Assignment

1.JSON

A. JAVASCRIPT OBJECT NOTATION(JSON):--

- JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax.
- It is commonly used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa).

- JSON is a **text format** for storing and transporting data.

- JSON is "self-describing" and easy to understand.

- JSON is a lightweight data-interchange format. JSON is plain text written in JavaScript object notation
- JSON is used to send data between computers. JSON is language independent.

- This example is a JSON string:

'{"name":"John", "age":30, "car":null}'

- It defines an object with 3 properties:
  - Name
  - Age
  - car
- Each property has a value.
- If you parse the JSON string with a JavaScript program, you can access the data as an object:

```
let personName = obj.name;
let personAge = obj.age;
```

Why Use JSON?

- The JSON format is syntactically similar to the code for creating JavaScript objects. Because of this, a JavaScript program can easily convert JSON data into JavaScript objects.
- Since the format is text only, JSON data can easily be sent between computers, and used by any programming language.
- JavaScript has a built in function for converting JSON strings into JavaScript objects:

  JSON.parse()

- JavaScript also has a built in function for converting an object into a JSON string:
  JSON.stringify()

2.JSON Array

A

- JSON is a lightweight data-interchange format. JSON is plain text written in JavaScript object notation
- JSON is used to send data between computers JSON is language independent
- Arrays in JSON are almost the same as arrays in JavaScript
- In JSON, array values must be of type string, number, object, array, boolean or *null*.
- In JavaScript, array values can be all of the above, plus any other valid JavaScript expression, including functions, dates, and *undefined*.
- You can create a JavaScript array from a literal:

Example

```
myArray = ["Ford", "BMW", "Fiat"];
```

3.Controller and RestController

A Controller:

- The `@Controller` is a common annotation which is used to mark a class as Spring MVC Controller
- `@Controller` is an old annotation, exists since Spring started supporting annotation, and officially it was added on Spring 2.5 version.
- The `@Controller` annotation indicates that the class is a “Controller” e.g. a web controller
- The `@Controller` is a specialization of `@Component` annotation

RestController:

- The `@RestController` is a special controller used in RESTFul web services and the equivalent of `@Controller + @ResponseBody`.

- The `@RestController` is relatively new, added only on Spring 4.0
- while the `@RestController` annotation indicates that the class is a controller where `@RequestMapping` methods assume `@ResponseBody` semantics by default i.e. servicing REST API.
- while `@RestController` is a specialization of `@Controller` annotation. It is actually a convenience controller annotated with `@Controller` and `@ResponseBody`

4.Soap and Restful Webservices

A. SOAP (Simple Object Access Protocol) is a standards-based web services access protocol that has been around for a long time. **REST** (Representational State Transfer) is another standard, made in response to SOAP's shortcomings. It seeks to fix the problems with SOAP and provide a simpler method of accessing web services.

Soap webservices:

- SOAP stands for **Simple Object Access Protocol.**
- SOAP is a **protocol**.
- SOAP **can't use REST** because it is a protocol.
- SOAP uses services interface to expose the business logic.
- **JAX-WS** is the java API for SOAP web services.
- SOAP defines standard to be strictly followed.
- SOAP **requires more bandwidth** and resource than REST.
- SOAP defines its own security.
- SOAP **permits XML** data format only.
- SOAP is less preferred than REST

Restful Webservices:

- REST stands for Representational State Transfer
- REST is an **architectural style**.
- REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP.
- REST uses URI to expose business logic.
- **JAX-RS** is the java API for RESTful web services.
- REST does not define too much standards like SOAP.
- REST **requires less bandwidth** and resource than SOAP.
- RESTful web services inherits security measures from the underlying transport.

- REST **permits different** data format such as Plain text, HTML, XML, JSON etc.
- REST more preferred than SOAP.

5.What is the difference between Web Application and WebService Application ?

