DATE 04/01
PAGE No.

Web Based Java Programming (Java E-E)

Java 5 Feature

→ Accessing static methods & variable

→ Method over ridding & additional support
- Override
- Convariant return (allow to use a subclass r.t. while overriding)

→ Method overloading
- Autoboxing & unboxing
- var-arg parameter method

→ Collection programing enhancement
- For each loop
- Generics

→ New Style Class
- Enum
- Annocation

→ C Style enhancement
- C style Formatting I/P & O/P
- Scanner class

→ Collection
- Queue type collection
- java.lang.concurrent package

DATE
PAGE No.

Java 6

→ primarily to improve performance & overcome the lags in Java
→ class introduce
    Console
    NavigableSet
    NavigableMap

Java 7 (Dolphin)

→ Execution flow change with respective to main method.
→ underscore in number literal is allowed.
→ switch allow
→ Try with resources
→ bug fixing for var-args in overloading

Java 8 (No codename from this version)

→ Interface issue resolved
    • default method support
      static method in interface
→ Functional Programming
    • Functional interface
    • @ Functional Interface
    • lambda expression
    • Method & constructor reference (::)
→ Stream API (bulk operation on collection)

→ New date & time API solve issue of thread safety
→ Multi core processing
    Base 64 API
→ Annotation enhancement

PAGE No.

Lambda Expression

→ It is in anonymous function that does not have
- a name
- return type
- A modifier

→ Benefits
- enable F(x)onal programming
  write more readable, reduce code
  using API easily
  to enable parallel processing

→ easy to use

```
(int num1, int num2) ->
{
    return num1 + num2;
}
```
{ lambda expression }

```
(num1, num2) ->
    num1 + num2;
```
{ Simplified lambda expression }

→ If lambda expression contain only 1 statement then
- the braces { } are optional
- the return statement is optional
- the return statement and argument type is optional if the it contains only 1 statement
- if the function is taking only one argument you can skip the () eg.

DATE 05/01
PAGE No.

Java Technology

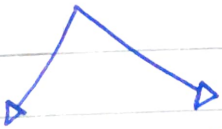
Java SE → (JDK + JRE)

Java EE → (Enterprise Edition) : Distributed technologies

(J2ee)

(a) Servlet
(b) JSP
(c) Webservices
(d) JPA
(e) JNDI
(f) JMS
(g) EJB
(h) JSF

Java EE profile



Web SDK
a, b, c, e, h.

Enterprise SDK
d, f, g + including web

Java ~~Datab~~ Database Connectivity

Driver specification

Type 1 : JDBC to ODBC Bridge driver
This is the only driver which was comming along with JRE. All JDBC calls were mapping to ODBC call which were then mapped to Database Driver Calls ultimate connected to database

(deprecated)

drop (i) performance
(ii)

Type 2 : Partly Java & Partly Native code driver (some code which is platform dependent which used in order for the driver to function, to improve performance and efficiency) i.e. concept of JNI (Java Native Interface).

This driver requires the client Interface ~~sout~~ software of the database to be installed on the client machine inorder for the drive to be used. (OCI - Oracle Client Interface).

Type 3: 100% Java based driver that uses a standard network protocal to communicate with the underlying database.
Effencicy is very low.

Type 4: 100% Java based driver but it uses PROPRIETORY PROTOCAL to communicate with the database.

* Type 3 & Type 4 drivers are also called THIN drivers, because they DO not FORCE you to install the client interface of the database.

Type 2 → server machine
most commly : Type 4. (mysql)

Connection :- to represent connection dataapp to db.
Statement :- to fire query.
resultSet :-

3 step

STEP I :- load

Step II :- Open db connection

step III :- create a statement

Step 4 : Fire the query.

Step 5 : iterate through the resultset.

Step 6 : close the resource.

06/01

JDBC is a set of specification that java to enable java application to connect, retrieve, manuplate data from db.

Since JDBC is a specification java declared what functionality should be performed and does not define how this functionality has to be implemented.

Every database ~~in~~ wonder must decide has to which driver it would support and provide an implementation for the respective drive.
JDBC specifies following 4 type of driver:

(1) Type 1 drive i.e. JDBC to ODBC that enable java application to connect to any database or change databases on the fly. (dynamically). In this driver all JDBC calls are mapped to ODBC which intern are mapped to the database driver calls and

finally communicated to database. This is a slow performing driver and hence it is a least use also it has been remove from Java 8 specification.

