Java

basics → oops

↓

variables

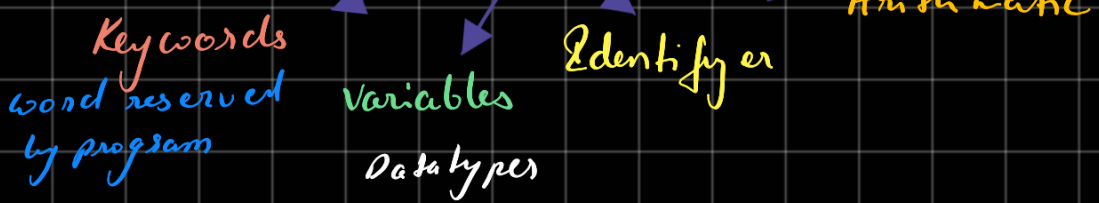
operations Identifiers Keywords

↓

Predefined

Token

smallest individual unit in a program.



Datatypes

assstruct

char

bit - 8 byte

short - 16 byte

int - 32byte

float - 32 byte

double - 64 byte

char - 8 byte

Data type

Primitive

int

float

double

boolean

Non Primitive

array

class

object

string

class →

A class is a group of object which have some common property. It is a template of blue print from which object is created.

It is a logical entity it cannot be physical.

Java class contains fields, methods, constructors..

this →

Block →

① Instance Initialization Block

② static Initialization Block

© It is executed before the constructor's.
Java compiler copy IIB in the constructor after the first statement of constructor. (super)

© Static keywords means that a particular member belongs to a type self rather then a instance of that type.

You can't use static keyword with class unless it is a inner class.
Static inner class is nested class which is static member of the other class.

Java supports →
- Static Instance
- Static Method
- Static Block
- Static Class

Static Inner class can also be a static, which means you can access without creating an object of outer class.

Inheritance →
is Java is a machanism in which one object accures all properties and behaviour of parent object.

Exception Handling →
Problem that arises during the execution time.

C++	Java
• Platform dependent	• Independent
• Software progrmg	• Application progrmg

- Goto
- operator overloading
- multiple inheritance
- support pointers
- interact with hardware

- Rust not
- not
- does not
- does not support
- does not

Private → with in the class.

Protected → with in the package.

Public → within, outside the class

Overloading
- In same class
- Static methods can be overloaded
- Static binding
- Compile time polymorphism
- Helps to extend functionality

Overriding
- only in derived class
- method either static or non-virtual overriding.
- Dynamic binding.
- Run time polymorphism
- overwrite existing function.

try → checks if there is an exception in the block.
- in the method

catch → if try block find exception, then the catch block will execute to handel the exception.

finally → will irrespective of there's any exception found or not in try block, will run.

final
- class variable

finally
- try { } catch { }

finalize
finalize ( ) { clean up;

method }

finally {

}

- a Keyword.
- It is a block
- Method
- deallocate the resources allocated by unused object.

Throw

- used to throw an exception
- used inside the method.

```
class m1() {
    throw new AECS   → created a new obj.
}
```

- can only handel one exception

Throws

- used to declare an exception
- used with method sign.

```
class m1() throws AECS
{
}
```

- can handle multiple exception.

Array list →

- made up of array, of dynamic size.
- stored in continuous manner.
- acts as list.
- slow because of shifting is required

Methods →

void add()

clear()

" " ArrayList extends — Abstract List ——

Linked List →

- Double linked list.
- stored in discontinuous manner.
- acts as list and queue.

Constructors →

- LinkedList()
- " (Collection c)

- faster than array list no bit shifting is required.

Methods →
- void add() · void clear()
- int size()

" " LinkedList extends AbstractSequentialList

Tree Map →
- based on red black tree.
- stores null values but "can't null key".
- stores key, value pairs in ascending order.
- store's unique data

" " TreeSet extends AbstractSet

Constructor →

TreeSet() TreeSet(Collection c)

Constructor →

TreeMap()

Methods →

boolean isEmpty() boolean remove()

void clear() int size()

Method → boolean isEmpty()
void clear()
int size()

" " HashMap extends AbstractMap, --

Hash Map →

Constructors →

HashMap() HashMap(int capacity)

- Stores in "key" "value" pair.
- Key is unique
- only one key can be "null", multiple values can be "null".
- stored in an continuous manner.

