# 7-july-2021

Task

# What is RDBMS?

**RDBMS** stands for **Relational DataBase Management Systems**. It is basically a program that allows us to create, delete, and update a relational database. **It is a database system that stores and retrieves data in a tabular format organized in the form of rows and columns.** It is a smaller subset of DBMS which was designed by E.F Codd in the 1970s. The major DBMS like SQL, My-SQL, ORACLE is based on the principles of relational DBMS.

 It has the capability to handle larger magnitudes of data and simulate queries easily.

Relational Database Management Systems maintains data integrity by simulating the following features:

* **Entity Integrity:** No two records of the database table can be completely duplicate.
* **Referential Integrity:** Only the rows of those tables can be deleted which are not used by other tables. Otherwise, it may lead to data inconsistency.
* **User-defined Integrity:** Rules defined by the users based on confidentiality and access.
* **Domain integrity:** The columns of the database tables are enclosed within some structured limits, based on default values, type of data or ranges.

**Characteristics**

* Data must be stored in tabular form in DB file, that is it should be organized in the form of rows and columns.
* Each row of table is called record/tuple . Number of such records is known as the cardinality of the table
* Each column of the table is called an attribute/field. Number of such columns is called the arity of the table.
* Number two records of the DB table can be same. Data duplicity is therefore avoided by using a candidate key. Candidate Key is a minimum set of attributes required to identify the set of records uniquely.
* Tables are related to each other with the help for foreign keys.
* Database tables also allow NULL values, that is if the values of any of the element of the table are not filled or are missing, it becomes a NULL value, which is not equivalent to zero.

**Advantages**

* Easy to manage: Each table can be independently manipulated without affecting others.
* Security: It is more secure consisting of multiple levels of security. Access of data shared can be limited.
* Flexible: Updation of data can be done at a single point without making amendments at multiple files.
* Users: RDBMS supports client-side architecture storing multiple users together.
* Data fetching is faster because of relational architecture.
* Data redundancy or duplicity is avoided due to keys, indexes, and normalization principles.
* Data redundancy or duplicity is avoided due to keys, indexes, and normalization principles.

Disadvantages

* High Cost and Extensive Hardware and Software Support: Huge costs and setups are required to make these systems functional.
* Scalability: In case of addition of more data, servers along with additional power, and memory are required.

#### Difference between DBMS and RDBMS

|  |  |
| --- | --- |
| 1. It stores data as a file | 1. It stores data in tabular form which consist of rows and columns |
| 1. Data Redundacy is present in DBMS | 1. RDBMS uses normalization to remove the data redundancy. |
| 1. In DBMS as data redundancy is there so it is difficult to modify the data at all the locations. If we make changes at one location and forget to make the changes at other places then there will be two different values for the same data. | 1. in RDBMS there is negligible data redundancy so it is easy to make the changes. We just make the changes in one place and our work is done. |
| 1. In DBMS the data is stored such that there is no relationship between data | 1. RDBMS the data is stored in the form of tables which are related to each other with the help of foreign keys |

# Normalization:

* Normalization is the process of organizing the data in the database.
* Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
* Normalization divides the larger table into the smaller table and links them using relationship.
* The normal form is used to reduce redundancy from the database table.

why is this required? without Normalization in SQL, we may face many issues such as:

1. **Insertion anomaly**: It occurs when we cannot insert data to the table without the presence of another attribute
2. **Update anomaly**: It is a data inconsistency that results from data redundancy and a partial update of data.
3. **Deletion Anomaly**: It occurs when certain attributes are lost because of the deletion of other attributes.

## Types of Normal Forms:



### **1st Normal Form (1NF)**

* A relation will be 1NF if it contains an atomic value.
* It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
* First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

In this Normal Form, we tackle the problem of atomicity. Here atomicity means values in the table should not be further divided. In simple terms, a single cell cannot hold multiple values. If a table contains a composite or multi-valued attribute, it violates the First Normal Form.



In the above table, we can clearly see that the Phone Number column has two values. Thus it violated the 1st NF. Now if we apply the 1st NF to the above table we get the below table as the result.

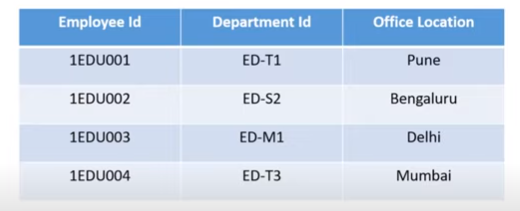


By this, we have achieved atomicity and also each and every column have unique values.

## **2nd Normal Form (2NF):**

The first condition in the 2nd NF is that the table has to be in 1st NF. The table also should not contain partial dependency. Here partial dependency means the proper subset of candidate key determines a non-prime attribute. To understand in a better way let’s look at the below example.

Consider the table



This table has a composite [primary key](https://www.edureka.co/blog/primary-key-in-sql/) **Emplyoee ID**, **Department ID**. The non-key attribute is **Office Location**. In this case, **Office Location** only depends on **Department ID**, which is only part of the primary key. Therefore, this table does not satisfy the second Normal Form.

To bring this table to Second Normal Form, we need to break the table into two parts. Which will give us the below tables:



As you can see we have removed the partial functional dependency that we initially had. Now, in the table, the column **Office Location** is fully dependent on the primary key of that table, which is **Department ID**.

Now that we have learnt 1st and 2nd normal forms lets head to the next part of this Normalization in SQL article.

## **3rd Normal Form (3NF)**

The same rule applies as before i.e, the table has to be in 2NF before proceeding to 3NF. The other condition is there should be no transitive dependency for non-prime attributes. That means non-prime attributes (which doesn’t form a candidate key) should not be dependent on other non-prime attributes in a given table. So a transitive dependency is a functional dependency in which X → Z (X determines Z) indirectly, by virtue of X → Y and Y → Z (where it is not the case that Y → X)

Let’s understand this more clearly with the help of an example:



In the above table, **Student ID** determines **Subject ID**, and **Subject ID** determines **Subject**. Therefore, **Student ID** determines **Subject** via **Subject ID.**This implies that we have a transitive functional dependency, and this structure does not satisfy the third normal form.

Now in order to achieve third normal form, we need to divide the table as shown below:

# What is Normalization in SQL and what are its types?

Data in [Database](https://www.edureka.co/blog/what-is-sql/) is stored in terms of enormous quantity. Retrieving certain data will be a tedious task if the data is not organized correctly. With the help of Normalization, we can organize this data and also reduce the redundant data. Through the medium of this article, I will give you a complete insight of  Normalization in SQL.

The following topics will be covered in this article:

* [What is Normalization in a Database?](https://www.edureka.co/blog/normalization-in-sql/#normalization)
* [What are 1NF, 2NF, 3NF and BCNF Normal Forms?](https://www.edureka.co/blog/normalization-in-sql/#1stNF)
  + [1st Normal Form (1NF)](https://www.edureka.co/blog/normalization-in-sql/#1stNF)
  + [2nd Normal Form (2NF)](https://www.edureka.co/blog/normalization-in-sql/#2ndNF)
  + [3rd Normal Form (3NF)](https://www.edureka.co/blog/normalization-in-sql/#3rdNF)
  + [Boyce-Codd Normal Form(BCNF)](https://www.edureka.co/blog/normalization-in-sql/#BCNF)

**What is Normalization in a Database?**

It is the processes of reducing the redundancy of data in the table and also improving the data integrity. So why is this required? without [Normalization](https://en.wikipedia.org/wiki/Database_normalization) in SQL, we may face many issues such as

1. ***Insertion anomaly***: It occurs when we cannot insert data to the table without the presence of another attribute
2. ***Update anomaly***:  It is a data inconsistency that results from data redundancy and a partial update of data.
3. ***Deletion Anomaly***: It occurs when certain attributes are lost because of the deletion of other attributes.

In brief, normalization is a way of organizing the data in the database. Normalization entails organizing the columns and tables of a database to ensure that their dependencies are properly enforced by [database](https://www.edureka.co/blog/what-is-mysql/) integrity constraints.

It usually divides a large table into smaller ones, so it is more efficient. In 1970 the First Normal Form was defined by Edgar F Codd and eventually, other Normal Forms were defined.

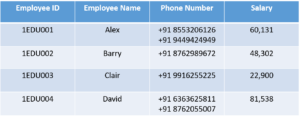
One question that arises in between is, what does [SQL](https://www.edureka.co/blog/sql-basics/) have to do with Normalization. Well [SQL](https://www.edureka.co/blog/what-is-sql/) is the language that is used to interact with the database. To initiate any interaction the data present in the database has to be of Normalized Form. Else we cannot proceed further as it results in anomalies.

Normalization in SQL will enhance the distribution of data. Now let’s understand each and every Normal Form with examples.

### NF - Normalization in SQL - Edureka

### **1st Normal Form (1NF)**

In this Normal Form, we tackle the problem of atomicity. Here atomicity means values in the table should not be further divided. In simple terms, a single cell cannot hold multiple values. If a table contains a composite or multi-valued attribute, it violates the First Normal Form.



In the above table, we can clearly see that the Phone Number column has two values. Thus it violated the 1st NF. Now if we apply the 1st NF to the above table we get the below table as the result.

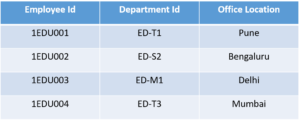


By this, we have achieved atomicity and also each and every column have unique values.

### **2nd Normal Form (2NF)**

The first condition in the 2nd NF is that the table has to be in 1st NF. The table also should not contain partial dependency. Here partial dependency means the proper subset of candidate key determines a non-prime attribute. To understand in a better way lets look at the below example.

Consider the table



This table has a composite [primary key](https://www.edureka.co/blog/primary-key-in-sql/) **Emplyoee ID**, **Department ID**. The non-key attribute is **Office Location**. In this case, **Office Location** only depends on **Department ID**, which is only part of the primary key. Therefore, this table does not satisfy the second Normal Form.

To bring this table to Second Normal Form, we need to break the table into two parts. Which will give us the below tables:



As you can see we have removed the partial functional dependency that we initially had. Now, in the table, the column **Office Location** is fully dependent on the primary key of that table, which is **Department ID**.

Now that we have learnt 1st and 2nd normal forms lets head to the next part of this Normalization in SQL article.

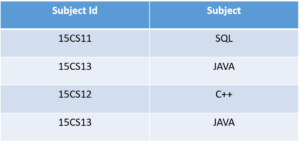
### **3rd Normal Form (3NF)**

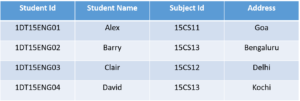
The same rule applies as before i.e, the table has to be in 2NF before proceeding to 3NF. The other condition is there should be no transitive dependency for non-prime attributes. That means non-prime attributes (which doesn’t form a candidate key) should not be dependent on other non-prime attributes in a given table. So a transitive dependency is a functional dependency in which X → Z (X determines Z) indirectly, by virtue of X → Y and Y → Z (where it is not the case that Y → X)

Let’s understand this more clearly with the help of an example:



In the above table, **Student ID** determines **Subject ID**, and **Subject ID** determines **Subject**. Therefore, **Student ID** determines **Subject** via **Subject ID.**This implies that we have a transitive functional dependency, and this structure does not satisfy the third normal form.

Now in order to achieve third normal form, we need to divide the table as shown

As you can see from the above tables all the non-key attributes are now fully functional dependent only on the primary key. In the first table, columns **Student Name, Subject ID** and **Address** are only dependent on **Student ID**. In the second table, **Subject** is only dependent on **Subject ID**.

# **ACID properties of transactions**

In the context of transaction processing, the acronym ACID refers to the four key properties of a transaction: atomicity, consistency, isolation, and durability.

**Atomicity**

All changes to data are performed as if they are a single operation. That is, all the changes are performed, or none of them are.

For example, in an application that transfers funds from one account to another, the atomicity property ensures that, if a debit is made successfully from one account, the corresponding credit is made to the other account.

**Consistency**

Data is in a consistent state when a transaction starts and when it ends.

For example, in an application that transfers funds from one account to another, the consistency property ensures that the total value of funds in both the accounts is the same at the start and end of each transaction.

**Isolation**

The intermediate state of a transaction is invisible to other transactions. As a result, transactions that run concurrently appear to be serialized.

For example, in an application that transfers funds from one account to another, the isolation property ensures that another transaction sees the transferred funds in one account or the other, but not in both, nor in neither.

**Durability**

After a transaction successfully completes, changes to data persist and are not undone, even in the event of a system failure.

For example, in an application that transfers funds from one account to another, the durability property ensures that the changes made to each account will not be reversed.

8-july-2021 10:35AM

Why Database?

SQL is a Language.

MYSQL, ORACLE, POSTGRESQL, SQL SERVER are Providers or propritory software

SQL is saying these are the language fundamentals we need to use for every database.

SQL (interface)

* 1. Oracle
  2. MySql
  3. Db2

SQL is Case Insensitive(SELECT, select, SeLEct)

SQL sub-languages

1. DDL(Data Definition Language)
   * 1. Create
     2. Alter
     3. Drop
     4. Truncate
2. DML(Data Manipulation Language)
   1. Insert
   2. Update
   3. delete
3. DQL(Data Query Language)
   1. Select
4. TCL(Transaction Control Language)
5. Commit
6. Rollback
7. Savepoint (save your transaction if any failure happen we can get back our transaction from where it left out eg:game )
8. DCL(Data Control Language)

You want to give access to user or guest we can use below commands

1. Grant
2. Revoke

**Clauses:**

WHERE - clause directly apply on that expression/query

GROUP BY - You can categorized anything based on city.

HAVING – If you want further filteration to the GROUPBY output

ORDER BY – Order by column1,column2 in either asc(default)/desc.

**Aggregate Functions(across multiple rows):**

Aggregate functions will give single result for a group or category.

Aggregate function can be used in **SELECT or in HAVING only**

1. **Sum()**
2. **Avg()**
3. **Min()**
4. **Max()**
5. **Count()**

**Scalar or Non-aggregate function(PER ROW):**

* + String function
  + Date/time function
  + Math function
  + Miscellaneous functions

9-July-2021

10:30

Joins

12:15PM

PL/SQL tables will be in documents , graphs etc but not in tables like SQL. Its is un structured.

Programming language + SQL

(loops,conditions,methods,switch cases)+SQL

Contains storedproceducres

callcalculateSalary(){

}

Contains 2 blocks

1. Stored procedure(void type)
2. Functions(should must return value)

Create or replace

Procedure calsalary

AS

Name varchar(20);

End calslary;

-------------------------------------------------------------------

Create or replace function

Myfunc

12:23PM

JDBC: Java Database Connectivity/Connection:

* JDBC is a breach driver which provide connection between JAVA and SQL or PL/SQL.
* Takes data from java and translate it into a format which is understandable by sql and delivers it to the sql.
* Takes data from SQL and translate it into a format which is understandable by Java and delivers it to the Java.

Steps to connect to SQL using JDBC:

1. Load or Register Driver(i.e postgress driver) adding dependencies of postgres.
2. Open Connection (url,username,password)
3. Create Statement. There are 3 types of statements
4. Statement

* If there are 1000 statements it compiles 1000 times and 1000 times it runs
* It is prone to SQL injection ex username = ‘drop table login’
* Ex : select username from where username = ‘tom’ and password = ‘jerry’;

1. PreparedStatement extends Statement

* Precompiles once and runs 1000 times and it recompile only and only when there is change in syntax of your query.
* Prepared statement uses query based parameters which are not prone to SQL injection.

1. CallableStatement extends PreparedStatements
   1. Specifically used for PL/SQL for calling stored procedures.
2. Execute Query
3. boolen execute() return type – DDL
4. ResultSet executeQuery() -It is a pointer points to the result of the select query(DQL).
5. executeUpdate() – we get output like 1row inserted/1 row deleted (DML). Output will be in int.