(ii) Type II driver

It is a partly java & partly native code driver, that use JNI (Java Native Interface) calls to use platform specific database drivers. This greatly improve the performance & efficiency & hence it is most perfer driver on the server set. In order to be able to use this driver the client software of the database known as the instant client or client interface must be installed on the machine using this driver.

(iii) Type III drivers

The ~~type~~-III driver is 100% java driver that uses network based protocol (IPS & SPX) or network ~~not~~ in order to communicate with the underline database. Because this driver uses a network standard protocal it is least/slow performing drivers and hence usally not prefer.

IV) Type IV driver

It is a 100% java based driver that use a proprietary (user define) protocol to communicate with the underline database. It is the most commonly use drive & hence it a light weighe and can be bundle along with the java application.

The vender of the database need to decide and provide respected driver to be the programer. For instance Oracle provide type II & type IV driver whereas MySQL provide only type IV drivers.

The type III & type IV driver are known as thin drivers.

07/01

XML

does not have any tag of its own.

2 type
- well formed
- valid

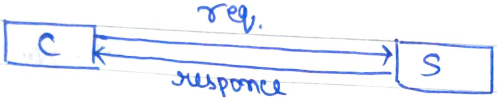
DTD / Xschema

Parser (validate)

DATE
PAGE No.

Servulet

client, server, protocal
(Http, Ftp, Smtp, Pop, ...)



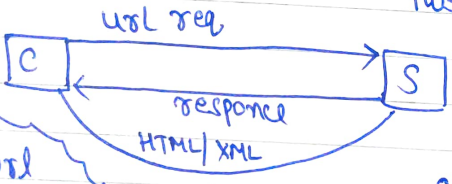
web server

after 50 yr.. same o/p

07/01/2022
12:04 PM

There are the static page, the ~~data~~ data remain same.

→ Server side programing



req url
valid url
if yes then process
else send 404 error

code are in markup lang. may be HTML or XML.

Current date & time
⇓
whenever we req server.

procedure

Tws : Technology enable web server

{Java, PHP, .NET}

Pre requisites for Servlets

01. JDK
02. Java EE web profile SDK
03. Java enable web server

(tomcat.apache.org)

Set-up of tomcat

HTTP/ connector Port 80

User name admin
pass-word admin

C:\Tomcat9

{Cataline_Home = C:\Tomcat9}

localhost / 127.0.0.1 / ip add of machine

→ Web Application

create project & java class with package.

inherited from HttpServlet (javax.servlet.with.HttpServ

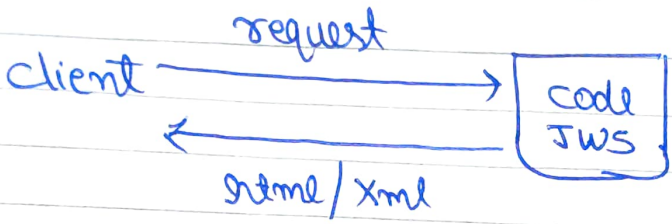
xml file (web.xml)

(webapp 3.0 ssd)
(prefix - elemenate)

DATE
PAGE No.

serv ⇒ server
let ⇒ script

It is a peice of java code that is exe. on servle side in java enable server (web-server).



01 Tomcat
02 JRun
03 webSphere
04 weblogic

diff step for different server.
{no standalizter}

Java come with WAR
Web application Archive .war

War
- html, jpg, gif, ttf. (root level
- WEB-INF (private folder) ← (connect to us only)
  - classes
    - .class file & packages → POJO, POJI servlet ...
  - lib (optional)
    - .jar file
  - web.xml (deployment description)

WEB-INF

1. classes
2. lib
3. now create xml file (web.xml)

webapp 3.0ssd
prefix - element

4. source
5. copy
6. cmd jar

jar of <filename> WEB-INF

jar tf FirstWeb.war

7. localhost - manager/html

war file to deploy

8. localhost/FirstWeb/index.html
9. localhost/GetTime

DATE
PAGE No.

Step to create a Servlet

STEP I : Create a java project & to it build path add the javax.servlet-api.jar located in the JavaEE home\glassfish\module folder.

STEP II : Create a java class that extends the javax.servlet.http.HttpServlet. Override the doPost method with the respective business logic.