Hash Set →

" " HashSet extends AbstractSet

- uses 'hashing' mechanism

Constructor →

- stores in non contegous manner.
- unique elements only.
- allows null values.

HashSet()
" " (Collection c)
" (int capacity)

Methods →

void clear() → size()

Method overriding →

- Method with same name / parameters declared in parent class.
- Inheritance (relation ship required)

Can we overload static method?

No, static method can't be overridden.

performed with in the class

Why can't we?

Because static Method is bound with class.

where as instance method is bound with Object.

Method overloading →

- Method with same name but different parameters.
- Increases the readability.

Error // Exception

Summary of Differences	
ERRORS	**EXCEPTIONS**
Recovering from Error is not possible.	We can recover from exceptions by either using try-catch block or throwing exceptions back to caller.
All errors in java are unchecked type.	Exceptions include both checked as well as unchecked type.
Errors are mostly caused by the environment in which program is running.	Program itself is responsible for causing exceptions.
Errors occur at runtime and not known to the compiler.	All exceptions occurs at runtime but checked exceptions are known to compiler while unchecked are not.
They are defined in java.lang.Error package.	They are defined in java.lang.Exception package
Examples :	**Examples :**
java.lang.StackOverflowError, java.lang.OutOfMemoryError	Checked Exceptions : SQLException, IOException
	Unchecked Exceptions : ArrayIndexOutOfBoundException, NullPointerException, ArithmeticException.

checked

- compile time
- handle using try{} catch{}

unchecked

- Run time
- not required

Collection →

Collection is a framework in Java which provides architecture to store

and manipulating group of object.

like operations like sorting, searching, insertion, deletion etc.

Frame work provides ready made architecture.

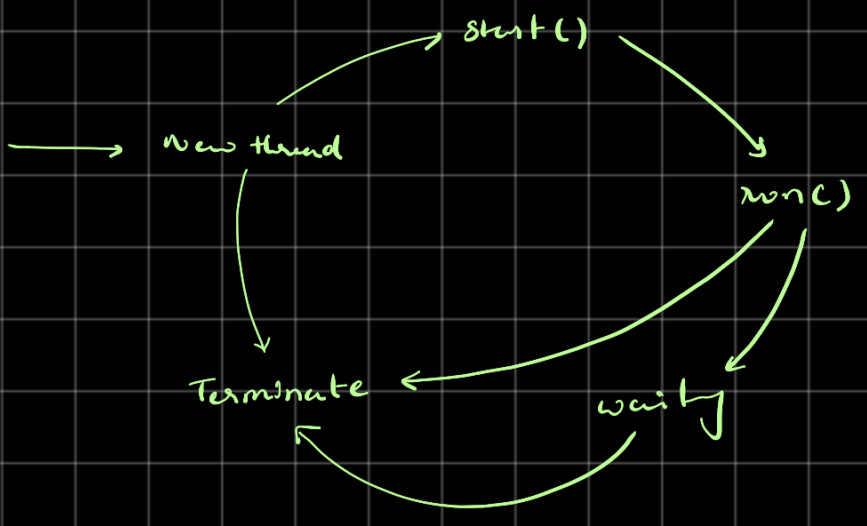
Threads →

- Smallest unit of processing.
- Threads are independent
- can perform multiple tasks at the same time
- Doesn't blocks the user.

How to create thread →

- By Extending Thread class
- By Implementing Runnable interface.

Life cycle of a thread →



Multithreading →

- executing multiple threads simultaneously.

Advantage

- Improves performance.
- Simultaneously can run multiple tasks.

Disadvantage

- Difficult to code
- " " debug.

Method →

Public void run()
" " start()
" " resume()
" " stop()

- " " testing.
- " " porting existing code.

ex →

```
class Multi extend Thread {
  --- run();
  --- println("Thread is running");

  --- main(--) {
      Multi t1 = new Multi()
      t1.start(); }
  }
```

By extending Thread class.

```
class Multi extend Runnable {
  --- run();
  --- println("Thread is running");

  --- main(String args[]) {
      Multi m1 = new Multi();
      Thread t1 = new Thread(m1);
      t1.start(); }
  }
```

By implementing Runnable Interface.

Why Synchronization is important?

Synchron is build around internal entity "monitor" / "lock".

- Different threads shouldn't try to access and change the same data at the same time.
- Threads must be synchronized.
- Only one thread can access the resource at given point of time.

