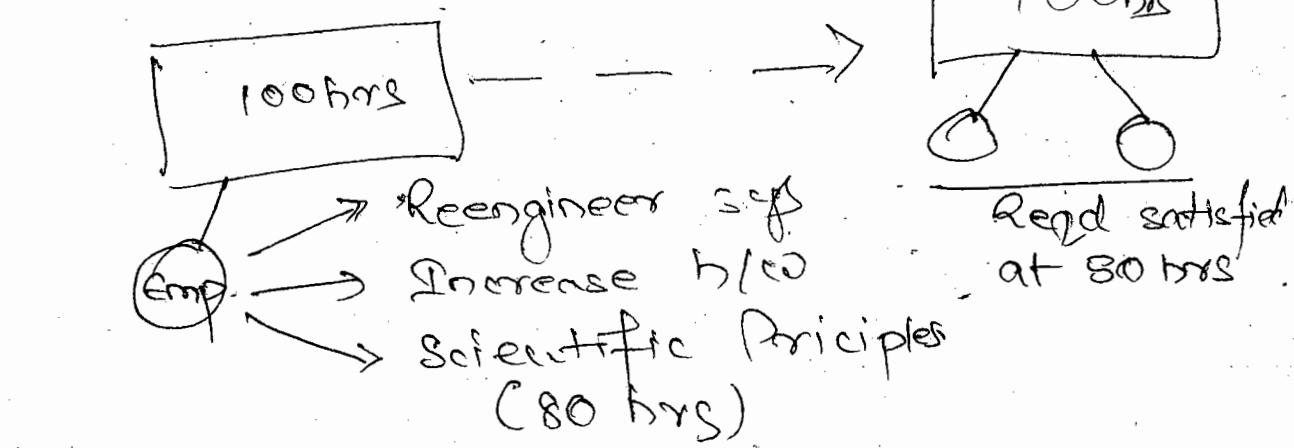
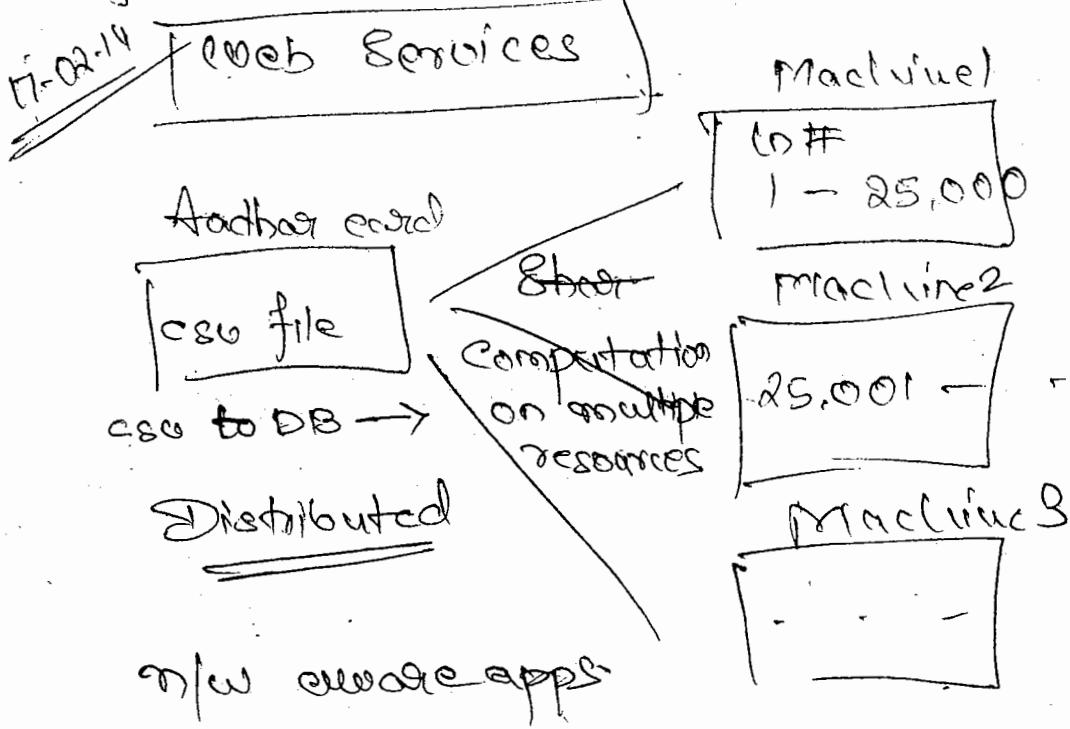
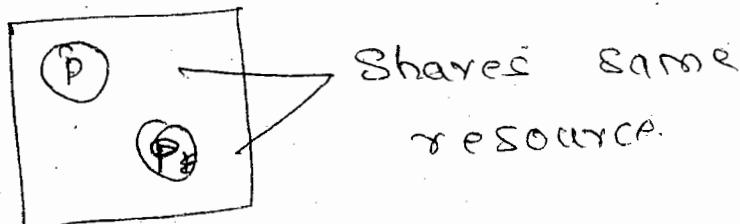