Step III : Under your project folder, create a folder with name WEB-INF, Create the folder 'classes' and lib under this.

Step IV : Create an XML file by the name 'web.xml' in the created WEB-INF folder. The web.xml file will be based on the webapp 3.0.xsd
Also remove the javaEE prefix

The web.xml also known as deployment discriptor which contain an entry for each servlet and its corresponding mapping.

Packaging your web-application

Step I Copy all the class files along with its package & paste it in WEB-INF classes folder

On the cmd, issue (use) the jar command as :-

jar cf <Name of WAR file> [contents of ROOT] WEB-INF

Deploying the WAR file in Tomcat

Step I : Launch the tomcat server by launching tomcat9.exe from the catalinaHome\bin

Step II : Open the browser & type localhost and click on the link manager app.

Step III : chose the section WAR file to upload (deploy) and select WAR file created by you and then click on deploy.

Step IV : To conform your app is succesfull deployed, located your application the application section and ensure the running state is true.

Invoking the

Step I : Open the browser instent & type http://localhost/<Name of WAR file without extension>/<URL pattern>

http://localhost/FirstWeb/GetTime

DATE 08/01
PAGE No.

Web → dynamic web project

Second Web Application
click on runtime (first time)
→ Apache
→ Apache Tomcat v 9.0
Tomcat folder
dynamic web module (3.0 / 3.1)

generate XML file
finish

Src/main/java (ctrl+N) servlet java class inherited

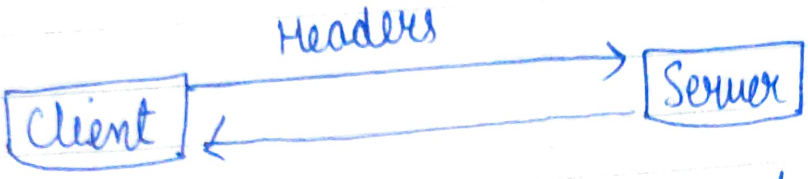
Run (ctrl+F11)

Error → Server → tomcat server → tomcat admin port (8000).

(Alt+Shift+S)
↓
for override

DATE
PAGE No.

HTTP



accept-lang = en-IN

internalization
localization.

en-US
en-UK
gu-IN

(lang, country code)

10/01

Registering Servlet
→ Entry in web.xml
OR
web @Servlet (Annotation)

Step I — create a web project

(II) src/main/java
↳ web
↳ Servlet

(III) next, packagename, classname

(IV) url mapping → (we can change urlname)

(V) Select method

action in HTML file, never start with "/".

Drawback of get

01. Data is visible in the URL
02. Data size/length is limited to 2048 characters.
03. Only text info. can be transmitted

Adv of Post

01. No data visible in the URL.
02. Amount of data will be unlimited.
03. Any kind of data can be transmitted.

Copy jar file in lib folder - to connect db

Web-app web inf lib

→ Servlet API

implement Interface javax.servlet.Servlet

5 method

public void init ()
public void service ()
public void destroy ()

String getServletInfo ()

~~Str~~ ServletConfig getServletConfig ()

Stage I deployment : provide war file

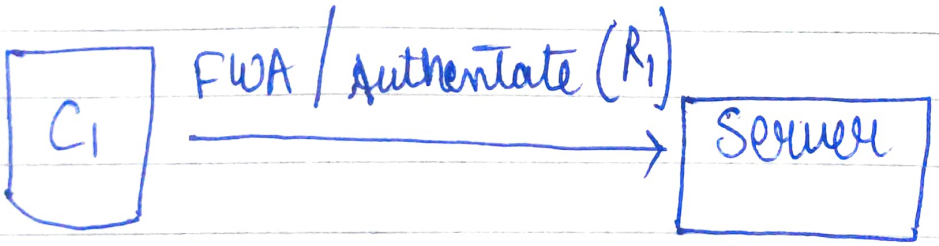
JWS → activates

(i) check is the structure of war file
(ii) web.xml

Create contextTable / first web app

Name	Class	M.A.	add.info	URL
Auth	fi.servlets.Authentication	0	-	/authenticate
Category	fi.servlets.Category	0	-	/Category

agar xml file nhi hai... toh us war file me jaake hr .class me @WebServlet (annotation) search krega.