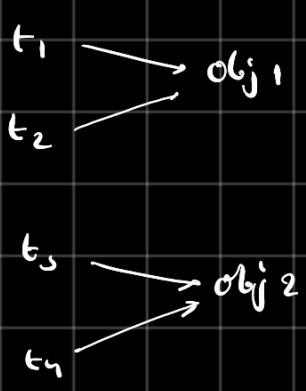
Thread Synchronization
- Mutual Exclusive
- Inter-Thread communication.

Mutual Exclusive →

Helps keeping threads from interfering while using.

eg → Synchronized method
Static synchronization

Static Synchronization →



- If make any static method synchronized, the lock/monitor will be on the class not on object.

Can have interference between
$t_1 - t_2$, $t_3 - t_4$
but can have →
$t_1 \to t_3$, $t_2 \to t_4$

Wrapper Class →

- As premitive data types are not used as objects in Java.
- We convert int, char, double, boolean to
Integer, Character, Double, Boolean
so wraper wraps primitive data type into object.

int i = 5 ← Premitive

Integer i = new Integer(2) → Boxing
Integer i = 2 → Auto boxing

int j = i.intValue() → Unboxing
int j = i → Auto unboxing

wrapper class

JUnit →

- Open source testing frame work for java.
- Automated unit testing.

Advantages →

- Open source
- Easy to use
- Can test single class at a time.

Anotation →

- New feature added in J2SE.
- Used to add Meta data.

Types →

1. @ Override – check if function is overriden
2. @ Deprecated – to mark a method obsolete.
3. @ Suppress warnings – to suppress the compiler.

JDBC →

Java Data Base Connectivity is an Application Programing Interface used for connecting Database with Java.

JDBC uses JDBC driver to connect with the data base.