Web Services	Web Application
❖ Web services are a type of API, which must be accessed through a network connection.	❖ APIs are application interfaces, implying that one application can communicate with another application in a standardized manner.
❖ All Web services are APIs.	❖ APIs are not web services.
❖ It provides supports only for the HTTP protocol.	❖ It provides support for the HTTP/s protocol: URL Request/Response Headers, and so on.
❖ Web service supports only XML.	❖ API supports XML and JSON.
❖ Web Services can be hosted on IIS.	❖ Web API can be hosted only on IIS and self.
❖ It is not open source, however can be devoured by any customer that comprehends xml	❖ It is open source and also ships with .NET framework
❖ It doesn't have lightweight design, needs a SOAP convention to send or receive data over the system.	❖ It has a light-weight architecture furthermore, useful for gadgets which have constrained transmission capacity like smart phone.
❖ Webservice is used for REST,SOAP and XML-RPC for communication.	❖ API is used for any style of communication

6. Response body and response entity

A @ResponseBody:

- @ResponseBody is a **Spring annotation which binds a method return value to the web response body**. It is not interpreted as a view name.
- It uses HTTP Message converters to convert the return value to HTTP response body, based on the content-type in the request HTTP header
- With **@ResponseBody**, *only the body is returned*. The headers and status code are provided by Spring.
- @ResponseBody puts the return value into the body of the response
- The REST API sends a response header and response body in JSON format with information about the success or failure of the REST API call.

The response header can contain the following information:
server
content-type
content-language
content-length
date
connection
transfer-encoding
severity
errorCode
description
logFile

- The response body can contain information about the output and additional output ports. If the output type was file, the response body contains a path to the output. If the output type was buffer, the response body contains the buffer contents.
- The response body can also contain a success or failure code and message The response body consists of the resource data requested by the client.
- In our example, we requested the book's data, and the response body consists of the different books present in the database along with their information.
- If you put **@RequestBody** annotation aside the parameter in a method, Spring will convert the http request body to that declare class type in the method signature.

@ResponseEntity:

- **ResponseEntity** extends HttpEntity but also adds a Http status code.

- **HttpEntity** represents an HTTP **request** or **response** consists of **headers** and **body**.
- **ResponseEntity<>** is a generic class with a type parameter, you can specify what type of object to be serialized into the response body.
- you can set headers using **ResponseEntity<>**

ResponseEntity *represents an HTTP response, including headers, body, and status* in a spring restful API. While @ResponseBody puts the return value into the body of the response of API, **ResponseEntity** also allows us to add headers and status code as well.

7. What is DDL and What is DDL-auto ?

A DDL (Data Definition Language):

- DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema.
- It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.
- DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.

DDL commands:

CREATE: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).

DROP: This command is used to delete objects from the database.

ALTER: This is used to alter the structure of the database.

TRUNCATE: This is used to remove all records from a table, including all spaces allocated for the records are removed.

COMMENT: This is used to add comments to the data dictionary.

RENAME: This is used to rename an object existing in the database.

DDL-auto:-

- hbm2ddl.auto is a hibernate configuration property. It is used to validate and exports schema DDL to the database when the SessionFactory is created.
- By defining the hbm2ddl.auto property, we can execute the DDL (Data Definition Language) commands from the hibernate framework, while creating the SessionFactory itself.
- The **hbm2ddl.auto** property of Hibernate either creates or validates a database table.

possible values for hbm2ddl.auto:

- create
- validate
- update
- create-drop

If the value is **Create** then the hibernate first drops the existing tables data and structure, then creates new tables and executes the operations on the newly created tables.

If the value is ***validate*** then hibernate only validates the table structure- whether the table and columns have existed or not. If the table doesn't exist then hibernate throws an exception.

Validate is the default value for hbm2ddl.auto.

If the value is ***update*** then, Hibernate checks for the table and columns. If a table doesn't exist then it creates new tables and where as if a column doesn't exist it creates new columns for it.

If the value is ***create-drop*** then, Hibernate first checks for a table and do the necessary operations and finally drops the table after all the operations were completed.