(i) load the class servlet
(ii) instantiate an obj of servlet
(iii) injection
(iv) on ~~servlet~~ servlet obj it call init method
(v) update the M.A. in ctr table
(vi) fire the servlet method

check the method & fire.

when new client send req, they first check entry in context table, check M.D. & skip step & then direct jump to step III.

In-short, he skip first five steps.

& if we restart the server, we have to perform all step.

- Destory the object
  - web application : stop / reload / restart
  - web server : stop / restart
  - threshold time period : time out.

Life-cycle

The servlet technology define various methods as the part of specification that describe how a servlet is instruated, process the request of client & finally its destroyed. These specifications are layed by java EE & have to be follow both by the programmer as well as server in which they are deployed.

The servlet API provide you with a predefine interface called the javax.servlet.Servlet

The lifecycle method inside interface are

(i) init
(ii) service
(iii) destory

The init method it called only one in lifecycle of the method where the programmer can write any initialization code such as connecting to the database.

The service method is called each time a client send a req to the URL mapped to this servlet, this is where we write our business logic which is were to write & generate the corresponding response.

The destory method is fire only once in the lifecycle of servlet which will occur in one of the follow 3 situations

(a) the web-application is stop or reloaded OR
(b) the web server is stop or restarted OR
(c) the thresold timeout period for the servlet lapse, this will happens whenever the time b/w two consecutive request for the same servlet is more than the timeout period.

→ The deployer module of web server

When a war file is given to a web server to be deployed the deployer module of the web server validate this war file structure and if found correct create a context table for war file either based on web.xml file or by scaning all the classes and checking for web servlet annotation or uses the combination of both web.xml & annotation. Based on the info retrieved create the following table

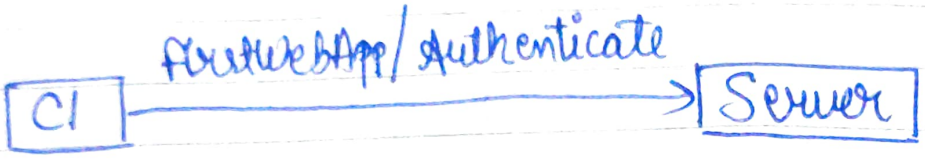
Context table/first web App

Name	Classes	M.A.	Addinfo	URL
Auth	fi.servlets.Authenticate	8000	-	/Authenticate
Category	fi.servlet.category	0	-	/Category

In the above table name represent the unique name to identified the servel with in the servlet, the class represented the cname of the class, the memory address into 0 identified the memory add of the obj i.e. to created to process the servlet request, the additional info is the specific parameter that the programmer can send to the servlet & the URL represent identified the URL that the client could use to request the servlet.

Exploring the I.C. 4

when a client send a request to a server by entering the URL the following stages occur



In the above diagram, when a client make a req for FirstWebApp/Authenticate the server check if a context table by the name "First Web App" is available on the server or not. If the context table is located it identified the URL in our case /Authenticate. If the URL is not found in the context table the error code 404 URL not found will be return. If the URL is located then it validates the memory address. If the memory address is zero then the fallow step occurs:

(i) load the servlet class as specified at the context table then instantiate an obj of servlet class and fire its constructor.

(ii) Perform the process of injection where which additional info as mentation in the context table is prepopulated with the object.

(iii) On the servlet object, it then calls the init method to initialized the servlet. If the initialization is successfull the entry of this servlet ~~is the~~ object is then updated in the context table.
The above will occur only once when the first time the req for this servlet recived. The servlet is now ready to service the request of client. Hence it go ahead and calles service method which intern internally call the doget and dopush method.
The destoryed method will not be called as it follows a different pattern as mentioned previously.

When a subsequent client make a request for this servl the context table is check for the URL pattern and when the URL pattern is located since the memory address is not equal to zero directly the service method for this servlet is executed.

DATE 11/01
PAGE No.

Cookies

2 type
- Non-persisted : life of the browser
- persisted : beyond the browser

on request of server.

Set Max (method)
- -1 : browser based, non-persisted
- 0 : destroy
- >0 : seconds

Session

01. Submitting data Post
02. URL
03. Session
04. Request
05. Application

The li

{session life cycle of servlet}

DATE 12/01
PAGE No.