Types of JDBC drivers →

① JDBC - ODBC

② Native

③ Network Protocol

④ Thin driver

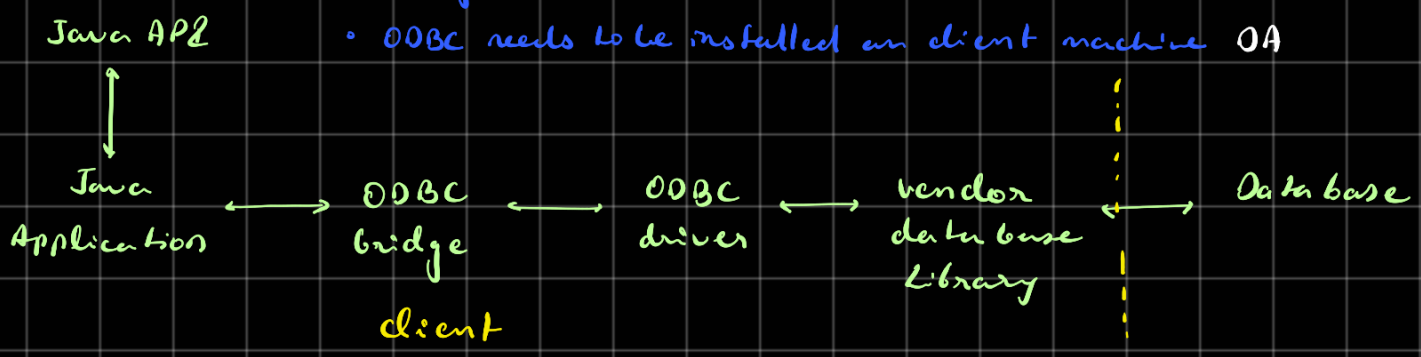
JDBC ODBC

- Written in Java
- Platform Independent
- Object Oriented
- Slower

- In C
- Platform dependent
- Procedural
- Faster

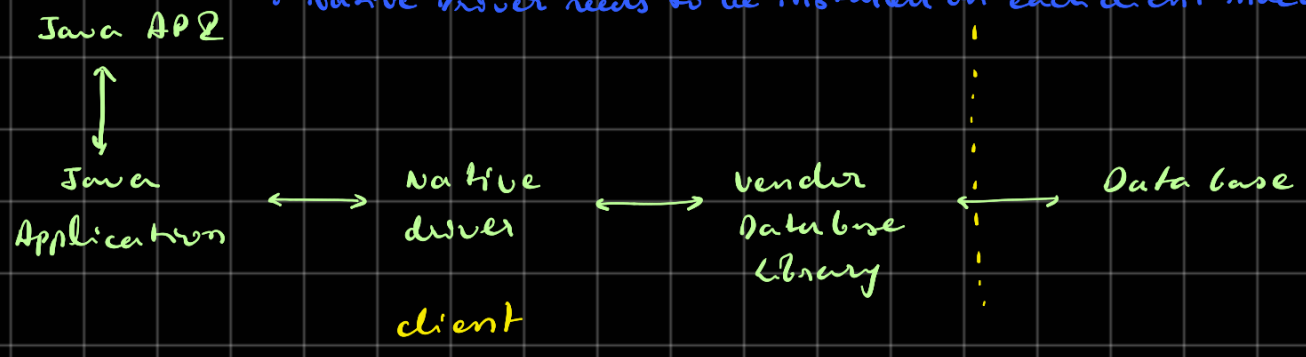
① JDBC-ODBC

- easy to use AD
- ODBC needs to be installed on client machine OA



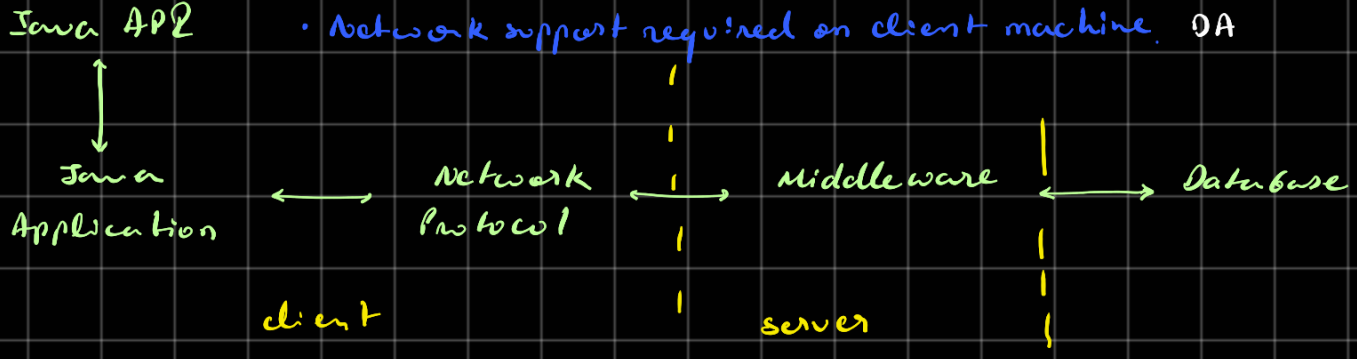
② Native

- Performance upgraded than JDBC-ODBC AD
- Native Driver needs to be installed on each client machine OA



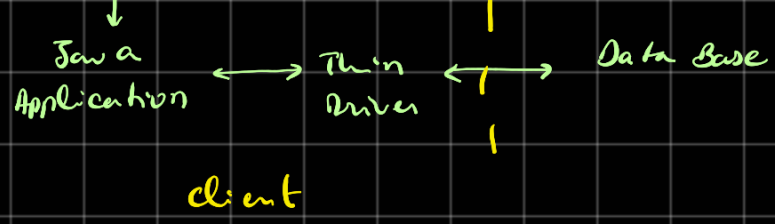
③ Network Protocol

- No client side library is required. AD
- Network support required on client machine. OA



④ Thin

- No software require on any side. AD
- Driver depend on database. OA



JDBC connection ⟶

- Register driver — forName( ) — Class.forName("oracle.jdbc.driver.OracleDriver");
- Get connection — getConnection()

Connection con = DriverManager.getConnection("jdbc.thin.@localhost:__, user, pass");

- Create Statement — createStatement() — Statement stmt = con.CreateStatement();
- Execute Query — executeQuery() — ResultSet rs = stmt.executeQuery("___");
- Close connection — close() — con.close();