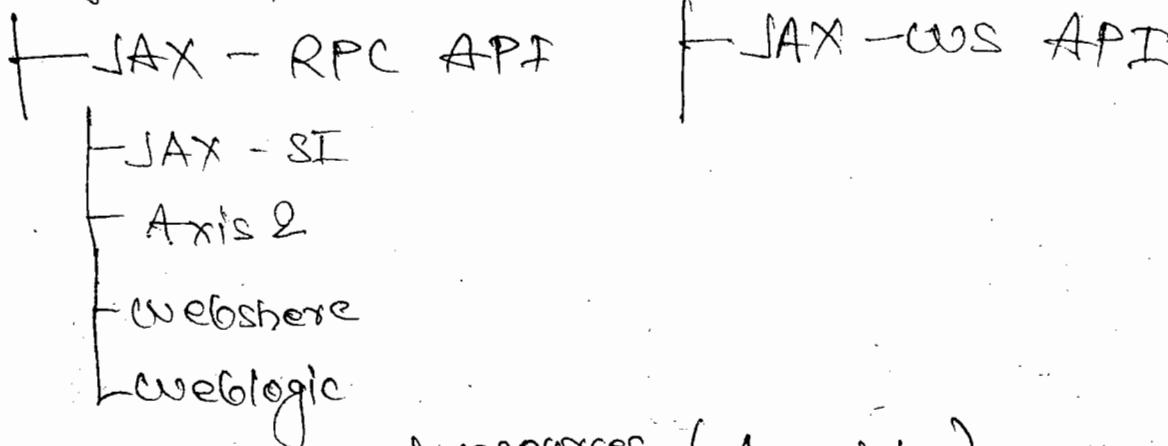
## Project Architecture



- Using IPC we can't build distributed apps.
- niche tool — very rare.

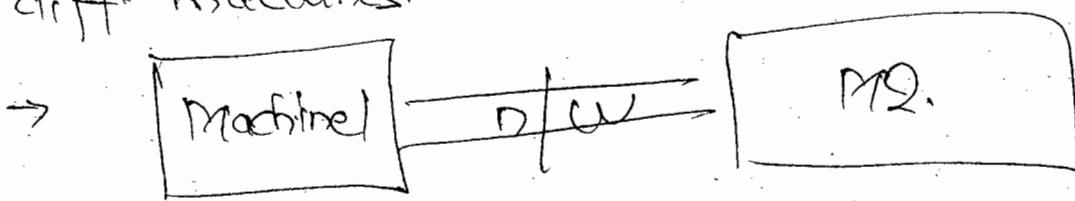


## Types of Web Services



- High utilization of resources. (1 printer)
- Higher throughputs. → High availability of resources.  
(If one fails sick) (shares load)
- Executing diff. instances of same prog<sup>n</sup>.

diff. machines.



- Executing common jobs is independent halves. (N/SW aware apps.)

Eg Shopping Malls → Lucky Draw (data) → Org →

Dataentry op → Inserting into file.

File → DB.

Lakhs of records. → Split ur jobs.

Gr0 for distributed prog.

- Initially there was no distributed prog. tech / concept
- Cloud - Azure API. (Today's tech)  
[Cloud Rain - video youtube]

### Java Support for Distributed Tech

- Java needs to have API's.

- ① CORBA (Deprecated)
  - ② RMI
  - ③ EJB
  - ④ Web Services
- } API's

CORBA : Common Object Ref. Broker Architecture

- Exposes obj over network

→ Interface Def. Language (IDL)  
Language independent way of writing.

→ IDL contains Java class ~~properties~~ structure.

→ CORBA compilers [corbac]  
Generates language specific class

→ corbac takes IDL + lang-type to generate  
lang specific class with structure. (attributes +  
dummy methods)

→ Supports any language

Message oriented architectures | CORBA CONTAINERS

IDL

IDL ( ~~IDL~~ Scripting file)

Corba compiler ← Language info.