All the above a,b,c can execute any statement like select update alter, delete but if you want are proper response u need to look for appropriate method() and to get appropriate result.

getInt(),getFloat() etc are overloaded methods.

1. Process Results : for storing the results of the query and stored into javaobject like list etc so that we can process it later. Once the resultSet is closed we loss the data so we use list or something to store it and later we can process it.
2. Clone the connection or Release the connection.

Interfaces here : Statement, Prepared Statement, Callable Statement, Result set and Connection

Classes : DriverManager is a class

Step1: load driver

Got to DBeaver> postgress>editconnection> Edit Driver Settings > copy Classname, URL Template and Default port.

Class.forName("org.postgresql.Driver");

Step2: Open connection

String url = "jdbc:postgresql://localhost:5432/postgres";

jdbc:postgresql://{host}[:{port}]/[{database}]

String user = "postgres";

String password = "12345";

connection = DriverManager.getConnection(url, user, password);

System.*out*.println("Connection Established");

A Connection is the session between java application and database. The Connection interface is a factory of Statement, PreparedStatement, and DatabaseMetaData i.e. object of Connection can be used to get the object of Statement and DatabaseMetaData.

Step 3 : Creating Statement

Statement statement = connection.createStatement();

String sql = "select productid, name, rating, cost, available from product\_schema.product order by rating";

Step 4 : execute Query

ResultSet resultSet = statement.executeQuery(sql);

System.out.println("Query Executed Successfully");

Step 5 : Process Result

//result is pointing to result---it will be like an array we use .next() read and increment

//if there is record need to be read it will say true and move ptr to that particular record

//result set will be pointing to before first and checks is there any record need to be read "Yes" then next()

// will also see is there any record need to be read "Yes" then print it out

while(resultSet.next()) {

System.out.print("Product Id "+ resultSet.getInt("productid")+" ");

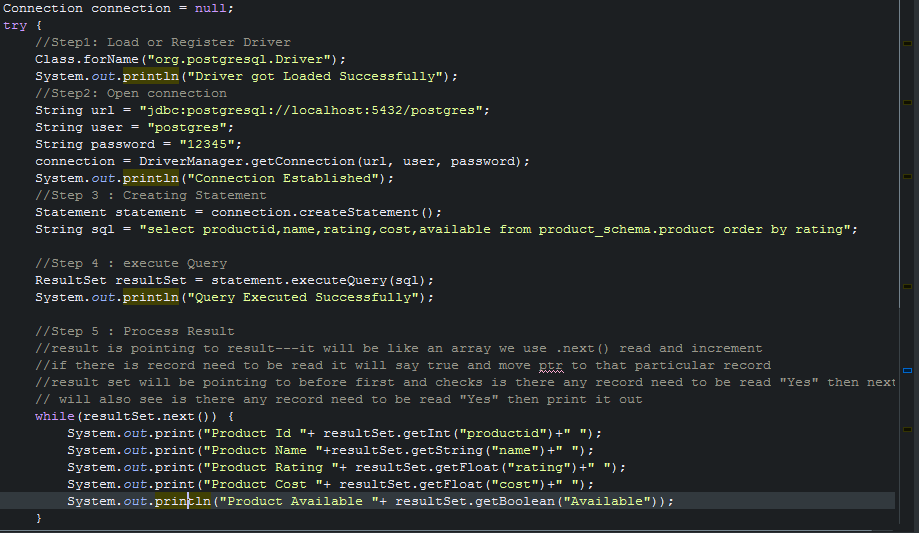
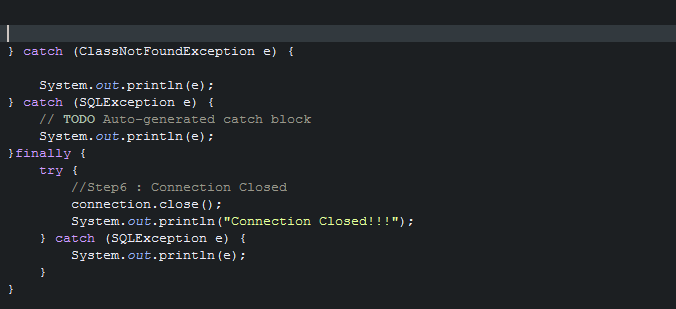
System.out.print("Product Name "+resultSet.getString("name")+" ");

System.out.print("Product Rating "+ resultSet.getFloat("rating")+" ");

System.out.print("Product Cost "+ resultSet.getFloat("cost")+" ");

System.out.println("Product Available “+ resultSet.getBoolean("Available"));}

Step6: Close the connection

Connection.close();

12-July-2021

11:05AM

Presentation layer : mouth of your project in casual. (psvm)

ProductService

Service layer : all the logic calculation everything is valid or not.

ProductDAO : Data Access object

I)Database layer : who will take care of db operatios, handshakes etc.

Product class which are commonly used by all the 3 layers.

First creational pattern- Singleton Design pattern

There will be only once instance of the object(all the common code will be put here. Like we opens connection once not like eveytime opening and closing)

Step1: create singleton

Step2: - make your constructor private so that nobody can access it.

Step3: Have a static method which checks is the instance is always available if yes? It will return that instance else it will create a new instance and revert to it.

Eg: do I have this object already then return it else create a new one and return it and also checking there should be only object remains specially for this class.

Statement.RETURN\_GENERATED\_KEYS is only for autogeneting column only!!!!

Step1: create maven project add the postgres depen

Step2: create product class with name Product.java and package name should be com.app.model in src/main/java

Step3 : how column names are there in the table you need to maintain here same.

Private int product id and so onn.. then generate constructors with fields, getters and setters also string.

Step4 : in the same src/main/java create the interface where all the req will go

Interface ProductDAO and package com.app.dao

All the customer requirements will go here like create a product

Step 4 : now go and create implementation class

Package : com.app.dao.impl

Classs : ProductDAOImpl which implements ProjectDAO interface

Step5: Creating singleton class to put the common code for connection

Package : com.app.dbutil

Class : PostgresConnection

Creating static class Connection

private static Connection connection; // its an sql class and its object

private PostgresConnection(){} //constr why private? Bcoz every1 wil create object of it then what is the use of creating common separate class for common code.

creating-> public static Connection getConnection() throws ClassNotFoundException{ }

result set is for reading, with the resultset only im poiting to that key and which is being genr and im getting that key and setting it to product object.

Step 7 :

Com.app.exception

Class : BusinessException

Extending Exception class //bcoz its checked exception

Source>create consstr with super class and only tick excpetion and string

Make final as string.

Step 8 : package = com.app.main

Class = Main.java

Product product = new Product(“name”,”rating”,”cost”,”available”,”manufacturerid”);

ProductDAO productDao = new ProductDAOImpl();

product=productDAO.createProduct(product);

getProductId()

whenever your are searching always create a Product class and object project also initially keep it as null

Product product = null;

Then say return product;

* Where ever you are expecting data create it there I mean new object
* When insert is happening ,Whenever you are expecting data from a presentation layer that is where u r creating the object.
* When search is happng im expecting data from the persistant layer, im not creating object,
* Ex: Product product = null;
* Means only if I finding an product then only im creating the poduct object
* Product = productDAO.getProductId();

**13-JULY-2021**

11:42Am

HTML

-hypertext markup language

- understood by all the browsers in the world

- no installation or no setup required all that you need a.html file and a browser

**14-july-2021(Week-3 Day-3)**

what is client-server architecture? who is client? who is server? what is web server and application server? Web server vs Application Server. What is http and https protocol? why http is called stateless protocol? different http methods and its uses. various http status codes(100-500).

JSP, SERVLETS, EJB CONTAINERS, APPACHE TOMCAT, HTML CSS JAVASCRIPT.

GET AND POST

PUT PACHE

HEAD

XML

J2SE, J2EE, JSE

jdbc:postgresql://localhost:5432/mydatabase?currentSchema=myschema

What is Client-Server?

11:15Am

Client/Browser/User/App : A client is a user program that runs on a local machine requesting service from the server.

Client Side Scripting Languages : HTML,CSS,JAVASCRIPT and its Libraries like Angular,React etc…

Server : A server is software designed to process requests and deliver responses to another computer over the internet.

A Client and a Server establish a connection according to a set of rules called a protocol. There are quite a few protocols for different purposes, but one of the most popular is the **HTTP protocol**. Once the connection is established, the Client sends **HTTP Requests** to the server in the form of XML or JSON, which both entities (Client and Server) understand. After parsing the request, the Server responds with appropriate data by sending back an **HTTP Response**.

Serverside scripting tool: Servlet, JSP, EJB here we are using Java for writing the response.

Database: Server interact with the database using the providers like JDBC /HIBERNATE/JPA

Other providers are IBATIS, LINQ

### **Types of Client-Server Architecture**

**2 tier architecture** - The user interface stored at the client machine and the database stored on the server. If Business Logic & Data Logic collected at a client-side, then it is known as a fat client thin server architecture. If Business Logic & Data Logic handled on the server, then it is known as a thin client fat server architecture. 2 tier architecture has some limitations in performance, security, and portability.

**3 tier architecture** - Three-tier architecture has a middleware between the user interface and database. The 3 tiers are named the presentation tier, application tier, and data tier. The presentation tier is the front end layer and consists of the user interface. The application tier contains the functional business logic which drives an application’s core capabilities. The data tier consists of a database system and the data access layer.

**n-tier architecture** - In n-tier architecture, there are multiple Business Logic & Data Logic layers. It increases the flexibility and reusability of applications but can be difficult to implement.

Client sends a request to server, server receive the request process it and sends it back to the client.

Here request consist of **request parameters**(in case of login, userid and pw are the request parameters), request type= either he wants to CRUD, search or process anything etc. and also it pases meta info in the sense what type of data im sending, is it a .xml data? or .json data? etc.

Server will process this request based on the reqParameters, requestType and Meta info etc, if required it also interacts with the DB and generate a response. Response also having data or the error msg like 404 error, or another metainfo message provided to the client based on the request by the server

Here we are using HTML CSS for designing our client pages and using JavaScript to make a request to the server and in the server side we will be having Servlet who reads the incoming request and it will process the request and generate the response.

We have HTTP METHODS for making calls for the resources in the server.

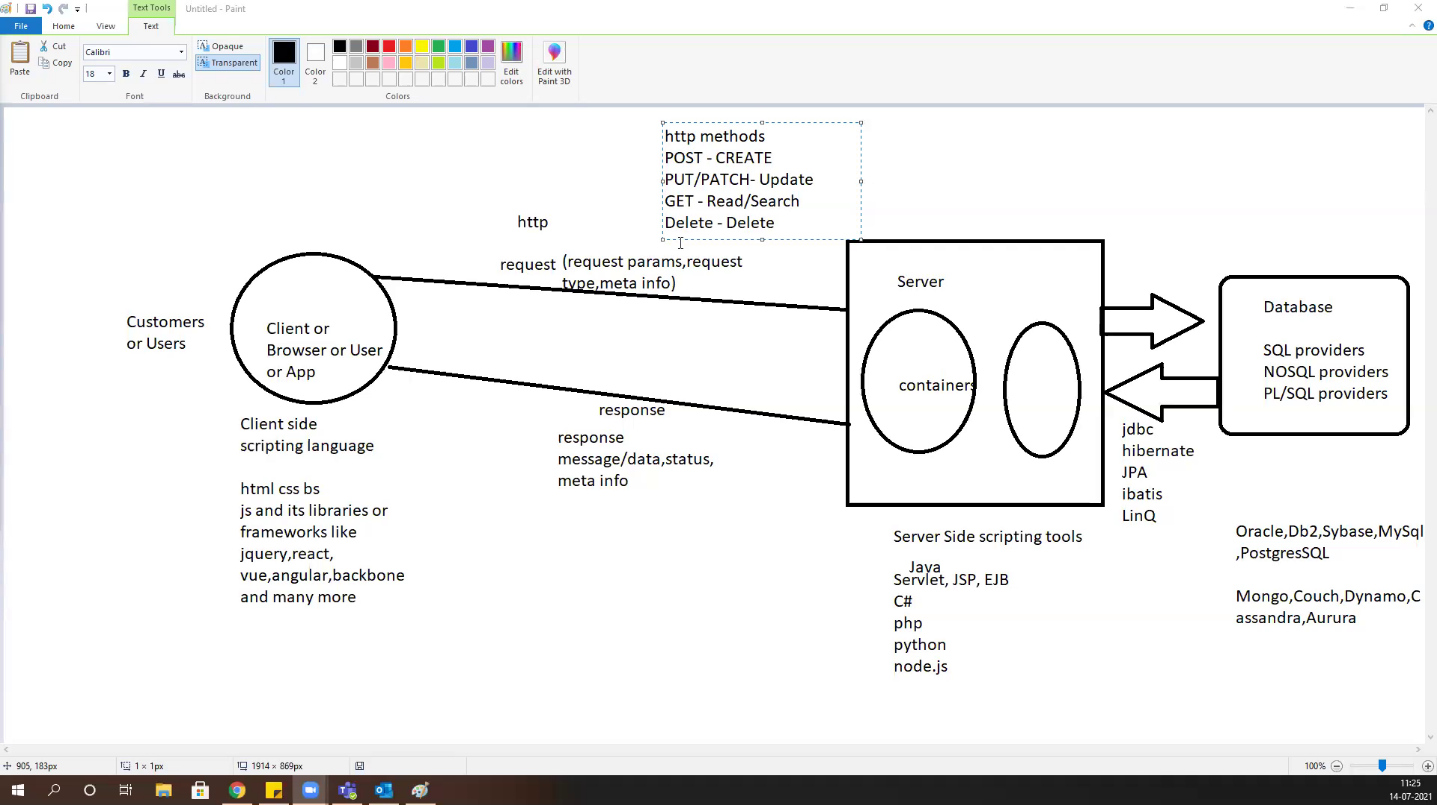
POST – Creating a resource in the backend like creating an employee

PUT/PATCH – Update update emp details

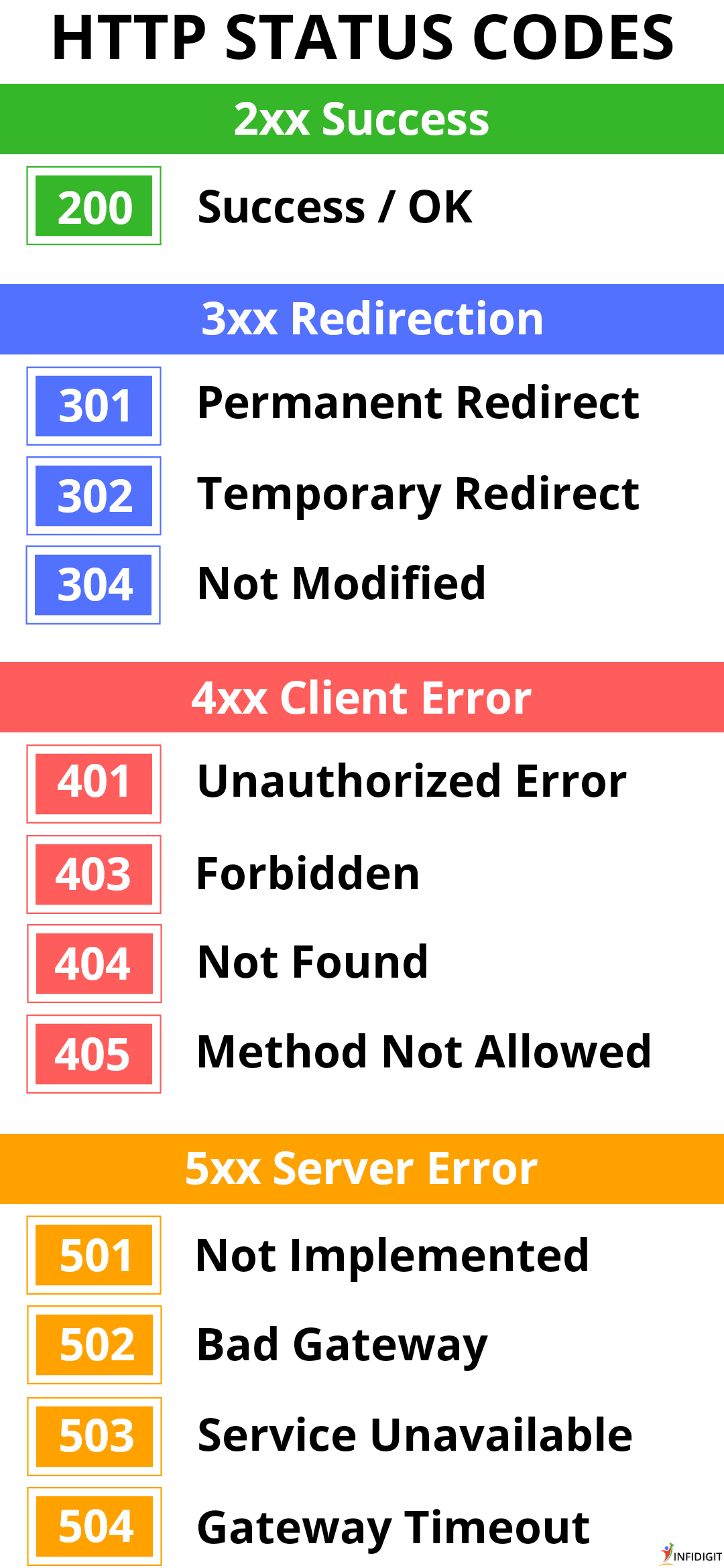
GET – Read/Search

Delete – Delete

Take example of employee here.