Connectivity with MySQL ⟶

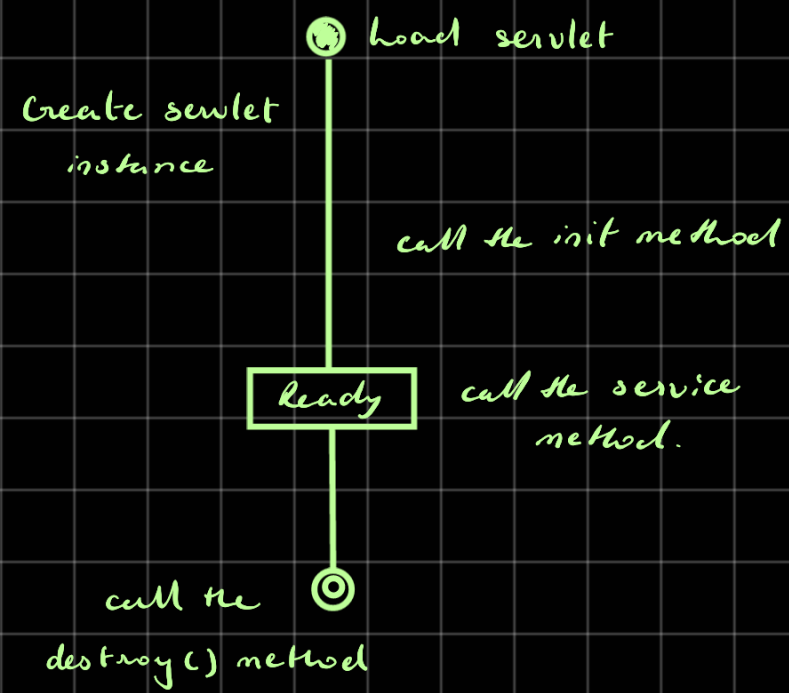
```
import java.sql.*;
class MySql {
public static void main (String args[]) {
try {
Class.forName(". ___ ");
Connection con = DataManager.getConnection ("jdbc.mysql://localhost.",
                                            "username", "password");
Statement stmt = con.createStatement();
Result rst = stmt.executeQuery ("select * from emp");

while (rst.next())
System.out.println (rst.getInt(1) + " " + rst.getInt(n));
con.close();
catch() {
        }
```

Dual Table ⟶ (can be accessed by any one)

- 1 row, 1 column table present by default in oracle database.
- Table has single VARCHAR column called DUMMY that has a value of 'X'.

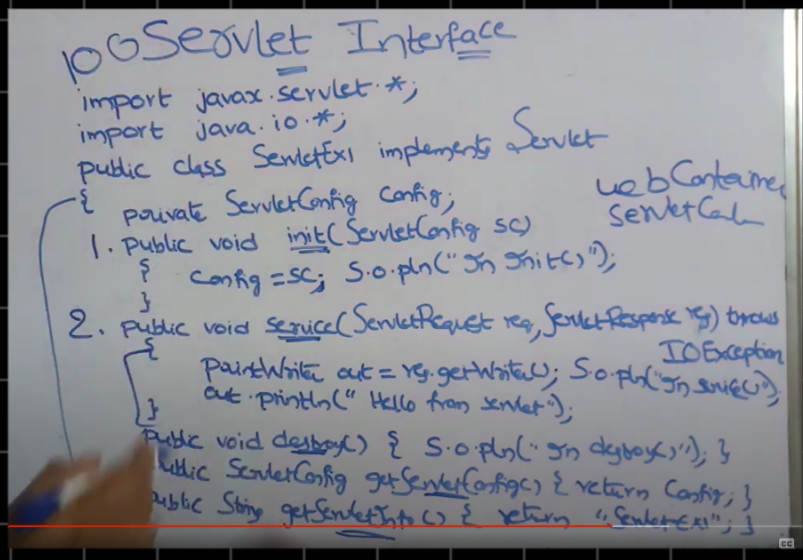
MySQL – Dual Table ✗
but can create one.

Life Cycle of a Servlet →



Methods in Servlet Interface →

- init()
- service()
- destroy()
- getServletConfig()
- getServletInfo()



Servlet life cycle Method →

① init ② service ③ Destroy.

① →

```
public void init(ServletConfig sc) { }
```

② →

```
public void service(ServletRequest rq, ServletResponse rs) throws ServletException { }
```

③ →

```
public void destroy() { }
```

Generic Servlet Class →

Implements servlet, servletConfig and serializable interfaces.

- It is protocol independent

Supports → Http, smtp, ftp

Does not → Cookies, Url rewriting, session tracking

Methods of GenericServlet class →

Servlet Interface

1. public void init(ServletConfig config)
2. public void service(ServletRequest req, ServletResponce res)
3. public void destroy()
4. public ServletConfig getServletConfig()
5. public String getServletInfo()

↓ can only be done by extending user defined Interface.