Lang. specific obj (structure)

Open class & provide logic.

Place this obj on servers

(MOM's/ Corba container) [Commercial server]

### Drawbacks:

- ① Java developer doesn't start with Java.
- ② IDL is scripting language. (Time to understand)
- ③ Corba compiler we have to work with { Think IS your basic  
it gives source file → we have to write code  
• class file in some compiler generated class.  
Deploy into servers
- ④ Corba Servers are costly.
- ⑤ Need of Corba Admin to pkg & deploy.
- ⑥ New tech. no proven success at that time.
- ⑦ Only supported for Java & C++.

## ~~(2)~~ RMI (Remote Method Invocation)

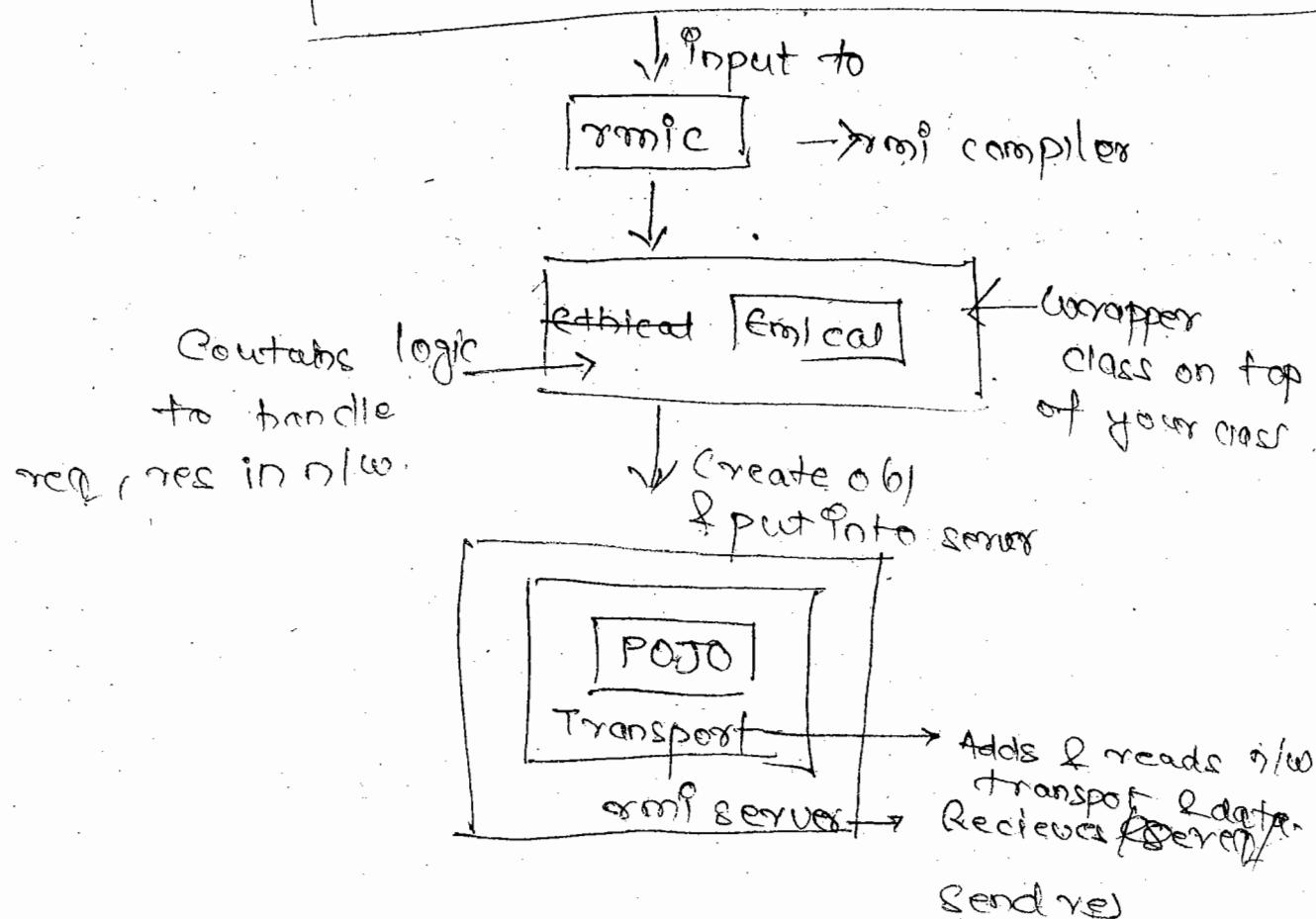
- Exposes obj over n/w as remote obj.
  - NO IDL.
  - RMI we start development in java class only
- Java class has the ability to talk over n/w.  
We don't need to write code ~~to~~ to talk over n/w.