**HTTP Status codes : are issued by a server in response to a client's request made to the server.**

* *1xx informational response* – the request was received, continuing process
* *****2xx successful* – the request was successfully received, understood, and accepted
* *3xx redirection* – further action needs to be taken in order to complete the request
* *4xx client error* – the request contains bad syntax or cannot be fulfilled
* *5xx server error* – the server failed to fulfil an apparently valid request

**HTTPS:** It is lightweight,fastest and stateless(don’t remember data where it goes. When it goes to next page it forgets where it there previously) protocol.

## **http vs https**

| http | https |
| --- | --- |
| Transfers data in hypertext (structured text) format | Transfers data in encrypted format |
| Uses port 80 by default | Uses port 443 by default |
| Not secure | Secured using SSL technology |
| Starts with http:// | Starts with https:// |

**Advantages of HTTPS:**

* **Secure Communication:** https makes a secure connection by establishing an encrypted link between the browser and the server or any two systems.
* **Data Integrity:** https provides data integrity by encrypting the data and so, even if hackers manage to trap the data, they cannot read or modify it.
* **Privacy and Security:** https protects the privacy and security of website users by preventing hackers to passively listen to communication between the browser and the server.
* **Faster Performance:** https increases the speed of data transfer compared to http by encrypting and reducing the size of the data.

You might have noticed that web addresses often start with https://. The HTTPS is the secure version of the HyperText Transfer Protocol guarantees that only the client and the server can understand the info that is send from one side to the other.

You can think HTTPS is a language that one computer uses to ask another computer for a document sharing.

Using GET can makes the person Intercept on our convo easily.

Let say you login to facebook.com well first thing you do is you make a POST request, that is a POST to facebook’s login page that has some data attached to it. It has your email and password that goes to the facebook’s server. Facebook’s server figures out that okay, you’re David. It sends a web page back to your browser that says, success! Logged in as David. But along with that web page, it also attaches a little bit of invisible cookie data that your browser sees and knows to save.

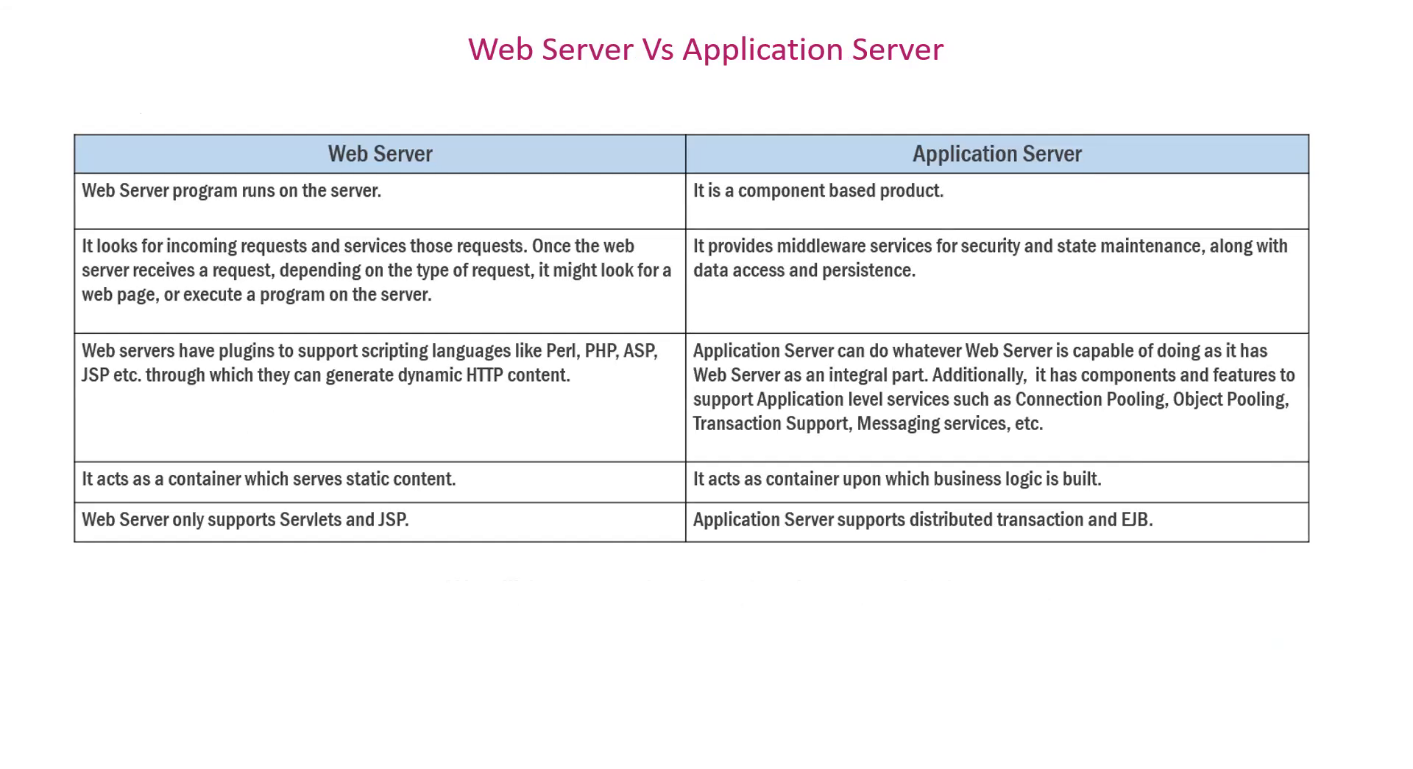
And it’s really important because it’s really the only way that a website can remember who you are. All that cookie data is an ID card for facebook. It’s a number that identifies you as David. And your web browser holds on to that number and the next time you refresh facebook, the next time you go to facebook.com, your web browser knows automatically attach that ID number with that request that it sends over to facebook’s servers, So now facebook servers sees the request coming from your browser, sees the ID number and knows okay this is request from DAVID.

**Java Enterprises Edition(JEE) :**  is designed to help developers to create large-scale, multi tired, scalable, reliable and secure network applications.

Lates Version of JEE is JavaEE7

Static page whose content may not change eg. Style.css, .html, .js .jpg

Dynamic page : whose content changes frequently eg. Userlogin page home page etc.



**CLIENT REQUEST TYPES**

**1.**GET Request :

* Used to GET the resource
* It can’t send unlimited data
* It is not secure, can be bookmarked and , the data is visible in URL
* It is default method for the https. All the urls in the world are GET by default

**2.**POST Request:

* It Is used to process data, save data, change data etc.
* It is secure and cannot be bookmarked
* No limit on data; sent as “payload” and not appended in URL

**3.** PUT is all about updating, it will be update completely.

**4.** Patch is use for complete update.

**5**.Delete request is used to delete a document of the target resource given by URL

**6**.OPTION request return the HTTP method which is supported by the server.

**7.**Head request is used for testing hypertext links for accessibility, validity and recent modification. It generally asks for META info. It is kind a similar to GET but here it asks for response without the response body.

All the html,css, js and other client side files will be in Web Content folder.

New -> dynamic web project-> next-> generate web.xml -> finish

When ever you are making changes always restart your server.

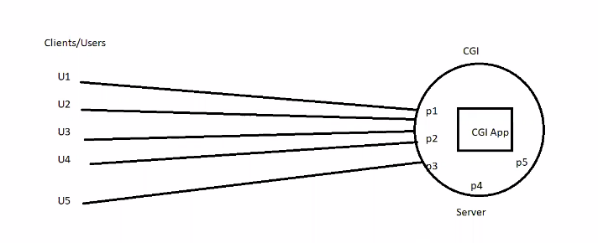
Tomcat is a Web server Hosting my application server, I gave my app to tomcat server what TomCat is doing is, first it goes into **web.xml file**  and understands is your welcome file is index.html and runs it. If tries to run difff file then it will give 404 Error.

Whoever missed generating web.xml file right clik on your project -> JEE and generate deployment descriptor.

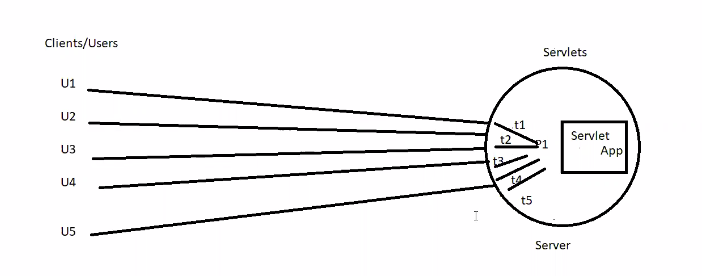
**19-july-2021 2:54PM (Week4- day1)**

**Servlets(JEE)**

* A servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol.
* Capable of handling incoming request and generate appropriate response.
* Written using java, we still write all the code in .java file itself. And handle all the repsonses using https request.
* Servlets handle the request and responses in multithreaded way which makes it lighter than traditional CGI(Common-Gateway-Interface).

What CGI will do is, if there are 5 clients are sends the request it makes separate process for each user which is heavy for the server. If there are n request coming if opens for n times, this may cause memory overflow. And need more hardware support if crowd increases.

To over come above problem we come up with servlet concept which handles the problem in a multithreaded way and also run by itself. What Servlet do is it will create and run a process which then each process consist of number of threads that shares same memory with the process and capable of handling multiple request parallery from the users, by this the, Process will execute only for the one time and internal threads serves the user requests. This approach helps to decrease or minimize the load on the CPU and work efficiently.



**Servlet’s Run on Life Cycle:**

1. void init(ServletConfig)

* This is the first phase of Servlet life cycle where the object of the Servlet Class is created when the request comes in and its been put in actively in the server to handle all the requests coming in.
* ServletConfig is a interface where we pass any custom configuration related to servlet.
* This LifeCycle stage will be executed only and only once.

1. Void service(ServletRequest, ServletResponse)

* This is the 2nd stage of lifecycle of servlet where we write the code of what exactly this servlet should do, you write the logic of handling incoming request and you will generate appropriate responses here, this lifecycle will be executed for number of requests once.
* **ServletRequest** is where the request objects or data come to the application which is input to servlet/service layer of servlet.
* **ServletResponse** is where when if servlet or service wants communicate back to the client.

1. void destroy()

* When there is no active request/thread on the particular Servlet, the servlet will be taken off from the main memory, that’s when the destroy() is invoked.
* This stage of lifecycle will be executed only and only once.

Tomcat(is a web server) has two containers:

1. Servlet Container (all life cycle will be executed)
2. JSP container(html,css)

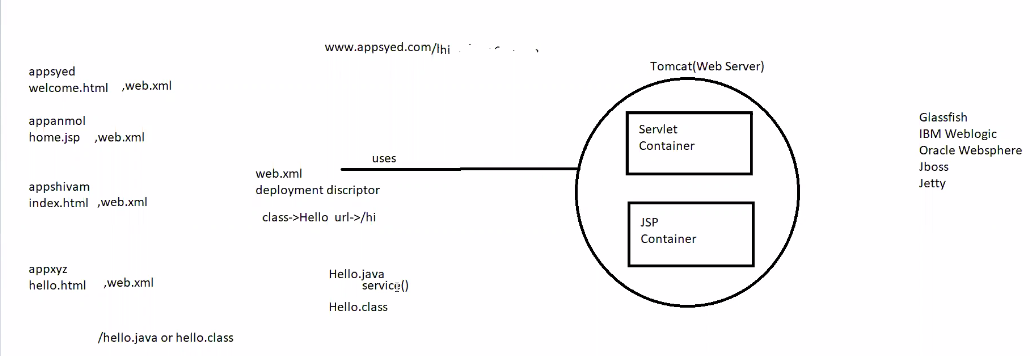
What ever libraies we required to run app its there in the tomcat (servlet-api.jar), we don’t need JDK anymore.

Other than Tomcat we have Glassfish, IBM WebLogic, Oracle WebSphere, JBOSS, Jetty etc provides Containers.

* Utilize web.xml and understand deployment and also called as deployment descripter.
* One tomcat can run multiple application in it.
* Web.xml is a source of communication between server and the application html file.

**Web.xml file** is popularly known as configuration file or deployed descriptor for any JEE apps, All the deployment and configuration details which will be used by the web server will be given here and the web server utilized this file for all configuration details purpose.

Containers means where life cycle of Servlet is executed.



Javax.servlet.Servlet(Interface) : is a root interface for all the servlets. This is where it consist of life cycle methods like init() service() and destroy().

1.Javax.servlet.GenericServlet(AC) : which is an Abstract class and it is generic to any protocol inclusive to http.

In GenericServlet we will get void service(ServletRequest, ServletReponse) method

All init, service and destroy will be used by above hierarchy.

2.Javax.servlet.HttpServlet(AC) : It is a child of GenericServlet() and it gets all the properties of GenericServlet that means it also have void Service() method, Additionally HttpsServlet() provides specific methods to handle the specific request of Http methods like GET,PUT,POST,DELETE etc.

For Handling specific request we have specific HTTP methods which is going to handle the incoming request.

* For Reading - doGet(HttpServletRequest,HttpServletResponse)
* For Creation – void doPost(HttpServletRequest,HttpServletResponse)
* For Updation -doPut(HttpServletRequest,HttpServletResponse)/doPatch(HttpServletRequest,HttpServletResponse)
* For Deletion - doDelete(HttpServletRequest,HttpServletResponse) to delete the resource

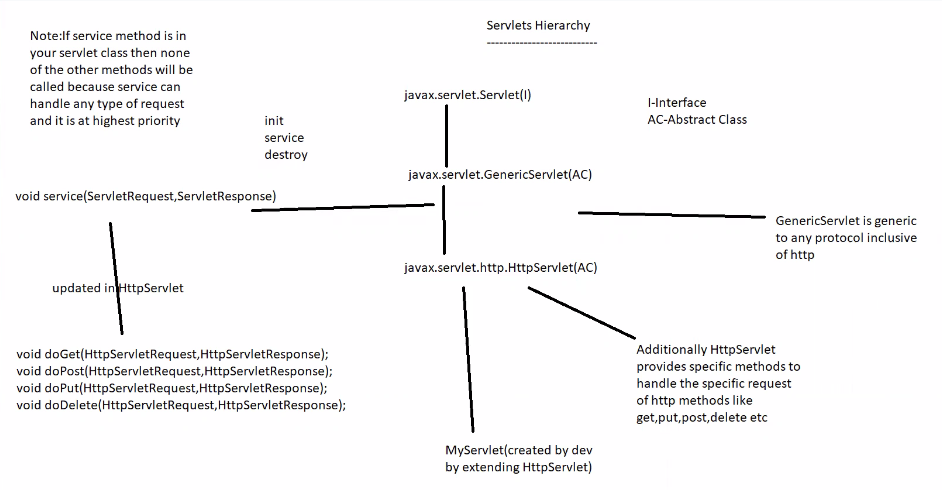
As **void service(ServletRequest, ServletReponse)** is generic one, We don’t require all CRUD methods, just in case if **void service()** not there we require CRUD METHODS. If void **service()** method is in your servlet class then none of the CRUD will be called because **void service()** is of highest priority and also it is generic one. So, It can handle any type of request.

