Thursday, January 6, 2022 10:32 AM

JSON = Java Script Object Notation

Book object bookId=90, bookCost=100, bookName="learning rest "

Representations of DATA ----------

JSON object format --

```
{
        "bookId" :"90",
         "bookCost":"100" ,
        "bookName":"Learning REST"
}
```

XML format

```
<book>
        <bookId>90</bookId>
        <bookCost>100</bookCost>
        <bookName>Learning Rest</bookName>
</book>
```

Array of books ----------

Json

```
[
{
        "bookId" :"90",
         "bookCost":"100" ,
        "bookName":"Learning REST"
},
{
        "bookId" :"90",
         "bookCost":"100" ,
        "bookName":"Learning REST"
},
{
        "bookId" :"90",
         "bookCost":"100" ,
        "bookName":"Learning REST"
}
]
```

Many books in XML

```
<books>
<book>
        <bookId>90</bookId>
        <bookCost>100</bookCost>
        <bookName>Learning Rest</bookName>
</book>
```

```
<book>
	<bookId>90</bookId>
	<bookCost>100</bookCost>
	<bookName>Learning Rest</bookName>
</book>
<book>
	<bookId>90</bookId>
	<bookCost>100</bookCost>
	<bookName>Learning Rest</bookName>
</book>

</books>
```

If Server side REST API SERVICE returns an Object ----

|

|----Java Object	Product obj = new Product("Cheese","slice",200);

|

| MessageConvertor

|

JSON (TEXT FORMAT)	<book> <bookId>90</bookId> <bookCost>100</bookCost> <bookName>Learning Rest</bookName> </book>

-------|

|

|

Client

DB Connectivity ---

1. DataSource ------ Jdbc
2. JPA

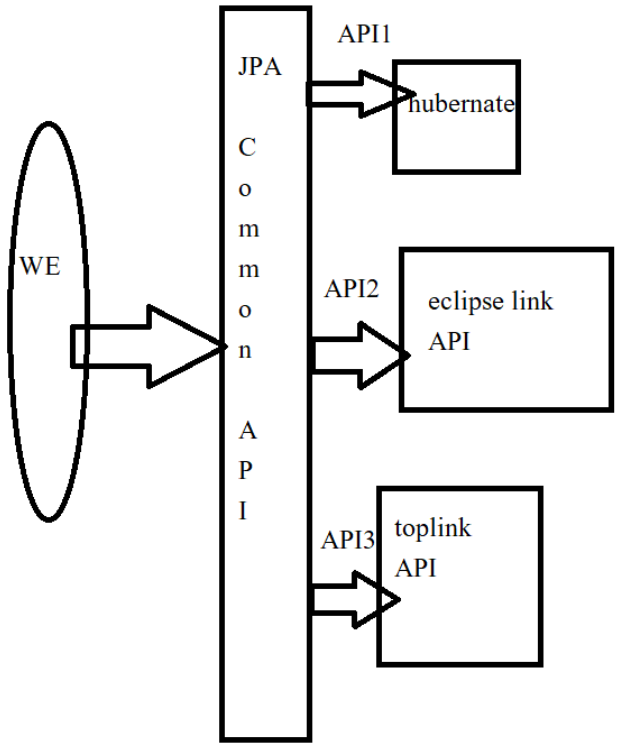
@Component / @Service /@Repository }}} tell spring container to create a bean !!!!

```
BookDAO
{
	@Autowired
	DataSource ds;


	addBook()
	{

	}
}
```

I want to see all records in book table on the postman [GET "/book/allBooks"] in JSON format !!!!!!!!

MANY ORMs
----- DON'T follow standard
----- DIFFERENT API
---Programmer has to RELEARN !!! --- BIG DISADVANTAGE !!!

To overcome the above disadvantage , JEE brought in JPA !!!
JPA = Java Persistence API !!!
COMMON API for all ORMs



ONLY JPA cannot work ! It always needs some ORM to perform jobs

---

We want to add record in book table using [POST /book/add/23/444/gdfgd]
We want to see all books using [GET /book/getAll]

Interact with the DB using hibernate ORM wrapped in JPA !!

STEP 1
Project should include
Spring-Web , Spring-DATA-JPA, MYSQL-Connector
STEP 2
Provide hibernate configuration in application.properties file in main/resources
STEP 3
Implement an interface that extends from a SPRING boot Interface JPARepository
STEP 4
Write the Entity class

HW ----

Product table ---------- id, cost, name, information

WRITE A CRUD application that will add , delete , update cost of a product
Show a particular product info and show all products

Use Spring boot JPA and REST Controller

ProductEntity , MyProductRepository interface , MyProductDAO (optional) ,
MyProductController

add [POST /product/add/id/cost/name/info]
showProduct [GET /product/show/id]
showAll [GET /product/showall]
Delete [DELETE /product/remove/id]
Update [UPDATE /product/change/id/newcost]

DAO LEVEL

Repo.save for add

For delete

1. **Entity = findById**
   repo.delete(Entity)

For Update

1. **Entity = findById()**
   Entity.setCost()

For showProduct

Entity = findById
Return entity

For showAll

repo.findAll()