- uses POJO model.

POJO

Eg

```
class Emical {  
    float cal (int p, int n; float) {  
        } // Normal Java class.
```



→ open source provided by Java.

Light weight Map DS

Hash
Name
obj

No Enterprise class level features.

- bind to bind obj with server & lookup to find it.
- Binding: Putting obj in Rmi Server.
- Lookup: Searching obj in server by name.
- RMI servers are light weight, open source.

### ③ EJB

server / runtime

- Managed ~~server~~ (Can pool concept)
- When obj are running in managed container we needn't write any code for managing obj.
- Enterprise class level apps = Distributed apps
- But, we want only list of other people to access our obj exposed over the n/w. ~~for~~

• else if's security breach.

Enterprise Level Apps features

① Security.

② Transaction

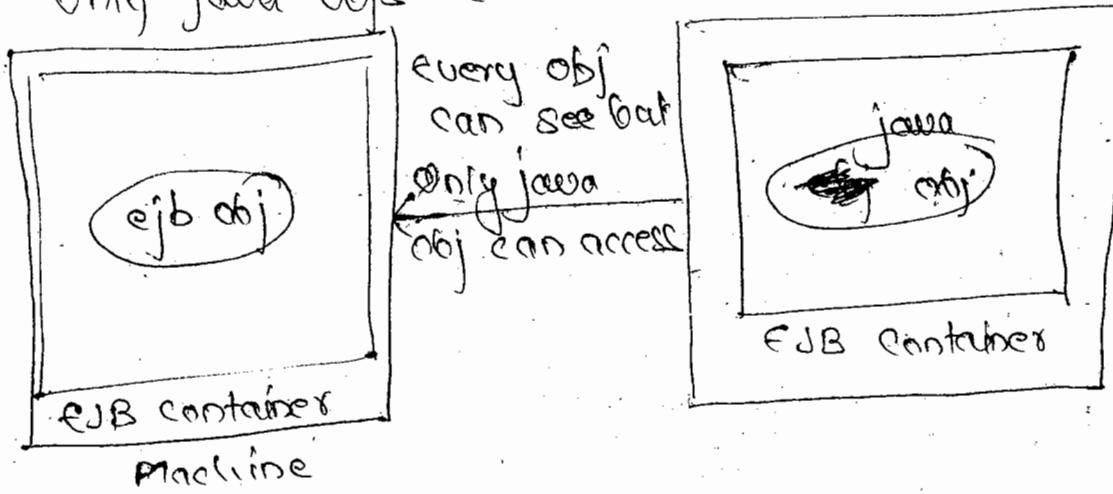
③ Logging.

→ RMI server doesn't provide con pool & enterprise apps. features.

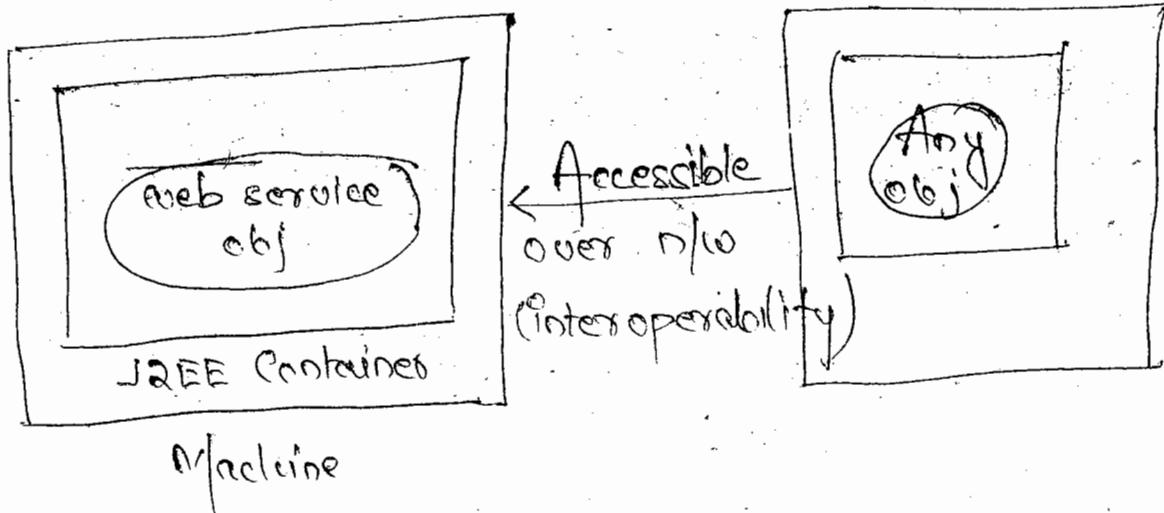
→ Weblogic server internally uses MUX and EJB objs.