**There are only two types of Servlet :**

1. javax.servlet.GenericServlet(AC)
2. javax.servlet.http.HttpServlet(AC)

All two types of servlets are abstract classes, GenericServlet implements servlet.Servlet(I) interface where all the life cycle methods are available.

* Note that if the service method is there then none of the other methods are eligible to take the call, all requests(get,put,post or delete) will be routed to service only, and if service is not there that is when it will be routed to specific https methods for specific calls.
* NOTE : All URL/s in the world by default are of GET Request.
* From html form we can only make GET AND POST request.
* Using JavaScript on the front end we can make any http method call.



**Hello\_servlet\_app\_demo program**

1)Setting the reponse object from service()

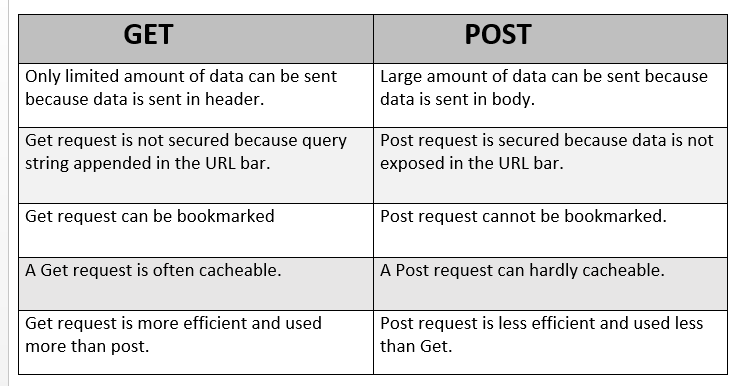
response.setContentType(“text/html”) : my reponses will be plain text or html

2)PrintWriter out = response.getWriter(); //it initilaze the **response object** which send response from the servlet to the browser or client

out.print(“Hello From Service Method of MyServlet()”)

* web.xml can be edited as per requirement
* we can give any number of names for the URL for a particular servlet
* if you are changing anything in the .java file or web.xml please restart tomcat server.
* HTML is only capable of doGet() and doPost(), JavaScript is capable of all the http methods.
* doGet() is universal for the html pages. Like if we don’t mention any method=’’ by default it will be doGet().
* Whenever you are making changes in static pages like html we don’t have to restart Server.
* If we want anything from the client we use **request**  and if we want to tell anything to the client we use **response**
* We use request.getParameter(“fname”) for taking input from the client.
* Where ever user is selecting any input we should pass **value=’’** to input tag
* Where ever user is typing any input we should use **name=’’** attribute there
* Short hand to print an Array without for-each loop by using Arrays.toString(request.getParameterValues(“hobby”));

**20-july- 12pm**



ServletConfig and ServletContext :

1)**ServletConfig :**

* It is a configuration based parameters which can be passed to specific servlet only.
* The Configuration params set using ServletConfig for a particular servlet which cannot be accessed by other Servlet.
* For eg: If Servlet1 has some config params which is been set using ServletConfig these params cannot be used in Servlet2, it is local to that Servlet only.
* To set config params in the web.xml file within servlet tag we can use <init-param></init-param> tag and give the param name and param value as a key value pair and you can access those params inside that particular Servlet Class.

ServletConfig config = getServletConfig();

Config.getInitParameter(“query1”) //here query1 is a key and will get the value

Config.getInitParameter(“query2”)

**2)ServletContext:**

* ServletContext is global to any servlets within this app, the context params set can be accessed by any Servlet throughout the entire application.
* We can set context params within web.xml file by using <context-param> tag within the <web-app> tag anywhere.

ServletContext context = getServletContext();

Context.getInitParameter(“driver”) //here driver is a key and it will give the value.

**servletConfig\_and\_ServletContext\_demo program**

**RequestDispatcher API :** The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.

It provides below 2 methods

1)forward(request,response) - forward will forward the current request and response to the specific url mentioned within the application itself and will send the response of the forwarded page only and nothing else apart from it.

2)include(request,response) - include will include the response of current page and also the response of the page given in the url of include both together as one response will be sent to the browser.

In both the cases the URL wont change it will be the URL of that particular servlet where this code is written.

**Myservlet\_login\_page**

**response.sendRedirect("url") vs forward**

->sendRedirect changes the URL and creates new request, forward keeps existing request alive.

->forward only within the application, sendRedirect we can even navigate outside the application.

3:00PM

1) Hidden Form Fields : here we pass hidden fields from pages to pages until it reaches data base, would be fine if there are 2 or 4 fileds but would be headache if there are n fields and we have to write code for that n fields and pass it on..

2. URL rewriting : here what happens is we are using GET request, you already know if we accept form data it will be visible in URL and it got appended to the next .html page which is not good if we are passing sensitive info.

3.Cookies : smallest part of info stored in client machine in a key value format both will be strings.

Adv : Light weight, everything maintains at client.

Disad : if client machine is public computer it is not safe

We can’t store objcts in cookies bcoz it is only n only stirng.

Ex: if there are 4 pages all 4 pages data will be stored in brower in terms of cookies at the end brwser can fetch cookie data from the browser itself.

4.HttpSession : is like an storage or an activity or you can say like between the user login and logout. What all the things he is doing between login and logout that is called **one session**. Every user will have their own indivisual session let say if 100 users 100 sessions will be there. Why ? between the session we can share the data.

Ex: I can store page1 data into session object, since session resides in server side and data is stored into java object. As I got to page 2,3 I can go on updating that java object collectively, When we reach last page I will be having all the pages data into session object, finally that I can pass it on to the Database.

Http session object : We are using to solve http problem. Which can be used as storage, it can store session time etc and we can also set session timeout and session expire.

Session is created when user logins and destroyed when user logout.

For every session getting created, the servlet created a unique id for per session. By this unique Id servlet can track all the information stored into the particular session. You can add, modify or remove the details from session and also you can destroy the whole session.

How did I solved that https stateless problem thru session?

I can solve it by storing all the field details of the all the html pages into a session and retrieving it from the session within the servlet and taking it to the database easily.

Disadvantage: It can be heavier which is not good for the server. Just keep it as light as possible and take it out which is not required.

**Servlets provide 3 types of Scopes**

1)ServletRequest - request scope – this is active until the current URL is not changed.

If the URL is changed request scope creates new scope which is again a new request.

If you want to work with request scope you should effectively used request dispatcher like forward include. That is when requestscope will be aligned if used request redirect request scope will die.

2)ServletContext – application scope – this is active until the entire application’s **destroy()** is invoked.

Once destroy() is invoked it will also destroyed.

3)HttpSession – session scope – this is per session/per user/its b/w user login to logout.

* What ever will be stored in session1 it won’t be accessible to other sessions. It will be local to that particular session only.Session will be unique to for every user.
* HttpSession session = request.getSession(); //return the current session associated with this request oir or if the this request does not have session it will create a new session.
* getSession(boolean) is a overloaded method, uses getSession(false) when you don’t want session to be get created but just participate into existing session.
* Uses getSession() when you want session to get created by default it will be false.
* So in next servlet we keep it false because it is for one session.
* It is also secure; secure in the sense when user directly going to access a particular page with the URL he cannot be able to do so, as said earlier one session is local to that session only other person cannot intercept what’s going on between session.
* session.setAttribute(“key”,”value”);
* sessioninactiveinterval
* listeners

5:05pm

Login app with session

21-july 10:30am

http/https methods

---------------------------

->GET(default) - Used for reading resource from server. Passes the data via the URL i.e it appends the data at the end of the url.

->POST - Used for creating a resource, it is secured comparable to get because it passes the data via the payload not the URL.

->PUT - used for updating the resource completely and it is idempotent in nature, passes the data via the payload.

Idempotent : I have a browser and a registration page , let say same user submitting the same data 100 times with same values. Here if I go with POST request, It will create 100 new resources where PUT will only start reacting when there is change in the values. If no change PUT won’t react. This is we called idempotent in nature.

->PATCH - used for partial update, sends data via the payload.

->DELETE - used for deleting the resource from backend, sends data via the URL similar to GET.

Complete Updation means object is getting modified completely

Partial Updation : Where as Partial Updation means only some fields in the object are getting modified.

Payload : browser storage we can say. That data can be send thru URL to the Server.

**hello\_postman\_demo : Demo to check out the http methods**

Postman is an API testing tool whatever Restful calls(doGet,doPost,doPut and doDelete) request we are making for all this things We don’t have to write Javascript to make a call. Before Writing JS for all of this calls we can Test those calls utilizing the tool just like POSTMAN. Apart from POSTMAN we have SOAP UI.

Which is used for API testing , this is part of testing only but technically we called as API Testing which means you are testing certain point in your application.

Eg : like how the Get request is doing, what is the response of a Post request. We as Dev already aware what is expected request and expected response during testing thru this tool we can test in advance before writing JavaScript for all of this calls.

As you know HTML is limited to get and post only so it not a good idea to write JS for other calls like put and delete, so its better to use POSTMAN for testing those calls.

Before UI construction we are just making sure that everything is going absolutely perfect by testing the calls using POSTMAN.

JavaScript will be sending data either in .xml or .json format. As server won’t understand .json so we need a communicator like Jackson library

It will take JSON object and convert into java object POJO class we say

Eg : var employee = {“id” : 100, “name” : “suleman”, “designation” : “SDE”} ==> class Employee{Int id; String name; String designation;}

Jackson library will also do java object -> JSON object.

Our front-end which is written using html and javascript is different and backend which is written in java is different thing both has to communicate commonly b/w each other. So, what’s the common thing we can use? We can use .xml because we have XML Parsers over here on both the sides here java script will be having JS xml builders which build xml and send it to the backend, here java also have xml parsers which I can use to map it the java objects and when Im sending back this info as a response by converting into XML so that frontend can read that. This XML we are using for a common entity to do the handshake b/w two different application which is commonly known as web services.

**Web Services**: One application taking the services from another application with three things over an internet, over an network, over a protocol is what is called as Web Services. In this one application is acting as a service provider and another application is acting as a consumer and they are using http protocols primarily with all https methods to signal b/w each other so that what client is asking server will be able to understand that and process the reponses based on that. And they also need a common way so that they can communicate with each other. To communicate between them.

For example : we already have .xml in which is globally understandable or we can consider JS0N which is understood by al JS F/w so we don’t have to parsing or reading. We can directly convert HTML data whatever it is coming into json and makes those http calls and in the backend we are utilizing Jackson library to convert the JSON object into java object for the incoming request. We process and sends back the reponse by converting java object into back to the JSON object. That’s what makes the typical client server architecture with web services this type of web services where are effectively using http request with the redable XML readable JSON so such web services are called as RestFul WebServices here REST stands for : Representational state and Transfer based on http/https and communicate using redable XML or JSON. It is light weight and faster.

For this **Jackson library** , We are using **GSON** (Google’s Script object notion) which will be helpful in doing conversions like JSON to Java object and Vice Versa.

**Student\_restful\_api\_demo**

22-july 11:51AM

DOM Events:

onclick

ondoubleclick

onsubmit

onload //what to do after page load

onfocus // after hovering

onblur

onmousehover

onmouseout

JSvalidation

Callbacks and promises in JavaScript

**22/7/2021 -> 2:20pm ajaxdemo.html**

Ajax – Asynchronous JS and XML is a callback style.

Has capability to get data from backend without refreshing the page.

This data reicve by JS once JS is recived and can put data on the html page for you.

XMLHTTPRequest() we can make http calls using XMLHTTP to make a call to backend api to make a request.

this keyword is local to its function in arrow function.

**fetchapidemo.html**

FetchApi is smiliar to AJAX but it is more of promise Style of writing the code which avoids all the boilerplate of code.

**26-07-2021**

**Before AWS:**

**Suppose you want to host a website, these are the following things that you would need to do:**

**1) Buying a stack of servers**

**2) if high traffic maintaince? more servers**

**3) Monitoring and Maintain servers**

**if you consider costs then this setup is expensive**

**troubleshooting problems can be tedious and may conflict with your business goals.**

**since the traffic is varying, your servers will be idle most of the time.**

**What is cloud? Cloud is a huge space which has many services available you can just go and rent those services as we want on a rent basis**

**Vmware is an desktop software runs on micrsoft windows, Linux and other OS.**

**-> It has Type-1 Hypervisor that runs directly on server hardware without requiring an additional underlying OS.**

**Virtualization: It means to create a virtual version of a device or resources,**

**such as a server, storage device, network or even on OS where the software divides**

**the resource into one or more execution environments.**

**-> Virtualization means ability to create a virtual version of a particular device wheather it is an**

**storage or server or a network.**

**Crtiteria to consider :**

**Operating System**

**Application Server**

**Application**

**Administrative**

**Network**

**Hardware and Storage**

**What makes virtulization feasible is Hypervisor.**

**Hypervisor is a piece of software that runs above the host computer, what hypervisor**

**basically do is it pulls the resources from the Physical Server and allocates them**

**to your virtual enivronment.**

**There are Two Main Hypervisor:**

**1) Type-1 Hypervisor: It directly installed on top of the physical server.**

**2) Type-2 Hypervisor: In type-2 hypervisor there there is an OS exist between the**

**physical server and the HV. ex: oracle virtualbox, vmware**

**On Hypervisor you can run mutiple virtual machines. It has there own OS, SERVER, NETWORK, and Storage.**

**Benifits**

**1) Cost Saving : You can drastically reduce physical infra, don't have to maintain separate servers, run an electricity and other micellneous stuff.**

**2) agility and Speed : spinning up a virtual machine is relatively easier than setting up a new Physical Environment.**

**3) Lowers your Downtime : suppose if a host goes down, you can move VM'S from one HYPERVISOR to another on a different physical server**

**this provides you a great backup physcially.**

**Virtulization** : Virtualization uses software to create an abstraction layer over computer hardware that allows the hardware elements of a single computer—processors, memory, storage and more—to be divided into multiple virtual computers, commonly called virtual machines (VMs).

Hypervisor : A hypervisor, also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing.

Hypervisor software : vmwarre,citric hyperv

Type 1 hypervisor does not have OS

Type 2 hypervisor

Int type -2 hypervisor : h/w->windows->vmware ->3 virtual machines

What is AWS?

Compute, storage, and network services need to be remembered.

Each availablility zone(A-Z) is one data centre.

**2pm**

What is cloud computing?

**Cloud is a huge space which has many services available you can just go and rent those services as we want on a rent basis**

Models of cloud computing

1. **SAAS : Application stack, OS, Network, Storage, Compute and you’re your application as well. Ex bankapp or outlook**

Cloud provider rent applications or softwares which are owned by them to its client. For example : **Salesforce.com** provides the CRM on a cloud infrastructure to its client and charges them for it, but the software is owned by **the Salesforce company only**. Here Customer only utilizes the software on a rent basis.

Here customer don’t have to worry about anything like upscaling down scaling/security issues and other maintenance, he just utilize it.

