Spring Certified #14

A question lead guide to prepare Spring certification

Spring Core

Which of the following annotations can be used to create a custom auto-configuration class in Spring Boot?Choose the BEST answer.

- ➔ **@AutoConfiguration**
- ➔ **@EnableAutoConfiguration**
- ➔ **@Configuration**
- ➔ **@ComponentScan**

@Configuration

Under the hood, auto-configuration is implemented with standard `@Configuration` classes.

Understanding Auto-configured Beans

Creating a Custom Auto-Configuration

In order to create a custom auto-configuration, we need to create a class annotated as *@Configuration* and register it.

Let's create a custom configuration for a *MySQL* data source:

```
@Configuration
public class MySQLAutoconfiguration {
    //…
}
```

https://www.baeldung.com/spring-boot-custom-auto-configuration#creating-a-custom-auto-configuration

Data Management

Which isolation levels are concerned by the Phantom reads phenomena? (select 3)

- ➔ **Read Uncommitted**
- ➔ **Read Committed**
- ➔ **Repeatable Reads**
- ➔ **Serializable**

Repeatable Reads, Read Committed, Read Uncommitted

Phantom Reads

A phantom read is a special case of fuzzy reads. This happens when another session inserts or deletes rows that match the where clause of your query. So repeated queries can return different rows:

Transaction 1 start
insert into bricks (colour, shape) values ('red', 'cube');commit;
select shape from bricks where colour = 'red'; //outputs: cube
Transaction 2 meanwhile
insert into bricks (colour, shape) values ('red', 'pyramid');commit;
Transaction 1 end
select shape from bricks where colour = 'red';// outputs:cube,pyramid

	Dirty Reads	Non-repeatable Reads	Phantom Reads
Read Uncommitted	✔	✔	✔
Read Committed	✘	✔	✔
Repeatable Reads	✘	✘	✔
Serializable	✘	✘	✘

Isolation Levels

To help you manage which read problems you're exposed to, the SQL standard defines four isolation levels. These state which phenomena are possible, as shown by this table:

https://livesql.oracle.com/apex/livesql/file/tutorial_GXA9ZDN9ODAIUOHO5LRWCPPQT.html

Data Management

Which comparison operator can be used in Spring Data JPA Repository Interface finder method declarations to produce a valid database query? (select 2)

- ➔ **Equals**
- ➔ **Plus**
- ➔ **GreaterThan**
- ➔ **InstanceOf**

Equals, GreaterThan

Good answers:

- "Equals": the "Equals" comparison operator can be used to generate a query that matches entities with a specific field value.
- "GreaterThan": the "GreaterThan" comparison operator can be used to generate a query that matches entities with a field value greater than a given value.

Wrong answers:

- "Plus": "Plus" is not a comparison operator, it's an arithmetic operator for addition.
- "InstanceOf": "InstanceOf" is not a comparison operator, it's a keyword used for type checking.

https://www.baeldung.com/spring-data-derived-queries

https://thorben-janssen.com/ultimate-guide-derived-queries-with-spring-data-jpa/

https://bit.ly/2v7222

https://spring-book.mystrikingly.com