### Drawback

→ Only java obj can access.



### Web Services



→ Irrespective of language it's Interoperable.

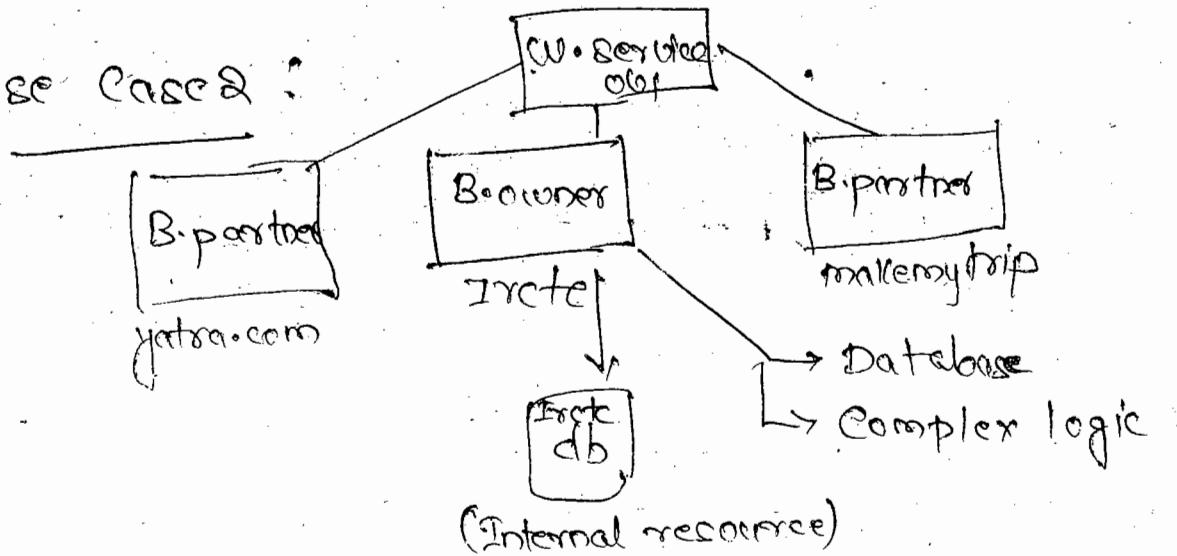
## Idea Behind Web Service (Where to use)

Ex① : Where we use web services.

Use-case : Credit card

- 2 billing cycles.
- Payable ~~is~~ within grace period.
- Else penalty will be attracted.
- Stringent (strict) timelines.
- We have to generate credit card statement in limited hours (like 12am - 8am).
- Same instances of prog. will run in diff. machine.

Use Case 2 :



Prob 1 : How to provide storage for B · partner  
to dB ( blog we can't give  
username & pwd)

Prob 2: B-rules should be imposed to all booking portals & are dynamically changing to reqd.

If B-partner develop rules then irctc has to test those rules first.

Prob 3: If Irctc changes rules then every B-partner has to modify.

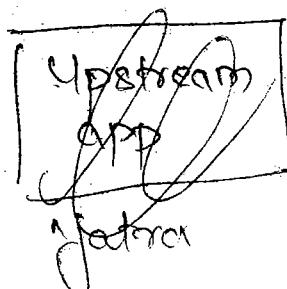
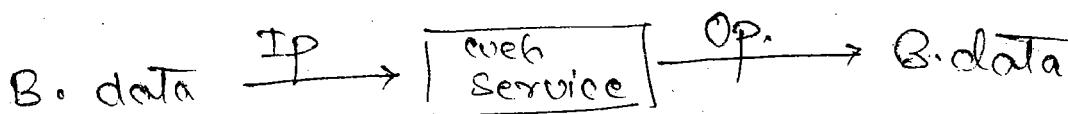
Changing of B-rules can't be easily imposed.