More over example : We order a pizza online it comes to your place and you consume it.

1. **PAAS : We provide network, storage, compute, ApplicationStack(TOMCAT, .NET and JBOSS) it is with OS.**

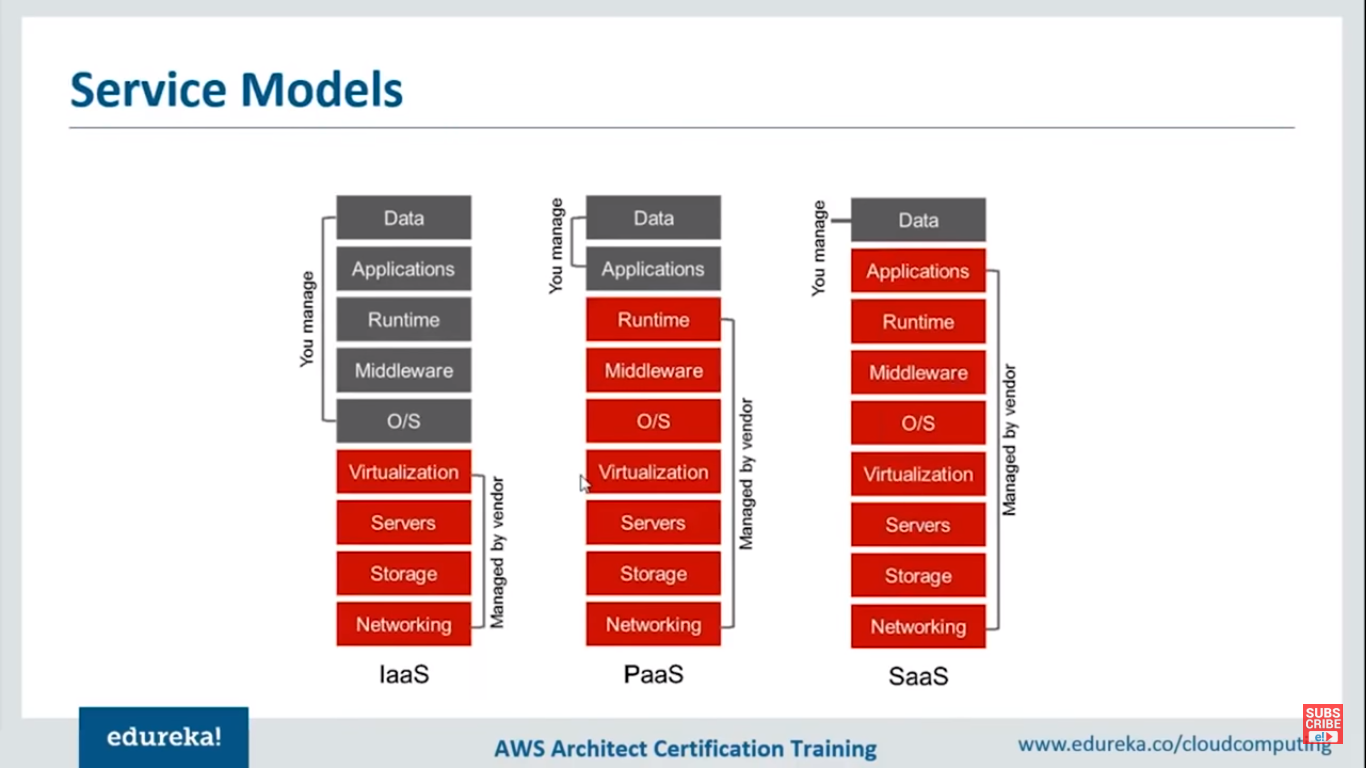
Here we are provided with an platform to build our own application.

More over example : Going to hotel and eating an pizza

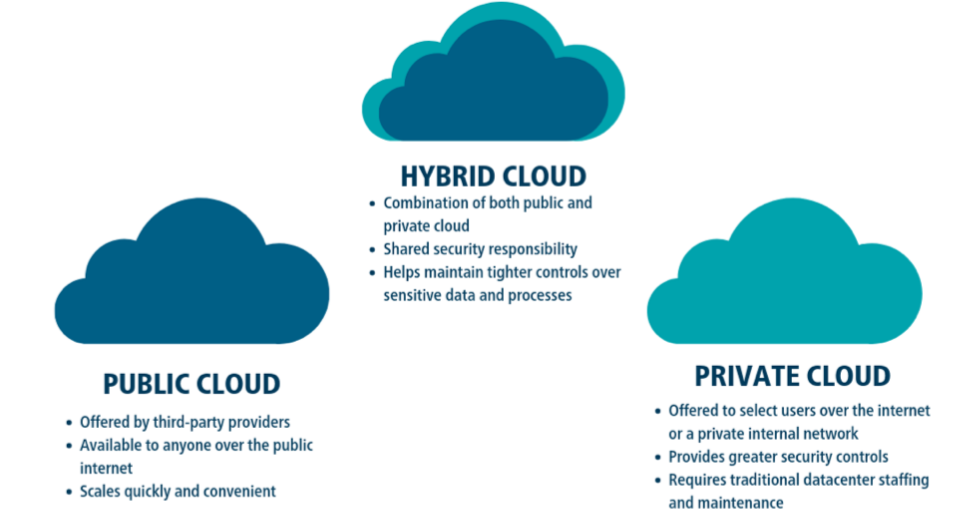
1. **IAAS : If we provide network,storage,and compute(processor) it is without OS**

Here you will be provided with an whole infrastructure you have choice to select hardware config OS and technologies you want to use etc.

More over example : You have the infrastructure like kitchen you have everything which requires to bake a pizza, only thing is you should know how to utilize tools to make it.



**Deployment models:**

* + 1. Public Cloud
    2. Private Cloud
    3. ****Hybrid Cloud

**Cloud Providers:**

1. **AWS**
2. **Microsoft Azure**
3. **Google Clout Platform**
4. **Digital Ocean**
5. **IBM Cloud**

**What is AWS?**

* It is one of the most popular cloud providers which provides services to the user on demand basis.
* It is one of the most secure cloud platform, offering compute power, database storage, and other functionalities to help business scale and grow.

**Features:**

1. **Flexibility**: It is highly Flexible, flexible in the sense AWS is Capable at managing the data, deployment and taken care of the data. Suppose if you are using any service only for one hour you would be paying only for one hour that is how flexible it is.
2. **Cost Effective**: AWS take cares of almost every aspects, Suppose if you are a beginner or a learner they have something called as **Free-tier,** that means you have sufficient resources to use for one long year. It also have PAY-AS-YOU-GO model, it charges only for the services you are using. Let’s you scale up and scale down nicely. That is why it is very much scalable and flexible.
3. **Scalability**: Since it is easy to scale up you can start with less when the traffic is low and you can easily scale up when traffic is high vice versa.
4. **Security**: AWS has great security mechanism it uses various encryption algorithms like AES. Here also the AWS is taken care of all the security aspects so that USER can focus on the business rather.

It also have automatic scheduling and various integrated Apis which help to switch between different programming languages.

**AWS ARCHITECUTRE:**

**AWS Regions & Availability Zones**

AWS now spans 77 Availability Zones within 24 geographic regions around the world, and has announced plans for nine more Availability Zones and three more AWS Regions in Indonesia, Japan, and Spain.

**AWS Region:**

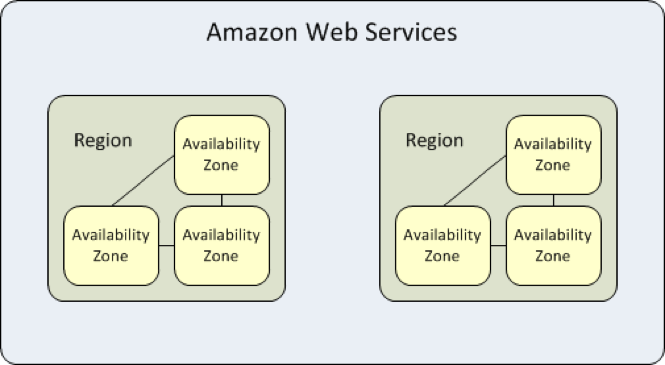
An **AWS Region** is a geographical location with a collection of availability zones mapped to physical data centers in that region. Every region is physically isolated from and independent of every other region in terms of location, power, water supply, etc.

This level of isolation is critical for workloads with compliance and data sovereignty requirements where guarantees must be made that user data does not leave a particular geographic region. The presence of AWS regions worldwide is also important for workloads that are **latency-sensitive** and need to be located near users in a particular geographic area.

Inside each region, you will find two or more **availability zones** with each zone hosted in separate data centers from another zone.

**Availability Zones**

An availability zone is a logical data center in a region available for use by any AWS customer. Each zone in a region has redundant and separate power, networking and connectivity to reduce the likelihood of two zones failing simultaneously.



Oregeon is an region which has 4 Availaibility zones

Mumbai is an region which has 3 availiability zones.

**AWS DOMAINS:**

**1)Compute(Elastic Compute** Cloud**):**

What it does is, it lets you have resizable compute capacity, It’s more of a raw server where you can host a website and it’s a clean slate.

Clean slate in the sense let say you go ahead and buy a laptop, it is an clean device where you can choose what OS you want and all those other things. EC2 is similar kind of thing which lets you host website in a clean state.

2)**Elastic BeanStalk** which lets you deploy your various applications on AWS, you do not have to worry about the underlying architecture now, it is very similar to EC2 the only difference between EC2 and EBS is, EBS have predefined libraries where EC2 is a Clean State.

When I say Predefined Libraries, all the technology like java, SQL and other things are already been there you just need to pick up those need and you are good to go.

**What is an instance?**

An Instance is a virtual server for running applications on Amazon’s EC2. It can also be understood like a tiny part of a larger computer, a tiny part which has its own Hard Drive, network connection, OS etc. But it is actually all virtual in nature.

**Types of instances:**

1)General Purpose Instance : For applications that requires a balance of performance and cost. Where you need a prompt response, cost effectiveness, less processing. For Ex: Email Response Systems

2)Compute Instance : For applcication that requires a lot of processing from the CPU. For Example : Analysing Streaming Data such as zoom live streaming.

3)Memory Instances : For application that are heavy in nature, therefore, require a lot of RAM, need for the application that requires multi tasking, multi processing. For example the application which takes the multiple inputs generate multiple outputs and serve multiple users at a time parallely.

4)Storage Instances : For applications that are huge in size or have a data set that occupies a lot of space. For Example: Big Data Applications which requires lots of Data storage and data processing.

**What is an EC2?**

Amazon Elastic Compute Cloud, EC2 is a web service from Amazon that provides resizable compute services in the cloud.

EC2 Autoscaling

## **EC2 Autoscaling**

Amazon EC2 Auto Scaling helps you maintain application availability and allows you to automatically add or remove EC2 instances according to conditions you define.

### **Benefits**

1. Improve Fault Tolerance
2. Increase Application Availability
3. Lower Costs

##### **Improve Fault Tolerance**

Amazon EC2 Auto Scaling can detect when an instance is unhealthy, terminate it, and replace it with a new one.

##### **Increase Application Availability**

Amazon EC2 Auto Scaling ensures that your application always has the right amount of compute, and also proactively provisions capacity with Predictive Scaling.

##### **Lower Costs**

Amazon EC2 Auto Scaling adds instances only when needed, and can scale across purchase options to optimize performance and cost.

### **Auto-Scaling Groups**

**Groups** are collections of EC2 instances with similar charcteristics. Using the auto scaling groups you can increase the number of instances to improve your application performance and also you can decrease the number of instances depending on the load to reduce your cost. The auto-scaling group also maintains a fixed number of instances even if an instance becomes unhealthy.

### **Launch Configuration**

The launch configuration is a template used by auto scaling group to launch EC2 instances. You can specify the Amazon Machine Image (AMI), instances type, key pair, and security groups etc.. while creating the launch configuration.

### **Scaling Plans**

Scaling plans tells Auto Scaling when and how to scale. Amazon EC2 auto-scaling provides several ways for you to scale the auto scaling group.

* Schedules Scaling
* Dynamic Scaling
* Predictive Scaling

### **How to Configure Autoscaling**

In order to configure autoscaling, you must create an AMI and launch template. You can find [full autoscaling documentation here](https://docs.aws.amazon.com/autoscaling/ec2/userguide/GettingStartedTutorial.html#gs-tutorial-next-steps). The basic steps are as follows:

1. Create a launch template
2. Create an Auto Scaling group
3. Verify your Auto Scaling group
4. Customize Auto Scaling plan.
5. (Optional) Delete your scaling infrastructure

**EC2 (Instance): Elastic Compute Cloud where you can scale up or scale down.**

Instances = VM =

VPC = Amazon Virtual Private Cloud

Step :1 AMI = Amazon Machine Image same like .iso

From the image you create any virtual machine that is called instance.

Image consist of OS, S/W, APPLICATION

Step2: Choose an Instance Type where we select system config like ram storage, vcpus, memory, instance storage etc.

Step3: configure Instance Details : Will provide default VPC, Availability zones like Oregon, Mumbai etc. and make subnet default

Step4 : Add Storage: We are calling hard disk as Elastic Block Storage(HDD)

IOPS = Input output process (no. of users can connect)

Don’t increase the size here SIZE(GIB)

Step 5 : Add Tags

It is for indentification where we provide key value pair

Key : name Value : Windows-server-2019

Step6 : Configure Security Group default subnet group means firewall

Step-7 Review Instance pair

Every instance vm we create it is map to one key pair(one is public key and private key) it is with .pem file PEM : public encryption mechanism

This .pem file consist of public and private key and that key pair is available in .pem file.

For remote connection between windows to windows we require RDP(Remote Desktop Program) mstsc.exe

For remote connection between windows to Linux we require SSH Client(putty.exe) open cmd -> check ssh ins installed or not

Firewall has rule if obey it allows else not

**Restrict by IP**

Custom : only specific member can access

Anywhere : anyone can access server

MYIP : only I can access my server

Alltraffic means : any one can access with anyport or protocol can access

**Restrict by protocol**

If we specify SSH OR TCP anything only that protocol members can access.

**Restrict by port numbers**

Any one access with port number 3389 etc…

EBC : If you create a voume on 2a it can be usable by only inside that 2a servers only not to other available zones.

LINUX Commands:

# install httpd (Linux 2 version)

==================================

yum update -y

yum install -y httpd.x86\_64

systemctl start httpd.service

systemctl enable httpd.service

echo "Hello World from $(hostname -f)" > /var/www/html/index.html

yum is a package manager, what ever we want we can ask yum it will provide the same.

Ex: yum install -y httpd.x86\_64

$ sudo is a root user

$ sudo yum update

$ sudo yum update -y here y is yes for all

$ sudo su means we are making us admin of the system so we need to have type sudo for every command.

12pm : RDS(Relational Database System)

Go to services -> Database -> RDS

1pm database and AWS Data base HANDSHAKE

2:40 Deployment .jar .war

5:05 docker and containers

Deployment on AWS

28/7/2021 11pm

docker hub : public docker image

spring web

change port

server.port = 9000

conternezation : technqiue rather run on vm we can run on docker

docker archi

docker engine : docker

client can be cmd or shell

docker\_host : docker daemon : it listen the commands internally it convert into restapi calls and see the command and execute it

docker have images it takes the image and create docker instance.

here image is nothing the softwares we want.

there are basically two images 1)local registory or remoterepo

2) remote registory or remote repo

from the image we can make containers

if image is not available the docker demon will go in public or remote registory and get it from there

Docker cmd:

docker --version

docker info

Image commands :

docker images : what images availbe in docker local registry.

docker rmi id\_of\_the images or name : To remove an image.

docker images : to list all the images

docker pull image\_name: to pull image from public repo ex : docker pull ubuntu

Container commands:

docker ps container\_name: if want to see what containers are running. ex: pull ubuntu

docker ps -all : to see all container which are running or stopped.

docker container rm container\_name -f : to remove a container but here images will not remove

How to load an image into an contianer?