8. JPARepository,pagingAndSortingRespository and CRUDRepository ?

A JPARepository:

- JPA also provides some extra methods related to JPA such as delete records in batch and flushing data directly to a database.
- JPA extend crudRepository and PagingAndSorting repository
- JPA repository also extends the PagingAndSorting repository. It provides all the method for which are useful for implementing pagination
- JPARepository ties your repositories to the JPA persistence technology so it should be avoided.

CRUDRepository:

- It provides only CRUD functions like findOne, saves, etc.
- Crud Repository is the base interface and it acts as a marker interface.
- Crud Repository doesn't provide methods for implementing pagination and sorting.
- We should use CrudRepository or PagingAndSortingRepository depending on whether you need sorting and paging or not.

pagingAndSortingRespository:

By extending PagingAndSortingRepository interface to define our repository, we can use the methods related to sorting and pagination defined by this interface. This interface is a sub-interface of CrudRepository.

```
package com.logicbig.example;

import org.springframework.data.repository.PagingAndSortingRepository;

public interface EmployeeRepository extends
PagingAndSortingRepository<Employee, Long> {

}
```

9. What are Mappings one-to-one, one-to-many, many-to-one, many-to-many?

A Mapping: The process to convert a request and the result from one level to another level is known as Mapping. The mapping defines the correspondence between three levels. The mapping description is also stored in data dictionary in the same file of data.

- The One-To-One mapping represents a single-valued association where an instance of one entity is associated with an instance of another entity. In this type of association one instance of source entity can be mapped atmost one instance of target entity.
- The One-To-Many mapping comes into the category of collection-valued association where an entity is associated with a collection of other entities. Hence, in this type of association the instance of one entity can be mapped with any number of instances of another entity.
- The Many-To-One mapping represents a single-valued association where a collection of entities can be associated with the similar entity. Hence, in relational database any more than one row of an entity can refer to the similar rows of another entity.
- The Many-To-Many mapping represents a collection-valued association where any number of entities can be associated with a collection of other entities. In relational database any number of rows of one entity can be referred to any number of rows of another entity.

10.What are relations in parent child table is-a, uses-a,has-a ?

A Is-A Relationship in Java

In Java, an Is-A relationship depends on inheritance. Further inheritance is of two types, class inheritance and interface inheritance. It is used for code reusability in Java. For example, a Potato is a vegetable, a Bus is a vehicle, a Bulb is an electronic device and so on. One of the properties of inheritance is that inheritance is unidirectional in nature. Like we can say that a house is a building. But not all buildings are houses. We can easily determine an Is-A relationship in Java. When there is an extends or implement keyword in the class declaration in Java, then the specific class is said to be following the Is-A relationship.

Has-A Relationship in Java

In Java, a Has-A relationship is also known as composition. It is also used for code reusability in Java. In Java, a Has-A relationship simply means that an instance of one class has a reference to an instance of another class or an other instance of the same class. For example, a car has an engine, a dog has a tail and so on. In Java, there is no such keyword that implements a Has-A relationship. But we mostly use new keywords to implement a Has-A relationship in Java.

11.What is Transient in JPA ?

A @Transient in JPA:

- @Transient annotation is used to ignore a field to not persist in database in JPA, where as transient key word used to ignore a field from serialization.
- **The uses of @Transient are:**
- Transient is a variables modifier used **in serialization**.
- At the time of serialization, if we don't want to save value of a particular variable in a file, then we use transient keyword.
- When JVM comes across transient keyword, it ignores original value of the variable and save default value of that variable data type
- The field annotated with @Transient still can be serialized, but the field declared with transient keyword not to be persisted and not to be serialized.
- And also @Transient can be used for property access as well, where as transient keyword allowed to use only for fields.
- To ignore fields to not persist in DB, **@Transient** annotation recommended to use, because it is specific to persistence.
- In some cases you may need to save the object state even though the fields are ignoring to not persist, which is not possible for the ***transient*** fields (fields are declared with **transient** keyword).