

**Aricent Interview:**

1. How to connect to the remote DB from my system?
2. What is thread dump ?
3. How to set thread name in thread pool?
4. How to use thread pool ? What are the parameters needs to be set when creating the thread pool ?
5. When there is blocking in MQ for multiple request? How to handle it?
6. How to use mockito?
7. How to create microservices
8. What is ansible ?
9. In executer framework , if any issues in submit ? How to handle it?
10. What OAuth2 and persistent token factory ?
11. How to install ssl ?
12. How to deploy application in prod environment using maven ?
13. Difference between controller and rest controller

**Spring Security Authentication Types:**