2 way to send URL
(i) request redirect
(ii) Request dispatcher
  → include
  → forward

post
Server
request redirect
req dispatcher
servlet content

xml : web app 3.0 , edit prefix

13th Jan

drawback in servlet
(i) HTML o/p in java buss. logic
(ii) Sb kaam apan ke ko ek hain
JSP → Java Servlet Page

Java + HTML (JSP File)

all Jsp ages are put in Webapp folder

<% → Service method
<%= → for o/p

path → { workspace/ .metadata/.plugin/org.eclipse.wst.server.core/
tmp0/ work/ Catalina/ ..... }

DATE
PAGE No.

Session 1 - Language Syntax

01. Directive <%@ %>
it is an instr. passing a JSP

a. page → current document (10-12 attributes)
b. include → statically include a pg into the current page
c. taglib → used to use other pre-define tag lib or user define tag libraries.

02. Declaration <%! %>
to define functions or variable that are avail on the current page.

03. Scriptlet <% %>
a block of java code that is inserted in b/w the presentation, it is can be of multiple line & each statement should terminated with semi-colon ;

You can have multipal scriplets with in a JSP pg as long as they are not nested

04. Expression <%= %>
An expression is a single statement that always generate an o/p, non-void value return, The statement of an expression is passed as a parameter to the out.print method.

Session 2 → by default available in Scriptlet or Expression

DATE
PAGE No.

05. Expression language ${ [object] }

All five of them are not in use able.

Session-2 Implicit Object in JSP (9 obj)

01.	request	HttpServletRequest
02.	response	HttpServletRequest
03.	session	HttpSession
04.	config	ServletConfig
05.	application	ServletContext
06.	out	JspWriter (buffered version of PrintWriter)
07.	page	Object (this)
08.	exception	Throwable
09.	pageContext	wrapper obj which hold the reference of other object.

Session 3 JSP tag library (predefine tag)

01. <jsp:useBean>

example:-
```
<jsp:useBean class="fi.beans.ShoppingCart" name="objCart"/>
<jsp:getProperty
```

It allows you to instantiate or obtaining an existance POJO obj

<jsp:setProperty> set the value of property of a bean/POJO

03. <jsp:getProperty> get the value of property of a bean/POJO

04. <jsp:forward> Forwards the req. to another URL

05. <jsp:include> Dynamically include the contents of another URL

06. <jsp:plugIn> flash, apple, excel, chart, pfd

14th Jan

ORM Object Relational Mapping

Save → insert
load → select
Setter → update
remove → delete

→ we don't need to repeat code

what we need

① POJO
② 2 XML file
   - configuration : how to connect to the database, (driver, connection, username, password)
   - mapping : Pojo → table (related & store)
③ JAR file
④ use the cfg to persist data (business logic) [wiring]

DATE
PAGE No.

Hibernate

Step I: create a java project & to build path add all files from the hibernet required folder, also add mysql connect.jar

Step II: In the src folder create an XML file based on the hibernet configuration DDT or schema. This XML file will contain all the information that is required to connect to the underline database. The configuration file also contains the entries of all the classes that are going be manage by the hybernet frame.

Step III: create an XML file based on the hibernate mapping ddt/schema This file will contain all the classes and their respective mapping which the hibernet framework will used to interact with the database.

Step IV: Create a POJO class with various properties & behaviours along with a pair of getter & setter for each property that have to be manage by ~~hybernet~~ hibernet. This class should contain default constructor

Step V: In the Entry class, create an object of the configuration and called the configured method To load the configuration XML file, created in step I.

Step VI: On the configuration called the build session factory which will return the session factory based on the configuration file. Now you can open the session

to insert or retrived the data from the underline db.

15th Jan

Maven

Step I: Create a new project Maven Project click on next & from the category chose the internal & select maven 1.2 (quickstart) or alternative under all catalog under filter type Maven-archetype-quickstart

Step II: Click on next & specified the group id & artific id, here artific id is your project name and the combination of group id & artific id forms a group package then click on next

Step III: Open the ~~form~~ ~~at~~ pom.xml under properties under properties add maven.compiler.source and maven.target.source tag.

Step IV: Under depandies add all the depandies that you are req for the current project.

Step V: Right click on the project & under the maven update the project.

HQL → Hibernate Query Language
It based on database structure.

example: Query allUsers = HibernetSession.createQuery("from userProfile");

2 type → namedQueries
→ SQL native queries

17th Jan

Maven web application

18th Jan

Shopping Card → Pojo class

2003: Rod Johnson → framework → loose
↓
Spring

Framework: It is an abstraction in which software provide generic functionality can be selectively changed by user code.

