Interview Question

RestFul web Services

CrudRepositiory Vs JPA Reposotiry

Friday, January 7, 2022 10:47 AM spring MVC

POSTMAN ------->REST-controller ---------->JPA repository ------>Hibernate ---->MSQL Driver --------->Database

SPRIGN provides these two interfaces -
Interface CrudRepository (count, delete, exists, findBy , save ----------basic crud operations)
|
|
Interface JPARepository sub interface (batch wise crud operations + pagination +sorting + flushing }}} API)

Pagination = 1000000 rows !!!--- break the chunk into parts ----pages of some size/limit ,maintain all the chunks in a sequence.Then after every next the next chunk should be given

Pagination happens at JPA repository level (server side)
Pagination happens at Browser level (client side)

_____

CRUDRepository (I)
|
|
JPARepository (I)
|
|
MyRepository (I)

Spring boot provides implementation of all methods of MyRepository and creates a Bean on that Impl class!!!
When we are AutoWiring MyRepositary --- this bean is INJECTED in the property reference !!!

_____

Different types of find methods ?

1. Find by the primary key = always yields **ONE** or ZERO result ----- findById() , getById()
2. Find all records = yield ZERO or **all** N records in the table -- findAll
3. Find by some other column value = yields 0 or **few matching records ---- add a method in MyRepository List<EntityType> findByColumnName(CoulumnType parameter)**
   Select * from book where bookName = "XYZ"
   Spring framework gives the auto impl of the above method !!!

4. Find by some other condition other than = !!!
   Use @Query
   Use JPQL------> HQL or Native Query

_____

HW ----
Add finder methods for all columns
----------and also add custom queries that searches for
product with cost less than ,
cost greater than ,
expiry date reached,
expiry date not reached (add the expiry date to the ProductEntity)

_____

RESTFul WEB SERVICES , **REST API**

1. We can call them from the POSTMAN
2. When you do JavaScript and REACT you can call REST API from AJAX calls !!!
3. When you write a core java client , it can also act as a REST Client

Optional -HW ---- create a new spring starter project with spring web
In the main simply write the RestTemplate and call all the CRUD APIs that you are calling from POSTMAN

_____

Interview Point

Spring MVC ----------

M = Model = that part of the application which has business logic and data operations (Utils, DAO, DO , PocessingClasses …)
V = View = that part of the application which has presentation logic (JSP , Thymeleaf)
C = Controller = that part of the application that CONNECTS Model to View (controller MAPS model and view)

Servlet -JSP -JDBC
M = DAO ,DO
V = JSP (html, css)
C = Servlet (can be used to accept all requests and then forward them to respective JSP (
LoginServlet
|
|
DAO used to get records of USER
|
|---logic to verify user
Forward to home.jsp or login.jsp with error

Controller DECOUPLES model and View
So that **Loose coupling** of model and view is achieved

M----------C----------V

HTML--> View

Controller -> Servlet JSP

Model->DAO,DO Classes

Spring M = Spring Beans (DAOBean) model
V = JSP view
C = We write a controller class that maps Model and View controls

---

Steps
1. create a spring starter using spring web dependency !!
2. configure a view resolver in application.properties
3. write a JSP and add it to main/webapp folder
4. write a controller

Whenever we run a spring application on web ---
Spring Application Web Context is created ---- this manages the lifecycle of all the beans!!!

ROLE of ServletInitializer ……

Tomcat -- web.xml ---------- general redirect to a DispatcherServlet !!
---WAR1 file
---WAR2 file
---WAR3 file

<servlet>
<servlet-name> A
<servlet-class> **org.springframework…….DispatcherServlet** </servlet-class> (the control is passed to SpringWebApplicationContext)
</servlet>

<servlet-mapping>
<servlet-name>A
<url-pattern> / </url-pattern> (**This is mapped to all requests**)
</…>

HW 3----------
STEPS to write a MVC application ---
1. Create a starter project , add spring web
2. Add following to the POM file

```
<dependency>
   <groupId>org.apache.tomcat.embed</groupId>
   <artifactId>tomcat-embed-jasper</artifactId>
</dependency>
```

3. add the following to application.properties file
   ```
   spring.mvc.view.prefix=WEB-INF/view/
   spring.mvc.view.suffix=.jsp
   ```

4. add the controller class with @Controller annotation
5. add a mapping method sayHello that returns the name of the view
6. Create the view file in src/main/webapp/web-inf/view folder
7. run the public static void main class
8. Access the url from POSTMAN and browser