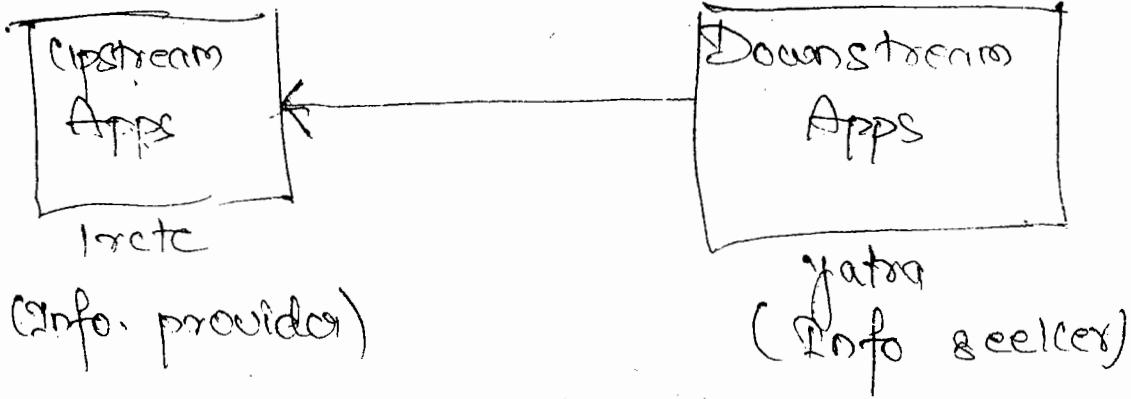
→ Web Service exposes distributed obj over a n/w in an interoperable manner.

→ Web Service is related to Integration tier.

→ It's not web app. (Web Service never gives HTML / JSP data as input or output)

→ Web Service ~~are~~ component <sup>exposes</sup> ~~register~~ classes with complex B-logic over the n/w.





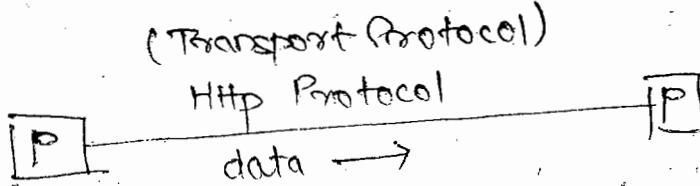
→ Exposing complex B. logic internal to our app.

with external apps in a distributed & interoperable manner is web service.

→ Servlet is web component. It needs data in web presentable format.

→ Web Service accepts data as objs of data & dispatches response as obj of data carrying B logic.

### Web Services Architecture



① Transmission medium.

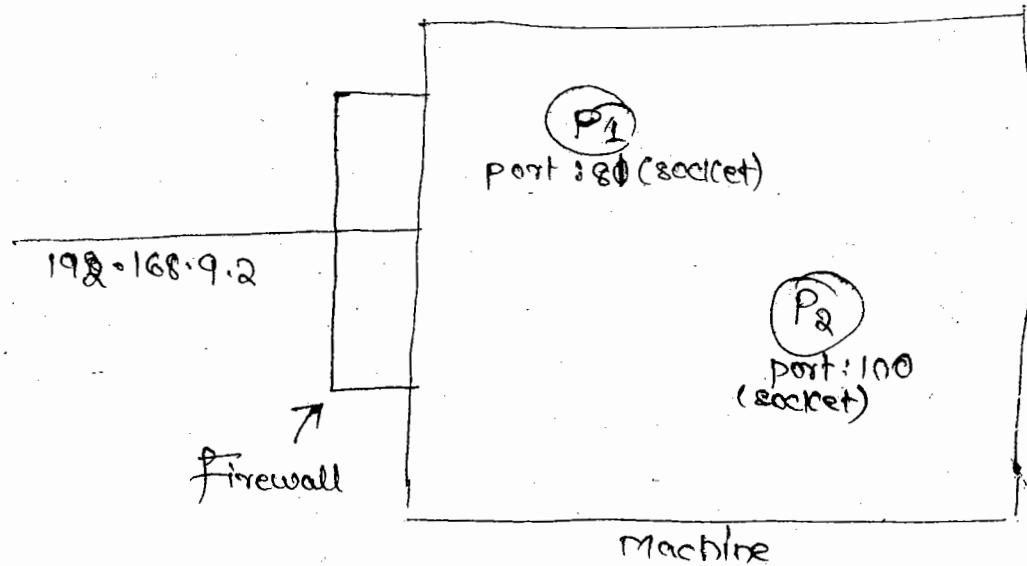
② Protocol. (guards medium for proper comm)

Divides n/w bandwidth to logical channels

Guaranteed delivery from sender → receiver

Current version HTTP 1.1

- Http protocol is firewall friendly. (default port 8080)
- Using physical n/w addr / IP addr. we can comm. to any computer.