ServletConfig Interface

1. public String getInitParameter(name)
2. public enum getInitParameterNames()
3. public ServiceContent getServletContent()
   " " " " config()

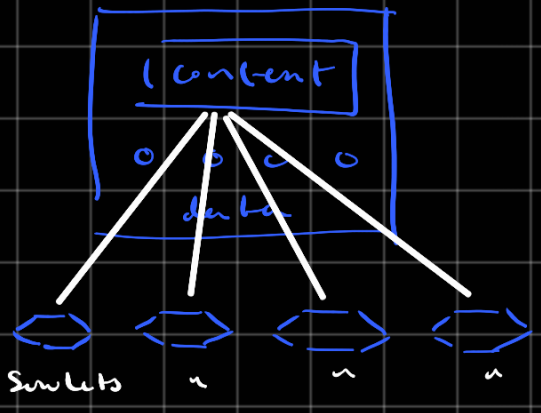
Generic Servlet class	Http Servlet class
1. javax.Servlet	1. javax.service.Http
2. Servlet Interface	2. extends GenericServlet
3. service() ← to override	3. ——— ∝ ——— ∝ ———
4. init(), destroy(), service(), getServletInfo(), getServletConfig(), initparameter(), getInitParaNames(), getservletcontent().	4. All of Generic + { doget(), dopost(), doput() } → in place of { service() }
5. protocol indipendent Implem	5. protocol dependent Implementation.
6. Any type of request	6. Http only request.
7. Can not implement all methods of Http.	7. ———

Http Servlets ⟶

Methods of Http Servlets ⟶

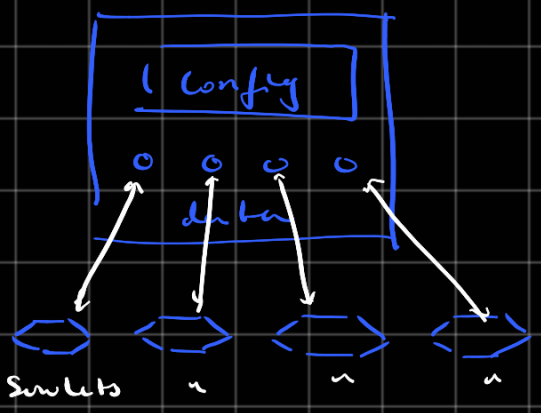
① public void init (ServletConfig sc)
② public void service (HTTPServletRequest req, HTTPServletResponce res)
③ public void doget (HTTPServletRequest req, HTTPServletResponce res)
④ public void dopost (HTTPServletRequest req, HTTPServletResponce res)
⑤ public void destroy ()
⑥ public String getInitParameter (name)
⑦ public Enum getInitParName ()
⑧ public ServletContext getServletContext ()

ServletContext

1. Method ⟶
   getServletContext()
2. 

   data available for all servlets.
3. Singleton Method
4. Start up at servlet(creation)
5. Destroy at server shutdown

ServletConfig

1. getServletConfig()
2. 

   data available for per servlet.
3. Non-Singleton Method
4. Creation at Init phase
5. Destroy() method.

Servlet with JDBC ⟶

```
public class DBConfigParamServlet extends HttpServlet {
    Connection con;
    PreparedStatement st;
    Statement stmt;
    ResultSet rs;
    ServletConfig config;
```

```
ServletConfig config;
public void init() {
    config = getServletConfig(); //Returns this servlet's ServletConfig object
    String driver = config.getInitParameter("driverName");
    String url = config.getInitParameter("urlName");
    try {
        Class.forName(driver);
        con = DriverManager.getConnection(url, "scott", "tiger");
        System.out.println("Connected by using init parameters..");
    } catch (Exception e) {    System.out.println("Error in connection..");  }
}
  ……doGet()…{} }
```

Servlet Chaining →

. for "Forwarding" or "Including" request from one to another servlet.