First create a container:

docker run --name name of the container image id or name

ex : $docker run --name myubuntu -d ubuntu

$docker run -itd myubuntu bash

here -it means interactive mode thru bash im going to interact with ubuntu

d means keep it running.

$docker exec -it docker\_cont\_id bash : for running and exeuting.

\*\*\*\*\*2:30pm

how to convert .jar to docker file

frist generate .jar by maven

second create docker file by cliking on new file -> Dockerfile

copy this in Dockerfile DockerfileFROM openjdk:11COPY ./target/icicibankapp.jar icicibankapp.jarCMD ["java","-jar","icicibankapp.jar"]

open commd with loc where that Dockerfile is located

then type this command

$docker build-t icicibankapp . -f Dockerfile

-t : image tag name

. -f file name

Will get conternzation after running above command.

\*remember the port number

next step running that image

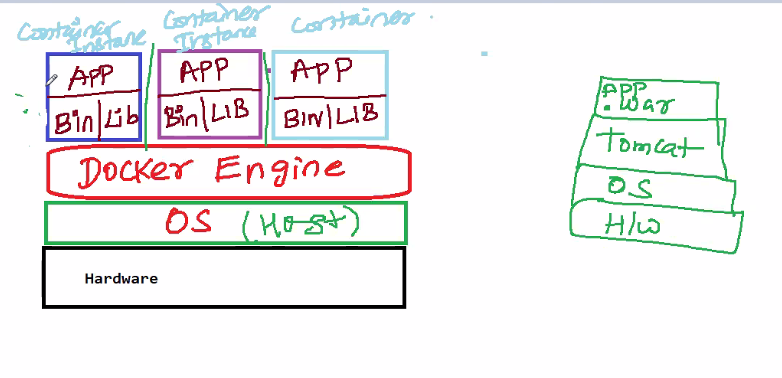
$docker run --name iciciapp -p 9000:9000 icicibank.app

28-07-2021 4:30pm

showed java n sql project into docker

$docker inpsect id\_of\_container //it will give information realted to container will be dispplayed.

-------------------------------------------------------------------

Elastic Beampack for deplyoing webapps

11AM 02-08-2021

JPA : Java Persistence API

What is an API?

It just means an interface, or method/way, for two pieces of software to communicate.

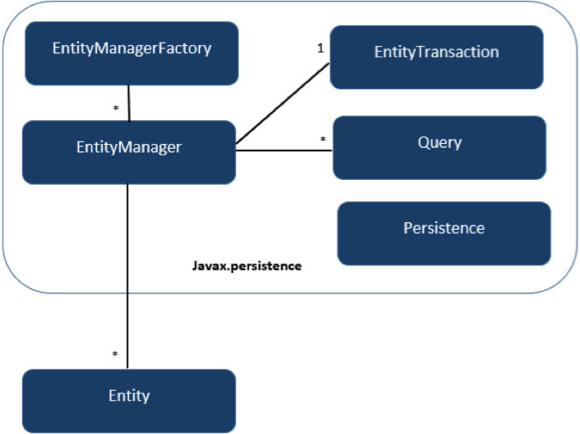
Before JPA we use JDBC:

* First, you have to database and tables manually
* Load the driver
* Get the connection object
* Statement or preparedStatementor callable statement
* Execute, executeUpdate, executeQuery
* If resultset is true
* Iterate the result set mapping into java object
* Java object put inside the collection
* Close the Statement or preparedStatement callable statement
* Close the connection

|  |  |
| --- | --- |
| JDBC | JPA |
| It Stands for Java Database Connectivity | It Stand for Java Persistence API |
| JDBC is a low level standard for interaction with databases. | JPA is higher level standard for the same purpose. |
| JDBC allows you to do more things with the Database directly, but it requires more attention. | JPA allows you to use an object model in your application which can make your life much easier. |
| Some task may be Resolved more Efficiently using JDBC | Some tasks can not be solved efficiently using JPA |
| It is Slower in process | It is relatively faster than JDBC |
| In JDBC, All the details needed for CRUD operations, such as table names, col names are exposed. | Where as in JPA you can specify those details of database metadata, but with the use of Java Annotations. |
| Shifting from one Database Provider to another is tedious and time taking process. | Shifting to another Database provide is relatively easier. |

What is JPA?

* The Java Persistence API (JPA) is a specification of Java. It is used to persist data between Java object and relational database. JPA acts as a bridge between object-oriented domain models and relational database systems.
* As JPA is just a specification, it doesn't perform any operation by itself. It requires an implementation. So, ORM tools like Hibernate, TopLink and iBatis implements JPA specifications for data persistence.
* JPS is a specification or standard to persist java objects into a database and has classes and interfaces that will help you to persist java objects into the database.
* JPA API’s are placed inside the package called javax.persistence.
* It does not have specification
* How to do is an implementation. Hibernate is one of the impl of JPA.
* We use ORM here
* Hibernate implements the classes and instances of JPA.
* Hibernate is called as ORM tool which impl JPA
* We have other tools eclipse link (ORM tool) which also impl JPA, same as Top-link
* JPA standard Query we have written in hibernate will also be run in other tools as well.
* In this project we are using the latest version 2.2



# **JPA Object Relational Mapping**

Object Relational Mapping (ORM) is a functionality which is used to develop and maintain a relationship between an object and relational database by mapping an object state to database column. It is capable to handle various database operations easily such as inserting, updating, deleting etc.



## **ORM Frameworks**

Following are the various frameworks that function on ORM mechanism: -

* Hibernate (JAVA)
* TopLink
* ORMLite
* Ibatis (.net)
* JPOX
* Eclipse clink

## **Mapping Directions**

Mapping Directions are divided into two parts: -

* **Unidirectional relationship -** In this relationship, only one entity can refer the properties to another. It contains only one owing side that specifies how an update can be made in the database.
* **Bidirectional relationship -** This relationship contains an owning side as well as an inverse side. So here every entity has a relationship field or refer the property to other entity.

## **Types of Mapping**

Following are the various ORM mappings: -

* **One-to-one -** This association is represented by @OneToOne annotation. Here, instance of each entity is related to a single instance of another entity.
* **One-to-many -** This association is represented by @OneToMany annotation. In this relationship, an instance of one entity can be related to more than one instance of another entity.
* **Many-to-one -** This mapping is defined by @ManyToOne annotation. In this relationship, multiple instances of an entity can be related to single instance of another entity.
* **Many-to-many -** This association is represented by @ManyToMany annotation. Here, multiple instances of an entity can be related to multiple instances of another entity. In this mapping, any side can be the owing side.

In general, entity is a group of states associated together in a single unit. On adding behaviour, an entity behaves as an object and becomes a major constituent of object-oriented paradigm. So, an entity is an application-defined object in Java Persistence Library.

## **Entity Properties**

These are the properties of an entity that an object must have: -

* **Persistability -** An object is called persistent if it is stored in the database and can be accessed anytime.
* **Persistent Identity -** In Java, each entity is unique and represents as an object identity. Similarly, when the object identity is stored in a database then it is represented as persistence identity. This object identity is equivalent to primary key in database.
* **Transactionality -** Entity can perform various operations such as create, delete, update. Each operation makes some changes in the database. It ensures that whatever changes made in the database either be succeed or failed atomically.
* **Granuality -** Entities should not be primitives, primitive wrappers or built-in objects with single dimensional state.

## **Entity Metadata**

Each entity is associated with some metadata that represents the information of it. Instead of database, this metadata is exist either inside or outside the class. This metadata can be in following forms: -

* **Annotation -** In Java, annotations are the form of tags that represents metadata. This metadata persist inside the class.
  + **@Entity is an javax.persistence Annotation which need to be placed just before the class you want to persist.**
  + **@Id is also an javax.persistence Annotation which need to be placed just before the variable you wanted to have primary key as a column.**
* **XML -** In this form, metadata persist outside the class in XML file.

# **JPA Entity Manager**

Following are some of the important roles of an entity manager: -

* The entity manager implements the API and encapsulates all of them within a single interface.
* Entity manager is used to read, delete and write an entity.
* An object referenced by an entity is managed by entity manager.

## **Steps to persist an entity object.**

**1) Creating an entity manager factory object**

The **EntityManagerFactory** interface present in **java.persistence** package is used to provide an entity manager.

EntityManagerFactory emf=Persistence.createEntityManagerFactory("Student\_details");

**Persistence -** The Persistence is a bootstrap class which is used to obtain an EntityManagerFactory interface.

**createEntityManagerFactory() method -** The role of this method is to create and return an EntityManagerFactory for the named persistence unit. Thus, this method contains the name of persistence unit passed in the Persistence.xml file.

**What createEntityManagerFactory() actually does is, it will go to the persistence.**xml file look for the tag with unit ‘ems’ and loads the config like postgresssql driver and other credentials whatever its there inside <persistence></persistence> tag into a object called ‘emf’ of interface EntityManagerFactory. Go to step 2.

**2)Obtaining an entity manager from factory.**

EntityManager em=emf.createEntityManager();

* **EntityManager -** An EntityManager is an interface
* **createEntityManager() method -** It creates new application-managed EntityManager

**3)** **Intializing an entity manager.**

em.getTransaction().begin();

* **getTransaction() method -** This method returns the resource-level EntityTransaction object.
* **begin() method -** This method is used to start the transaction.

**4)** **Persisting a data into relational database.**

em.persist(s1);

* **persist() -** This method is used to make an instance managed and persistent. An entity instance is passed within this method.

**5) Closing the transaction**

em.getTransaction().commit();

**6)  Releasing the factory resources.**

emf.close();

em.close();  // This method is used to releasing the factory resources.

For Video Demo : Watch 02-08-2021 11AM Video on Revature Pro

For Code Demo : Look for the Project **JPADemo.**

**Hibernate:**

* Hibernate is a JPA provider, JPA has classes and interfaces which can be implemented by hibernate.
* Hibernate internally converts HQL to JDBC query and execute into respective DB
* Query languges we write on hibernate in HQL, Native SQL and CREITERIA API
* When we want to save anything we use **persist()**
* For remove we use **remove()**
* For finidng anything we use **find().**
  + This method is converting into select method when goes to jdbc smiliarly persist is also for persist(), remove() delete

Entity manager factory and enity manager are the interfaces

Javax.persistance package

persistence.xml is a config file for JPA

hibernate.cfg.xml is config file for hibernate

config file will be placed in **src/main/resource.**

Use @Enity from persistence

@id primary key

@GenerateValue for generating auto gen values

@GenericGenerator(name=””,strategy=””)

@Column(name = “column\_name”) //use it when you don’t want the same which is there in the model

<dependencies> <!-- https://mvnrepository.com/artifact/javax.persistence/javax.persistence-api --> <dependency> <groupId>javax.persistence</groupId> <artifactId>javax.persistence-api</artifactId> <version>2.2</version> </dependency> <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core --> <dependency> <groupId>org.hibernate</groupId> <artifactId>hibernate-core</artifactId> <version>5.5.5.Final</version> </dependency> <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>42.2.23</version>

</dependency>

</dependencies>

**04-08-2021**

**3pm**

**EmshibernateCRUDHQL**

**HQL:** Hibernate Query Language is same as SQL but it doesn’t depend on

the table of the database. Instead of table name, we use class name in HQL.

So it is database independent query language.

### **Advantage of HQL**

There are many advantages of HQL. They are as follows:

* database independent
* supports polymorphic queries
* easy to learn for Java Programmer

### **Query Interface**

It is an object oriented representation of Hibernate Query. The object of Query can be obtained by calling the createQuery() method Session interface.

The query interface provides many methods. There is given commonly used methods:

1. **public int executeUpdate()** is used to execute the update or delete query.
2. **public List list()** returns the result of the relation as a list.
3. **public Query setFirstResult(int rowno)** specifies the row number from where record will be retrieved.
4. **public Query setMaxResult(int rowno)** specifies the no. of records to be retrieved from the relation (table).
5. **public Query setParameter(int position, Object value)** it sets the value to the JDBC style query parameter.
6. **public Query setParameter(String name, Object value)** it sets the value to a named query parameter.

**EmshibernateCRUDNativeSQL**

**EmshibernateCRUDNamedQuery**

**EmshibernateCRUDNativeNamedQuery**

**Level1Level2CacheDemo**

**05-08-2021**

**10:20am**

**DIFFERENCE BETWEEN XML AND HTML**

**Difference between JSON and XML**

By using DTD in xml we set our own rules.

so that we limit the attributes we don't want.

if user tries to write it it will give you the error

XSD : xml schema definition

DTD: Document Type Definitoon

using in pom.xml

we can check xml is well formed or not by using DTD and XSD

DTD is making sure that whatever user is writing .xml file accroding to hibernate rules.

### **XML Overview**

# **XML - eXtensible Markup Language**

* like HTML, not a programming language but a markup language
* designed to transport and store data in a way that is both human and machine readable
* language agnostic

<?xml version="1.0" encoding="UTF-8"?>

<students>

<student>

<firstName>Veronica</firstName> <lastName>Jones</lastName>

</student>

<student>

<firstName>Paul</firstName> <lastName>McCormick</lastName>

</student>

<student>

<firstName>Lola</firstName> <lastName>Nunez</lastName>

</student>

</students>

{students:[

{ firstName:"Veronica", lastName:"Jones" },

{ firstName:"Paul", lastName:"McCormick" },

{ firstName:"Lola", lastName:"Nunez" }

]}

{"student":[

{ "firstName":"Veronica", "lastName":"Jones" },

{ "firstName":"Paul", "lastName":"McCormick" },

{ "firstName":"Lola", "lastName":"Nunez" }

]}

* faster parsing
* more compatible w/ JavaScript
* less verbose
* more universal (tags won't change with different developers)

#### **Well-Formed vs. Valid XML**

Well Formed XML

* Follows basic syntatic rules
  + begins with XML declaration
  + unique root element
  + starting and ending tag must match
  + elements are case sensitive
  + elements must be properly nested
* Well formed document is not necessarily valid

Valid XML

* XML which follows a predefined structure
* Validated using:
  + Document Type Definition
  + XML Schema Definition
* Valid XML will also be well formed

#### **Document Type Definition (DTD)**

* written in its own DTD syntax
* define element names and relationships
* can define range of values
* PCDATA and CDATA - parsed character data and unparsed character data
  + PCDATA is text that will be parsed by a parser. Tags inside the text will be treated as markup and entities will be expanded.
  + CDATA is text that will not be parsed by a parser. Tags inside the text will not be treated as markup and entities will not be expanded.
* can't be more specific with types

#### **XML Schema Definition (XSD)**

* written in XML
* can declare simple and complex types
* can declare number and order of child elements
* can declare default and fixed values for elements and attributes

#### **XML Namespace**

* anyone can create their own markup with their own tags
* namespace allows us to differentiate from people using the same tag name for different purposes
* namespace declaration xmlns:prefix = "namespace"
* unique identifier URI - so if you have a registered URL
* doesn't necessarily have anything to do

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs=".../xmlSchema">

<xs:students>

<xs:student>

<xs:firstName>Veronica</xs:firstName> <xs:lastName>Jones</xs:lastName>

</xs:student>

<xs:student>

<xs:firstName>Paul</xs:firstName> <xs:lastName>McCormick</xs:lastName>

</xs:student>

<xs:student>

<xs:firstName>Lola</xs:firstName> <xs:lastName>Nunez</xs:lastName>

</xs:student>

</xs:students>

#### **Parsing XML**

**JAXP - Java API for XML Processing**

Document Object Model (DOM)

* loads full xml file into memory and creates a tree representation of it
* much faster for small/medium sized files because of in-memory retrieval
* can get very expensive (memory-wise) for larger files

Simple API for XML Parsing (SAX) - read only - event based - "streams" XML data without loading all of it into memory

#### **XML Binding**

**JAXB - Java and XML Binding**

**Marshalling**: converting XML to Java Objects

**UnMarshalling**: converting Java Objects to XML

By using DTD in xml we set our own rules.

so that we limit the attributes we don't want.

if user tries to write it it will give you the error

XSD : xml schema definition

DTD: Document Type Definitoon

using in pom.xml

we can check xml is well formed or not by using DTD and XSD

DTD is making sure that whatever user is writing .xml file accroding to hibernate rules.