- ⇒ The data over the n/w can be recd. by prog. by using socket (port) number.
- ⇒ Firewall intercepts the comm.
- ⇒ 8080 is open port on firewall.  
So, we don't need to cfg explicitly.
- ⇒ Web services gets benefitted from HTTP protocol.
- ⇒ Data format should be same

Lottery ticket when u click it gets installed  
2 open port 2  
leaks data

Character Encoding	binary representation	Character Encoding
ASCII	✓	ASCII
UTF	✗	ASCII

ASCII → No I18N ✗ , utf-8 - I18N ✓

→ Every info can't be sent as single character. So we frame the char.

→ Most oftenly we use "CSV" format.

[1, raja, 24, Birth, Bhilai, 17/07/89]

↓ misinterpreted the info

No semantics / structur.

Data format

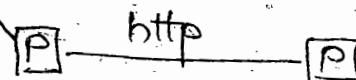
Semantics: Info abt. Info

→ So, we shouldn't use CSV.

\* xls, → Only MS can understand.  
pdf, word } - Not interoperable

XML

→ Use XML representation.



XML → Extensible Markup Language (ISO) W3C

spec

→ XML is used for representing data.

<operation>

<sno> 1 </sno>

<name> raja </name>

<age> 24 </age>

<place> B </place>

<type> BC </type>

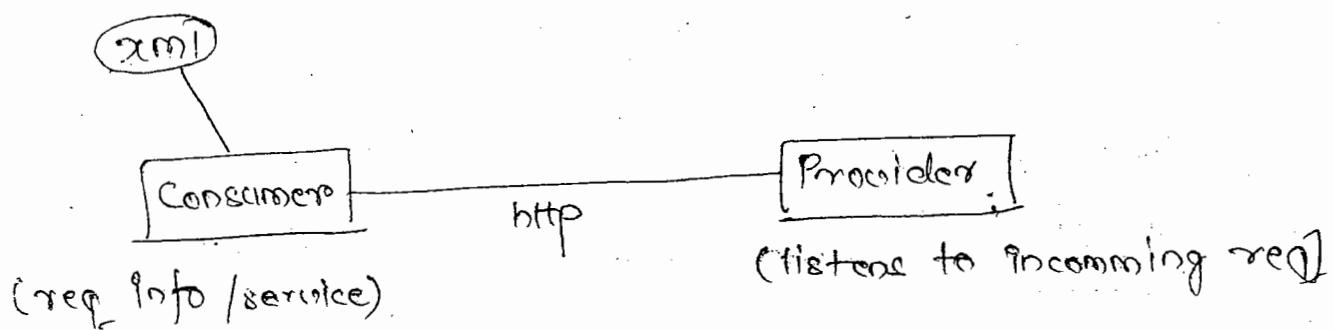
<date> 17/07/1989 </date>

</operation>

contains well def semantic & structure.

→ Data in XML doc is independent of prog lang & platform. (interoperable)