• RequestDispatcher interface →

- Requ.Dispatcher.forward(request, response)
- Requ.Dispatcher.include(request, response)

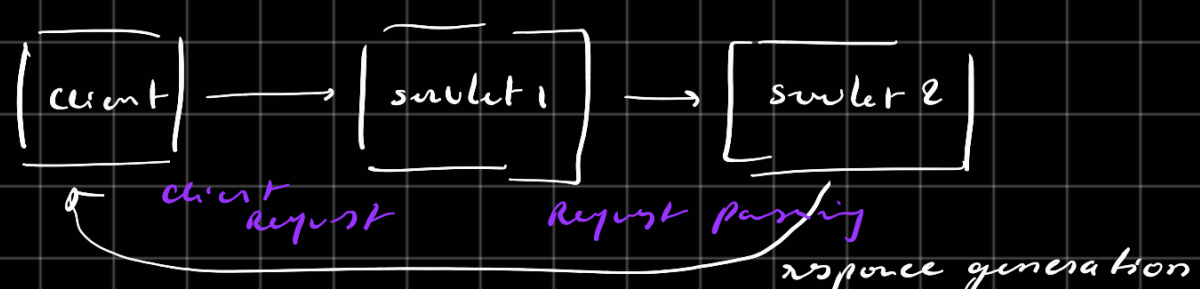
eg →

ServletContext sc = getServletContext();

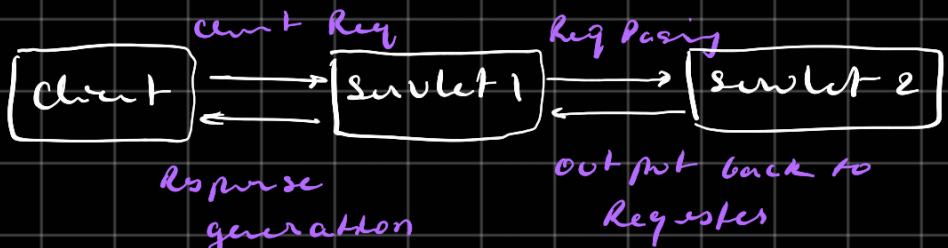
RequestDispatcher

rds = sc.getRequestDispatcher();

rds.forward(request, response);



rds.include(request, response);



JSP → Java Servlet Package

Advantages over servlet →

• Extension of servlet

- Easy to maintain
- No need to configure
- No need to recompile and redeploy
- Less code than servlets.

JSP Scripting elements →

1. Scriptlet tag →

contains any no. of java statements, methods, variables, declaration and expression.

<% code %>

2. Expression →

<%= expression %>

→ value recieved is converted to string.

3. Declaration →

<%! Declaration %>

declares one or more methods.

Implicit Objects JSP →

1. request	Servlet / HTTP Request
2. response	Servlet / HTTP Response
3. Config	servlet Config
4. content / applictes	servlet content
5. Page	→ in web pages.
6. page content	PageContent class
7. Out	JSP writer
8. Session	HTTP Session
9. exception	Throwable

JSP Directive Elements →

1. page 2. include 3. taglib

Syntax →

<%@ directive attribute = "value" %>

1. page →

<%@ page import = "java.sql.*" %>

2. include →

<%@ include file = "student.html" %>

3. taglib → JSTL (JSP Standard tag Library)

<%@ taglib prefix = "prefix of tag library"
uri = "uri of tag library" %>

JSTL → JSP Standard Tag Library

- Fast
- Code reusable
- No need to scriplet tag

JSTL tags →

core → variable support https://.../jsp/jstl/core

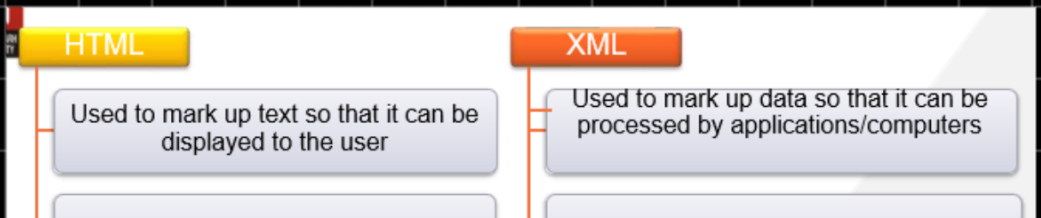
function → string manipulation and length https://.../jsp/jstl/functions