----------------------------------------------

11:40AM

Criteria API: Hibernate criteria is a programmatic way of wrtiting and typesafe way to fetch data from the relational database.

-> you have to build a CriteriaBuildQuery

CriteriaBuilder builder =session.getCriteriaBuilder();

->CriteriaBuilder is an interface here

createCriteriaDelete/update

builder.createQuery(Employee.class) //this method will give you Criteria API select Query.

CriteriaQuery<Employee> cQuery = Builder.CreateQuery(Employee.class); //createQuery

cQuery.from(Employee.class); //create a query from Employee class

Root<Employee> root = cQuery.from(Employee.class)

**2:30pm**

**One-to-one relationahip**

**06-08-2021 many to one relationship**

**One to many should have list here**

Oncascade-delete.update,deatch,insert

One to one by default in hibernate is lazy(fetch only primary key objects) fetching here we are using .find()

One to many default is Eager(fetches primart key objects as well as foreign key objcts) here we are using .getCountry() improve performance

4:15 docker

Conternatization means deploying your image into a continer or docker environment.

Docker Volume : A storage for the dcoker container

How docker communicate with each other that’s called the container

In order to convert .jar/.war into a docker image we use

Docker build –t image\_name . –f

Docker run –name container name imagename or id

<build> <pluginManagement> <plugins> <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-jar-plugin</artifactId> <configuration> <archive> <manifest> <addClasspath>true</addClasspath> <classpathPrefix>libs/</classpathPrefix> <mainClass> com.test.hellp.App </mainClass> </manifest> </archive> </configuration> </plugin> </plugins> </pluginManagement> </build>

4:45

Maven assembly plugin

5:30

Devops

Dev team : Coding debug, test and pass to testin gteam

Testing team : will do testing and pass it on to the operation team

Operation team : build and deploy on the server

Build testing and operation want to automate it done using Jenkins

Jenkins is a automation tool, it will pull the file from git and put it into Jenkins area and then create a docker image and deploy it to the container.

CI and CD: contininous integration as sson as make changes in code and push it on git autojenkins pull the code and intrgrate and deploy it.

**9-aug-2021/11:07am**

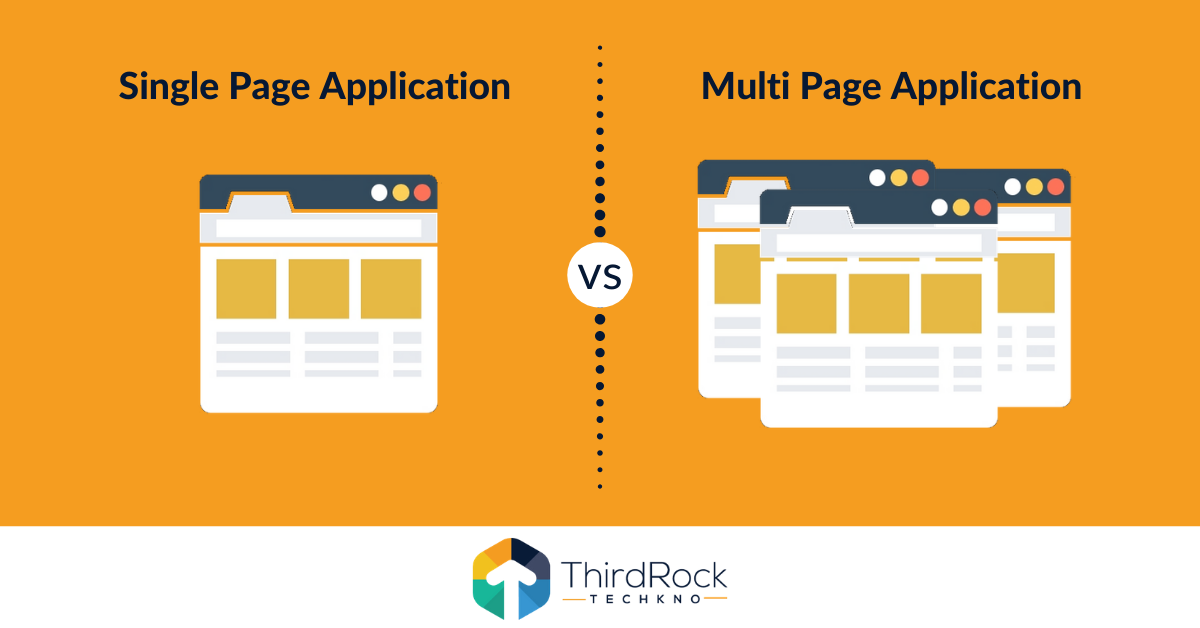
**Before Angular**: Back in the day, web developers used VanillaJS and jQurey to develop dynamic websites but, as the logic of one’s website grows, the code became more and more tedious to maintain. And development process becomes slower.

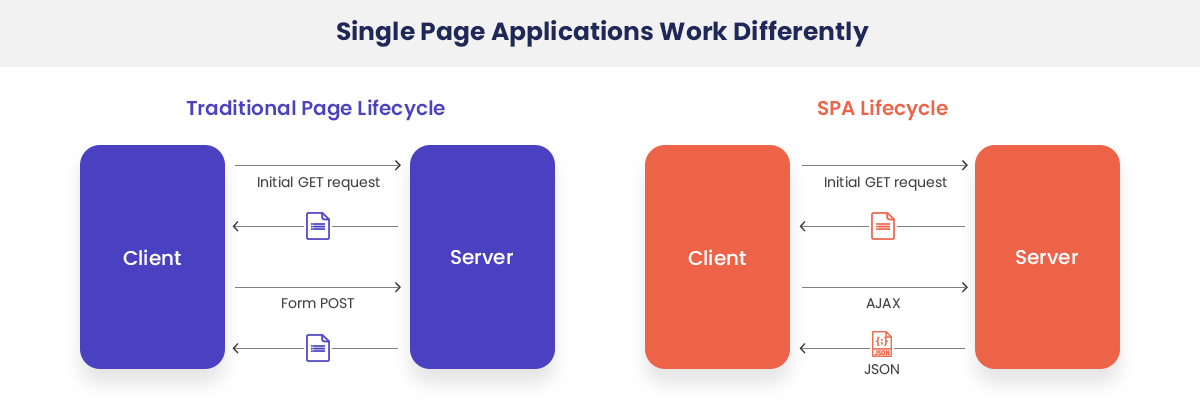
For applications that uses complex logic, developers had to put in some extra effort to maintain the separation of the code, Also jQuery did not provide facilities for data handling across views.

For tackling the above p.roblems, client-side frameworks like Angular came into the picture, which made developers life easier by handling separation of concerns and dividing code into small manageable pieces i.e components.

**Single Page Application(SPA):-**

* A single page application is a web app or site, which loads an entire page only once.
* Any interactions with the page or subsequent pages do not require a round trip to the server, which means the entire page is not reloaded
* Single page applications are a great way to create engaging and exceptional experiences for your website visitors
* Examples of SPAs include Netflix, Gmail, Google Maps, Facebook, Twitter
* SPA uses AJAX and HTML5 to build responsive web applications





* Advantages:
  + Fast and Responsive
  + Caching capabilities ( for the first time web page get loaded entirely) - offline
  + Pleasant user experience
* Disadvantages
  + SEO – google search
  + Security – Cross site scripting attack

(To avoid Server side validation (Bean Validator, Hibernate Validator) –code, filter)

Website :

Static content : The content will not change.

Dynamic Content : The content will changes frequently

Static and dynamic pages -> Web App

(HTML,CSS,JS,JQUERY,BOOTSTRAP,ANGULAR)

Will refresh data everytime goes to server and load multiple pages

Where as in single it only loads the part of the pages

Multiple Application :

Single Page Application(SPA):-

* A Single page Application is a web app or site, which loads an entire page at once

Nodejs

**AngularJS: AngularJS is an open-source front-end JS framework for creating single page web applications developed by Google in 2002.**

**Drawback: Large bundle size,performance issues and did not support mobile devices.**

**Angular: Angular is an open-source web application based on TypeScript, a programming language developed by Microsoft.**

**Why use angular?**

**1.easy to use**

**2.provide MVC Architecture**

**3.It is compatible for both desktop and mobile app**

**4.Two-way binding**

**5.Flexible and extensible**

|  |  |  |
| --- | --- | --- |
|  | **AngularJS** | **Angular** |
| **Architecture** | AngularJS uses MVC architecture, Where **Model** contains the business logic, **Controller** process the information and **View** shows the information present in the **Model.** | Angular replaces **Controllers** with **Components. Components** are nothing but directives with a predefined template. |
| **Language** | AngularJS uses JavaScript language, which is a dynamically typed language. | Angular uses TypeScript language, which is a statically typed language and is a superset of JavaScript. By using statically typed language, Angular provides better performance while developing larger applications. |
| **Mobile Support** | AngularJS does not provide mobile support. | Angular is supported by all popular mobile browsers. |
| **Structure** | While developing larger applications, the process of maintaining code becomes tedious in the case of AngularJS. | In the case of Angular, it is easier to maintain code for larger applications as it provides a better structure. |
| **Expression Syntax** | While developing an AngularJS application, a developer needs to remember the correct ng-directive for binding an event, or a property. | Whereas in Angular, property binding is done using "[ ]" attribute and event binding is done using "( )" attribute. |

* Every Angular application consists of components and templates which the browser cannot understand. Therefore, all the Angular applications need to be compiled first before running inside the browser.

**Angular provides two types of compilation:**

 JIT(Just-in-Time) compilation: In JIT compilation, the application compiles inside the browser during runtime.

 AOT(Ahead-of-Time) compilation: in the AOT compilation, the application compiles during the build time.

**The advantages of using AOT compilation are**:

* Since the application compiles before running inside the browser, the browser loads the executable code and renders the application immediately, which leads to faster rendering.
* In AOT compilation, the compiler sends the external HTML and CSS files along with the application, eliminating separate AJAX requests for those source files, which leads to fewer ajax requests.
* Developers can detect and handle errors during the building phase, which helps in minimizing errors.
* The AOT compiler adds HTML and templates into the JS files before they run inside the browser. Due to this, there are no extra HTML files to be read, which provide better security to the application.
* By default, angular builds and serves the application using JIT compiler.
  + ng build
  + ng serve
* For using AOT compiler following changes should be made
  + ng build –aot
  + ng serve—aot

**Angular file explorer:**

**.editconfig :** it is used to provide config for the code editors

angular.json : execution of angular starts here, all the CLI default config for all projects in the workspace including config options for build serve test tools that the cli used such as tslint,karma and protractor

.gitignore : intentionally unctracked file that should be ignored by GIT

Karma.js : karma is a test runner consist of app specific karma config

Package.lock.json : Version info all packges installed for node modules installed by npm client

Package.json : This file config npm file depn that are available for all the projects that are available

Tsconfig.app.json : all the app specific ts config inculidng ts and angular compiler options

Tsconfig.json: Default ts config for project in the workspace

typescript.spec.json : ts config for all the app

tslint.jsonThis is an app specifc ts lint config

main.ts file : It is an entry point for our application, it compiles our application with JIT compiler and bootstrap the root module.

Pollyfills.ts : Provides scripts for browser supports.

**How angular executed?**

Angular.json -> main.test -> app.modules.ts ->[AppComponent] -> app.component.ts -> index.html

**Components in Angular:**

**CMD:** $ng generate component componentName or $ng g c componentName

**Description:** In Angular, components are the basic building blocks, which control a part of the UI for any application.

A component is defined using the @Component decorator. Every component consists of three parts, the template which loads the view for the component, a stylesheet which defines the look and feel for the component, and a class that contains the business logic for the component.

One can see the generated component inside src/app/test folder. The component will be defined inside test.component.ts and this is how it looks:

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-test',

templateUrl: './test.component.html',

styleUrls: ['./test.component.css']

})

export lass TestComponent implements OnInit {

constructor() {}

ngOnInit() {

}

}

As we can see in the above image, our component is defined with **@Component** decorator.

**Module in Angular:**

**CMD:** $ng generate module moduleName or $ ng g m test-module

**Description:** A module is a place where we can group components, directives, services, and pipes. Module decides whether the components, directives, etc can be used by other modules, by exporting or hiding these elements. Every module is defined with a @NgModule decorator.

**By default, modules are of two types:**

* Root Module
* Feature Module

Every application can have only one root module whereas, it can have one or more feature modules.

A root module imports **BrowserModule,** whereas a feature module imports **CommonModule.**

In the application that we created before, one can see that the root module is defined inside **app.module.ts** and this is how it looks:

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

import { TestComponent } from './test/text.component';

@NgModule({

declarations: [

AppComponent,

TestComponent

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

We can see in the above image that the component we created earlier is already imported in the declarations array.  
  
To create a feature module, run the following command:

ng g m test-module

The module is created inside the src/app/test-module/test-module.module.ts file:

import { NgModule } from '@angular/core';

import { CommonModule } from '@angular/common';

@NgModule({

declarations: [],

imports: [

CommonModule

]

})

export class TestModuleModule { }

As one can see, **CommonModule** is imported since this is a feature module.

**Services in Angular:**

**Cmd:** $ng g s test-service

**Description:** Services are objects which get instantiated only once during the lifetime of an application. The main objective of a service is to share data, functions with different components of an Angular application.

A service is defined using a **@Injectable decorator**. A function defined inside a service can be invoked from any component or directive.

The service will be created inside src/app/test-service.service.ts:

import { Injectable } from '@angular/core';

@Injectable({

providedIn: 'root'

})

export class TestServiceService {

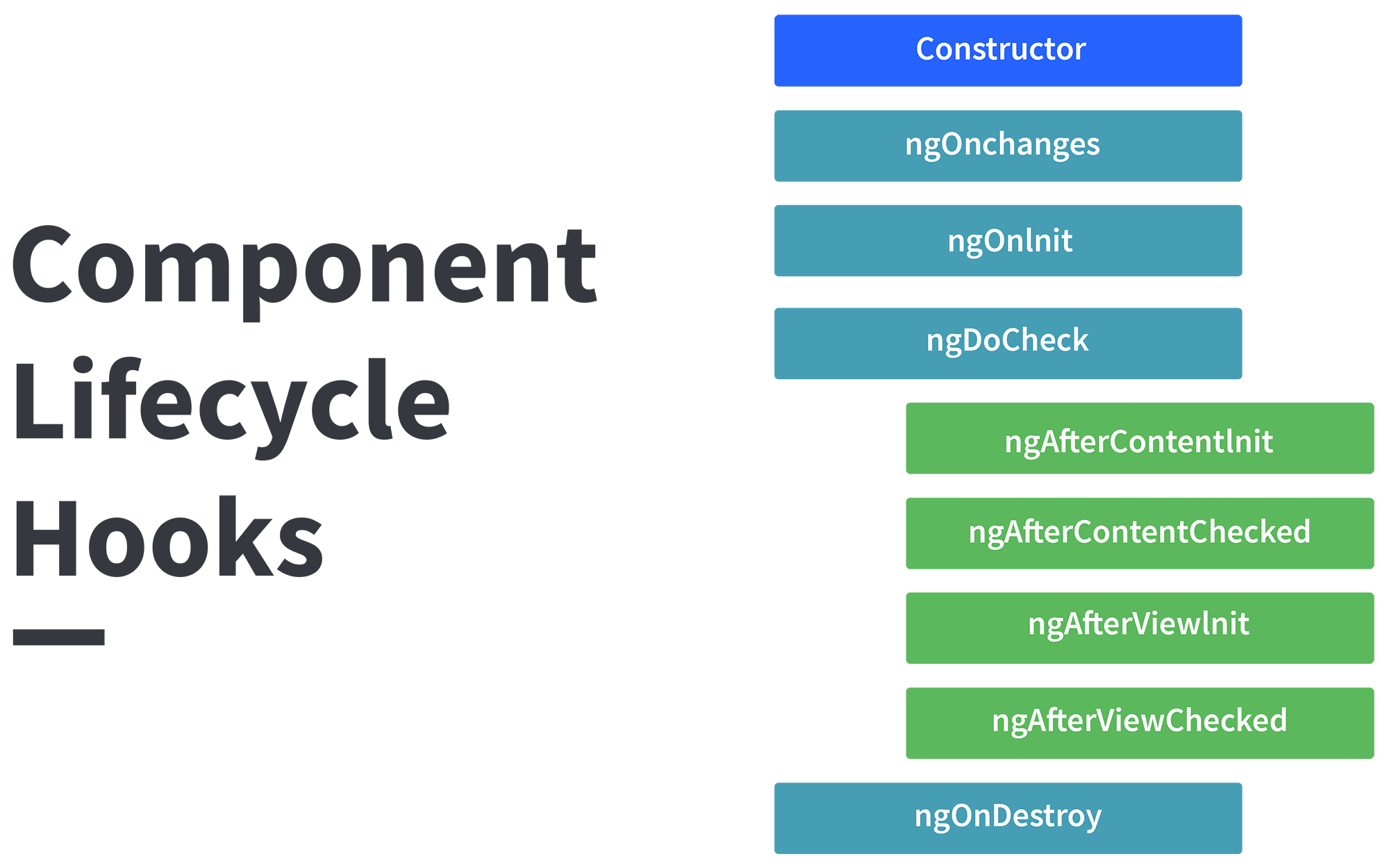
constructor() { }

}

Any method/function defined inside the TestServiceService class can be directly used inside any component by just importing the service.