It is universal, reusable software platform used to develop application, products & sol".

It include programs, lib, api, compilers.

They contain key distinguishing feature:

→ Inversion of control
→ Default behavior
→ Extensibility
→ Non-modifiable framework code.

Types

01	Application	framework	CUI or GUI
02	Web application	framework	web
03	Multimedia	framework	Gaana, Spotify

Spring
Open source
It integrate specification from the java EE

Servlet API	WebSocketAPI	Concurrency Utilities
JSON Binding API	Bean Validation	JPA
JMS	JTA	

inversion of control

Aspect oriented Programming (AOP)

Container, BeanFactory, Application Context
- ClasspathXML Application Context
- FileXml XML Application Context
- Xml Web Application context
- Annotation Config Application Context

DATE
PAGE No.

Bean Scope
- Singleton
- Prototype
- Request
- Session
- Global

Auto-wiring : It is used to automatically populate the properties of bean without we having to explicitly provide information.

Type
- ByName
- ByType
- Constructor

dis-advantage
- once specified, cannot be defined as property in xml
- if two prop. are of the same type spring will raise an exception
- wiring is not

DATE
PAGE No.

Spring Jdbc

19th Jan

Step I

Create a meven project based on quick start archetype & add the dependencies for spring Beans, context, core, Java X annotation, mysql connector & spring data.

Step II

Create a pojo class for User, providing it with properties & getter setter

Step III

Create an interface called UserDAO that will contain all the method that you wish to expose to the user for the database interaction

Step IV

Create a class called as UserDAOImpl that will implement UserDAO interface. In this implementation class create datamember of the type JDBC templet & enviornment that will be auto-wired. This class will use the JDBC templet to interact with the database.

Step V

Create a class called userMapper which implement the RowMapper. This class is responsible for converting each row of the resultset as the object of user.

Step VI

Create DB connection. property in the src.main java folder which will contain properties of connection details To mySQL.

VII

Create sql.properties that will contain all the queries used for jdbc

VIII Create the AppConfigurator that defines 3 beans
(a) Jdbc of type data source
(b) Jdbc Templet of type JDBC templet
(c) UserDAO as type userDAO

IX Prepared the main f(x).

Step I Create a meven project and Hibernate core, mySQL connector, Spring Beans, context, Core.

Step II Create a user pojo class that act as hibernate entity, annotating it with entity & table. Define each property in this entity with their respective Column annotations.

Step III Create a userDAO interface that will expose all the methods threw which the user will interact with your Bean

Step IV Create a class called DAOImpl that implements the interface userDAO. In this class have the datamember

of session factory that will be autowired
Each method implemented will used the Hibernate
Session to store or retrived datas on your entity.

Step V Create the appConfiguration that will instance beans
(a) hibernate configuration
(b) Session factory
(c) UserDAO

Step VI Prepare the spring Hibernet.cfg.xml with the Hibernate configuration

Step VII Create the application.properties which contain the name of the configuration file

Step VIII Create the app.java that define the control flow.

20th Jan

MVC Pattern

Model → data
View → presentation
Controller → controller

Create a class called the AppConfigurator which enables the webmvc framework & specifies the package containing all the controllers classes. This configurator class is also responsible for registering a bean that will be used to tell the spring controller where the views are located and how these views have to process.

Create a class that will implement the web application initilizer which is used to register front controller of the Spring framework (uses the Fromtcontrol Pattern). Using this class we will add the dispacher servlet as the registered servlet of the web-application.

Under you web-app folder, prepared user Index.jsp file which will provide the link to the URL as mention in the controler.

Under the webINF floder, Create a floder name of view that will contain all jsp pg, that will the result of the controler.

DATE
PAGE No.

22th Jan

Web Service

→ It is a client-server application or app component for communication.
→ set rule for communication b/w two device over the network.
→ use for interopeable machine to machine communication.

Type
SOAP: Simple Object Access Protocal Representation

SOAP

→ used for accessing a web service using XML
→ based on W3C standards
→ it formed a WSDL (web-server description Language)
→ UDDI (
  · It is a interface.

→ messing Protocal specification
→ Released in June 1985
→ use for exchaning structure info in the implement of web services in computer network.