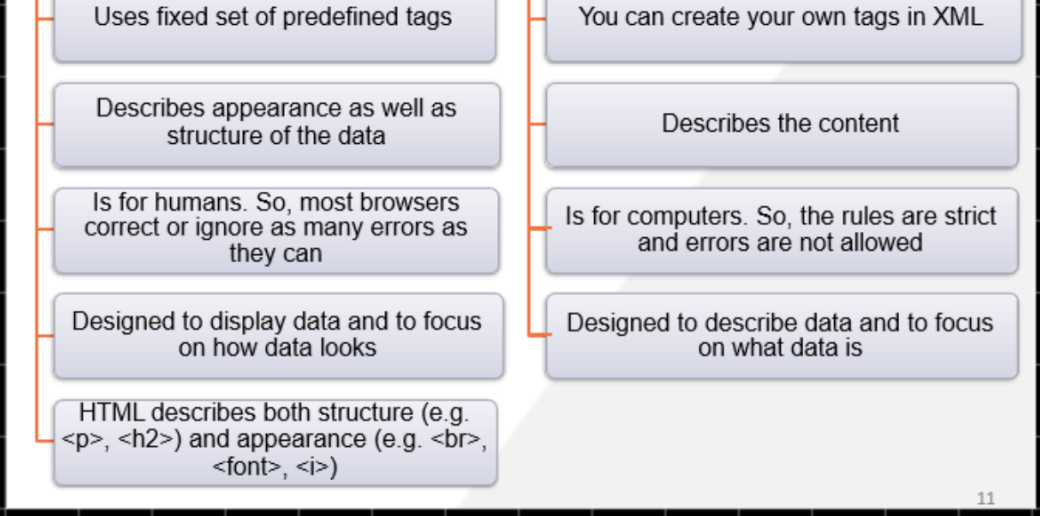
formating → message, number, date formatting. https://.../.../fmt

XML → provides flow control, transformation https://.../.../xml

SQL → SQL support http://.../jsp/jstl/sql

XML - Extensible Markup Language





Usage →

- Seperates data HTML
- Stores data
- Share data
- Exchange data

Syntax Rules →

- case sensitive
- One root node
- closing tag
- Value in " " or ' '
- Elements should be nested.
- Comments <!-- Hello -->

DTD - Document Type Validation

- To validate XML.
- Defines what tags are keeped in XML document.
- Provides grammer for one or more XML documents.

syntax → <!DOCTYPE>

Alternatives →

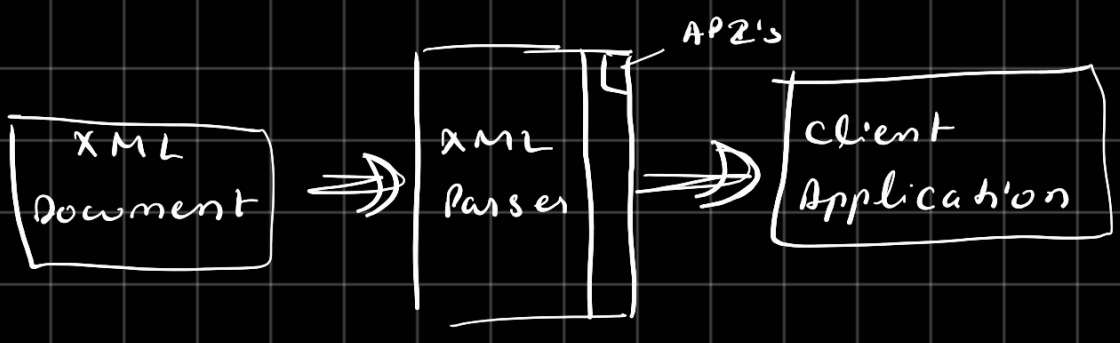
- Schema
- Relax

Internal DTD ⟶ Internal subset
<!Doctype> inside XML document

External DTD ⟶ External subset
<!Doctype> outside XML document

<!ELEMENT library ANY>
<! " book (#PCDATA)>

Parser ⟶



XML parser is a package that provides interface for client Application to work with XML document.

Types of XML Parser ⟶
① DOM Document Object Model
② SAX

DOM ⟶
- Provides a standard way for accessing and Manipulating XML document.
- Follows W3C standard.
- Composed like a tree structure.
- Supports both read and write operations.

- Memory inefficient
- slower than other parsers

SAX

- Doesn't create any internal structure.
- Memory efficient.
- Event-based parser

- Broken into pieces.

Rest →

- Used for deploy application that can be accessed over the network.

Annotation →

- @GET, @PUT, @POST

RMI → Remote Method Invocation

- It's an API used to create distributed application in Java.
- Provides Remote connection between applications → ① stub and ② skeleton.