What are lifecycle hooks in Angular? Explain a few lifecycle hooks.

Ans: Every component in Angular has a lifecycle, different phases it goes through from the time of creation to the time it's destroyed. Angular provides **hooks** to tap into these phases and trigger changes at specific phases in a lifecycle.



**ngOnChanges( )** This hook/method is called before **ngOnInit** and whenever one or more input properties of the component changes.  
This method/hook receives a SimpleChanges object which contains the previous and current values of the property.  
  
**ngOnInit( )** This hook gets called once, after the **ngOnChanges** hook.  
It initializes the component and sets the input properties of the component.  
  
**ngDoCheck( )** It gets called after **ngOnChanges** and **ngOnInit** and is used to detect and act on changes that cannot be detected by Angular.  
We can implement our change detection algorithm in this hook. **ngAfterContentInit( )** It gets called after the first **ngDoCheck** hook. This hook responds after the content gets projected inside the component.  
  
**ngAfterContentChecked( )** It gets called after **ngAfterContentInit** and every subsequent **ngDoCheck**. It responds after the projected content is checked.  
  
**ngAfterViewInit( )** It responds after a component's view, or a child component's view is initialized.  
  
**ngAfterViewChecked( )** It gets called after **ngAfterViewInit**, and it responds after the component's view, or the child component's view is checked.  
  
**ngOnDestroy( )** It gets called just before Angular destroys the component. This hook can be used to clean up the code and detach event handlers.  
  
Let’s understand how to use **ngOnInit** hook, since it’s the most oftenly used hook. If one has to process lot of data during component creation, it’s better to do it inside **ngOnInit** hook rather than the constructor:

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-test',

templateUrl: './test.component.html',

styleUrls: ['./test.component.css']

})

export class TestComponent implements OnInit {

constructor() { }

ngOnInit() {

this.processData();

}

processData(){

// Do something..

}

}

As you can see we have imported OnInit but we have used **ngOnInit** function. This principle should be used with the rest of the hooks as well.

3:15pm TypeScript intro and fundamentals

12-08-2021

|  |  |
| --- | --- |
| TypeScript | JavaScript |
| Typescript is a modern age Javascript development language. It is a statically compiled language to write clear and simple Javascript code. It can be run on Node js or any browser which supports ECMAScript 3 or newer versions.  It is a superset of JS | JavaScript is a scripting language which helps you create interactive web pages |
| Typescript code needs to be compiled first | JavaScript code doesn’t need to compile. |
| Typescript uses concepts like types and interfaces to describe data being used | JavaScript has no such concept. |
| Typescript is a powerful type system, including generics & JS features for large size project | JavaScript is an ideal option for small size project. |
|  |  |

| **Parameter** | **Typescript** | **JavaScript** |
| --- | --- | --- |
| What is | Powerful type system, including generics & JS features | Lightweight, interpreted, object-oriented language with first-class functions |
| Data Binding | TypeScript uses concepts like types and interfaces to describe data being used. | No such concept is available with JavaScript. |
| Ecosystem | The Ecosystem is quite powerful and intuitive. Thus, it allows you to statically type various type of idiomatic JavaScript features like union types, intersection, discriminated union. | Javascript offers the option to explore and create code without a build step. |
| Npm package | With Typescript, many npm packages either come with static type definitions or have an external one that is easy to install. | Javascript offers the option to explore and create code without a build step. |
| Learning curve | Stiff learning curve. Requires prior scripting knowledge. | Flexible and easy to learn, scripting language. |
| Prototyping | Typescript has a feature of prototyping. | JavaScript doesn't have this feature. |
| Community | Typescript does not have a large community of developers. | The JavaScript has a huge community of developers |
| Compilation | TypeScript code needs to be compiled | No need to compile JavaScript. |
| Annotation | To get the most out of TypeScript features, developers should constantly annotate their code. | No Annotations Required is need for Javascript. |
| Famous Company using the Technology | Asana, Clever, Screen award | Airbnb, Codecademy, Instagram |
| Salary | The average salary for "Typescript developer" ranges from approximately $148,027 per year in United States | The average salary for a Javascript Developer is $110,777 per year in the United States. |

TypeScript : TypeScript is an programming language also a superset of JS developed by Miscrosoft.

Ts is oops ->JS is just scripting

Ts can support static features(MODULES, INTERFACES)-> JS cannot

To install typescript

$npm install -g typescript

how to transpile your TS code

$tsc name\_of\_tsfile.ts –watch //so that we don’t have to compile it again n again

It will generate .js file

For running ts file

$node file\_name.js

tsc --init

Decorators:

Nothing but it’s a function. It will execute during runtime

Metadata or meta info or addi info to the class exe at runtime

Constructor can executed before class executed using @info

## **Webpack**

In our web application, we use many javascript files that are added into the HTML pages via <script> tags. For each user request, the browser loads these bunch of script files inside the HTML page. This is inefficient as it reduces the page speed since the browser requests each script file separately.

This can be solved by **bundling** several files together into one file to be downloaded by the browser in one single request.

**Module bundlers** are used to bundle a group of JavaScript modules with their dependencies and merge them into a single file in the correct order, which can be executed by the browser.

[Webpack](https://webpack.js.org/) is a powerful static module bundler for JavaScript applications that packages all modules in our application into a bundle and serves it to the browser.

Webpack builds a [dependency graph](https://webpack.js.org/concepts/dependency-graph/) when it processes the application. It starts from a list of modules defined in its config file (webpack.config.js) and recursively builds a dependency graph that includes every module our application needs, then packages all of those modules into a small bundle that can be loaded by the browser.

Angular CLI:

* Run the command npm install/update -g @angular/cli on the terminal to install the Angular CLI using npm.
* Run the CLI command ng new my-app to create a new angular app with the my-app name.
* The Angular CLI includes a server so that we can easily build and serve your app locally. First, go to the my-app workspace folder and Launch the server by using the CLI command ng serve --open.

The ng serve command launches the server on HTTP port 4200, which watches our files and rebuilds the app as we make changes to those files. The --open (or just -o) option automatically opens the browser to [http://localhost:4200](http://localhost:4200/).

After running the ng server -o command, we will see

->main.ts file is the first file which angular app hits.

Inside the main we have bootstrap

Bootstrap means automatically starts

AppModule it’s a class in app.module.ts it is also a root module

If any class is decorated with @NgModule it will be angular module.

Whatever u r creating weather It is component,pipe,services all must be attach to root module i.e AppModule

Pipe will use for presentation how do we present our data.

Allthe components will be in declaration : [Appmodule,…..]

Imports : [browsermodule,approutingmodule]

Providers: [] //service provides modules will be provided

Bootstrap:[AppModule]

Component is nothing but a typescript class.

Angular app can have many components.

The module bootstrap[AppModule] which component is the root component we specify inside []

Every component will have one .ts file, .html file, .css file and spec.ts file

Appcomponent.spec.ts file is used for unit testing cases

Steps to create a component in ANGULAR:

$ng generate component component\_name

Or ng g c component\_name

5:25pm

Angular Directive: It allows to manipulate DOM Tree just like we do in JS

1. component Directive: How component should be processed, instantiated and used at runtime.

Ex: Componenet creation basically

<app-root></app-root> is a directive tag

2.Structure Directive : It is used to manipulate/change the structure of the DOM ELEMENTs

Structure dir starts with \*

Sd we have

1.\*ngIf : it will renders the element content based on condition.

13-aug-2021

2.\*ngFor:

3.\*ngSwitch:

String polation will show dynamic data which comes from the TS file {{ts\_var\_name}}

1. Attribute Directive: it will effect the attributes of html tags for ex : h1 have id attribute or other attributes it will effect only those only.

Here we have to use [] square brackets instead of {{}}

[] is a one way binding

ngClass :changes the DOM classes dynamically

ngStyle :it can changes only 1 style dynamically

3pm mypipe project

Pipe: It is used to transform the data.it does not change the original value

Can be used to format present data in the UI

* + - * + Symbol representation |

Types of Pipes:

1.Builtin Pipes

->Number Pipe

->Date Pipe

->Currency Pipe

->Uppercase Pipe

->Lowercase Pipe

->Slicing Pipe

2.Create a Custom Pipe:

Steps

i.create a TS class and that class should decorated with @Pipe

ii.TS pipe class must implement one interface i.e PipeTransform

iii. Must Override transform()

iv. once TS class is creaed you need to register custom pipe to @NgModule TS Class in app.module declaration in decorator.

v. cmd to create custom a pipe $ng generate pipe pipename

3. Parametrized Pipe:

Pipename : ‘passparameter now’,morethanonepipe’’

4:30pm Bindig

Binding : To bind data from .ts file to .html or UI.

Name : Suleman in .ts file

{{Name}} binding in .html

There are two ways of Binding

1.oneway binding : .ts file to .html file

2.Property binding : can be binded by using [].

In .html file

<input type="text" name="username" [value]="username" class="text">

In app-component.ts

username= "sul123";

output:

[sul123]

**Even binding:**  from the ui side u can bind to the .ts file.

Whatever u enter into html textbox that gonna be binded in .tsfile variable

5:30

Routeroutlet is a placeholder for components html file.

Router Guards : authorization to users to control the access to which path can access by the user.

**routerdemo ng app**

Router in angular

Replace  href="register"

With router link

16-8-2021

http client : The HttpClient is used to

Observable(Interface): it always listens the responses from the server.

obervable asynchrounsly reads the data from the source.

Ex: When http is making an request and server is responding that response is read by observable. And keep it with itself, that data will be expose to user when user calls subscribe() method.

Cmd to create component : ng g c componentname;

**App.routing.moudle.ts** where we give routes

Const routes : Routes = [

{path:”hyperlinkname”,component:componentName},{path:”path or hyperlink name”,component:componentName}

]

->we remove href in app.component.html and replace it with <a routerLink=”/register”>Add Employee</a>

<router-outlet></router-outlet>

Angular Form:

There are approaches to make angular form

1)Template Driven form – it is use when want to create simple forms

2)Reactive form - It is use when we want to handle complex data forms

Two important attributes in Reactive forms

i)FormGroup attribute: FormGroup is used with FormControl to track the value and validate the state of form control.

ii)FormControlName: for binding the form input to typescript variables we use ReactiveFormGroup..  
18-8-2021 10:20AM

**publisher and subscriber model**:

publisher is going to publish(event) data into EventQueueor channel in which publish will post the data. Consumer(Subsriber) will consume the data from the eventQueue, there could be n number of subscriber. Whose subscriber want to subscribe the data they can do it by subscribing into even channel.

Parent to child can comm using @Input

Child to parent can comm using @Output

2:30pm

Even emitters in Angular

An Evenmitter is used to emit a custom emitter.

Pending example with error..

**Testing jasmine and karma:**

In JUNIT5:

Testsuit:Multiple test cases written in a single name is called testSuit

Thru Runner we run all the test cases in TestSuit.

For java we have JNUIT

For Angular we have jasmine is an open source framework and default Test tool.

It is an BDD(Behavioural Driven Development ) :

Here we write the behavriour first and go for writing test cases

Behaviour in the sense TS function

function add(n1:number,n2:number){

return n1+n2;

}

@Test

In TDD we write testcases first then go for writing the code implementation

@Test

Public void testAdd(){

assertEquals(30,add(10,20))

}

.java

Public int add(int n1,int n2){

Return n1+n2;

}

In Jasmine Describe(TestSuit) -> A test

Specs or test script or test cases -> specs are defined by calling it(in java we have @Test)

Expect(just like assert in java) - > assert the expected value and actual value

Matcher(Equals,True,False,Null,NotNull) -> it is a chained matcher function, which takes expected value.

* 1. Describe has two parameter string and can be a arrow or any function.
     + Ex: Describe(“”,()=>{})

it(“testCaseName”, ()=>{testcasecode comes here}, it(anothertestcase,()=>),(),()

})

You can test compo,services,directive thru jasmine

Cd angulardemo/jasminedemo : $ng test

Karma is a tool for running test cases of jasmine test cases

4:50pm

**SDLC : Software Development Life Cycle**

The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software. in detail, the SDLC methodology focuses on the following phases of software development:

* Requirement analysis
* Planning
* Designing
* Coding
* Testing
* Deployment
* Maintenance

<https://www.javatpoint.com/software-engineering-software-development-life-cycle>

in waterwall model there is no proper interaction with the client if client need not like the feature and ask to change it, the dev has to go from starting to make changes which is very tedious.

For small project we can go with WaterFall model, else we have to switch some other methodology like agile.

What is Agile is?

Agile methodologies?

Ans: scrum and XP

We can approach agile principles?

We can achieve it by using methodologies like SCRUM and Extreme Programming

Product owner : is a skilled person who takes req from end users and instruct dev team to develop the product as It is.

Product owner will conduct a meeting inside meeting there will be scrum master deve team, Here product owner explains abt the product to the dev team. This meeting is called sprint planning meeting.

User stories

<https://www.mountaingoatsoftware.com/agile/user-stories>

product backlog : it contains a set of user stories that we wanted to implement

all priority stories will be put in one sprint and decide how long time it will going to take in the meeting. Later on it will going to put in sprint backlog to track the work.

There will be atleast 5 + 2 dev members in a team.

After completing **sprint**  the product will going to release.

Sprint Review : Will have sprint review they will review the functionality of the release. If any one of the stories not be able to complete it will be pull back to sprint backlog to complete with the new stories.

Retrospective meeting : how the sprint went what are the problmes they have gone all the positive and negative thing will be discuss.

In the next sprint release they will resolve the issues in sprint 2 and quickly develop the product.

if you 25 stories, product owner willpriotize the stories.

We have SCRUM Team:

In which we have

1)SCRUM Master : he just motivate dev team and make sure that their relationship is going good.

2) Dev Team  
---------------------------------------------------------------------------------------

19-aug-2021

Tight Coupling : Tight Coupling means one class is dependent on another class