jsp - java server page -->

it is used for creating dynamic web content

allows embedding java code in html via jsp tags

kinda say presentation layer for web app

jsp r compiled in servlet via server

in short , for dynamic web app , via embedding java code in html

cons are hard to maintain as mixing of everything

servlet -->

java programs run on the server , handles request and responses, controls flow

it is an object which take req and generate res based on tye of req

within servlet container(like tomcat)

promotes mvc architecture

cons required to setup configs

jdbc - java database connectivity -->

kinda api to connect and execute queries with database

operations like crud with db

provides set of interfaces and classes

cons boiler plate for db access

<--- using jdbc ---->>

1 import packages

2 load drivers (in old versions)

3 register drivers

4 create connection

5 create statement or command (check statement types)

6 execute the statement

url , passwords, user in connection

jdbc vs jpa

jdbc kind a say java implementation playing with or having work with other techs when interact with relational databases

issue is managing overtime

after jdbc template came in picture

jpa – java persistence api is a collection of classes and method used for storing and retrieving large data in database

presentation layer 🡪 logic/code 🡪 jpa or .xml 🡪 database

to store business logics entities as relational entities , 🡪 plain old java object -> as entities , how to manage

entityManagerFactory 🡪creates and manages multiple entityManager instance

entityManager 🡪 perform persistence entity on object , kind a we say factory for query instance

entity 🡪 is a persistence object , stores as record in dB

entityTransaction 🡪kinda one-to–one relation with entity manager, its operations managed by this class

persistence 🡪contains methods for entityManagerFactroy

query 🡪 to obtains relational objects to meet criteria

entityManagerFactory relation with entity manager is one-to-many

entityManager relation with entityTransaction is one-to-one

entityManager relation with query is one-to-many

entityManager relation with entity is also one-to-many

orm (object relational mapping)

ability to convert data from object to relational type or vica-versa

in jpa , we divide this in different things 🡪

firstly , it has object data , contains pojo classes, services interface and classes

like its business face do getting setting , cruds

secondly , like mapping or persistence , contains jpa provider, mapping file (orm.xml) , jpa loader , object grid

jpa provider – javax.persistence file (eclipselink , toplink, hibernate

mapping file – orm.xml file contains config btw pojo class and relational database

jpa loader -- like a cache memory in browser , makes a copy of db so interaction for service classes for pojo data or attributes of pojo class

object grid – basically a temp location for db , queries affect this first then to main one

annotations 🡪

@entity - declare class as entity or table

@id – use as indentity for class (primary key in table)

@table – declare table name

@basic – for non cpntraint feilds

@embedded – to mark a class that can contain data or used in other

@generatedvalue – how indentity attribute can be initialized

@transient - which value not be stored in database

@column – to specify column

@sequencegenrator - generate sequence

@tablegenerator – create table for value generation

@accesstype – specifies access type like by field or property   
@joincolumn - in entity collections  
@uniquecontraints – specifies field as unique for tables

@columnresult – as select clause

@manytomany – define relations in join tables

@onetomany - define relations in join tables

@manytoone - define relations in join tables

@onetoone - define relations in join tables

@namedqueries – list of named queries

@namedquery – using query as static name

Now things are like jpa , spring data jpa , hibernate these are different but corelated in some ways . hibernate is implementation of jpa , spring data jpa is abtraction on jpa and hibernate which make much easier to work

Jpa Spring data

Spring.datasource.url

Spring.datasource.username

Spring.jpa.password

Spring.jpa.hibernate.ddl-auto 🡪 control behaviour for schema database creation -- none ,validate, create-drop, create, update

Spring.jpa.show-sql=true 🡪 show in console

Spring.jpa.properties.hibernate.dialect

Spring.jpa.properties.hibernate.format\_sql