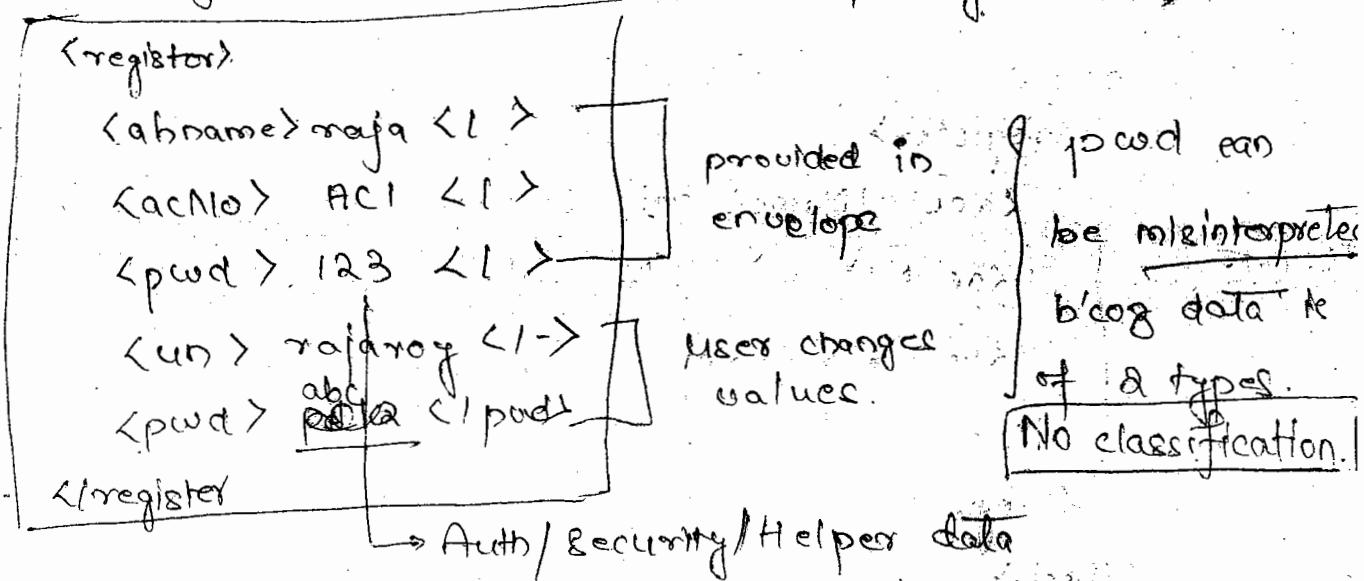
→ De Facto std of exchanging data b/w 2 sys over n/w is XML.



→ we need something on-top of XML.

### Bank System Use case

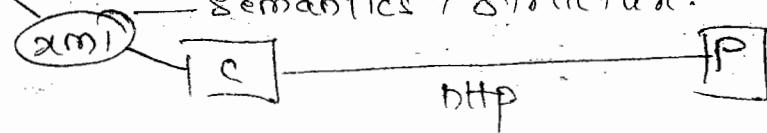
online user registration.



Semantics + Structured + Classification.

SOAP — classification

XML — Semantics + Structure.

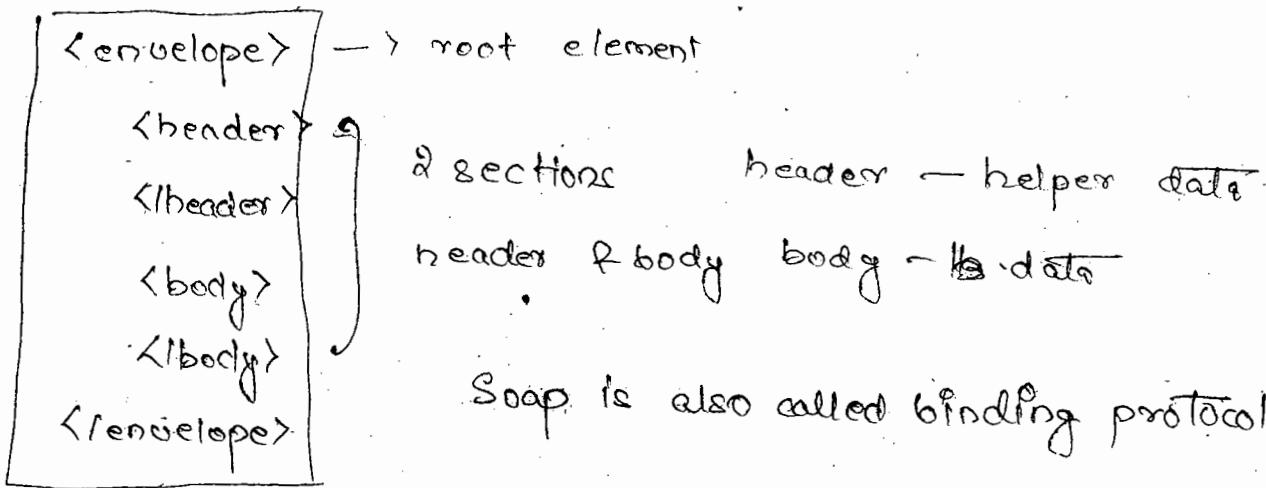


Simple Object Access Protocol: It's an XML language.

But, every XML isn't SOAP. (Markup language)

→ SOAP has predefined tags set of tags & structure.

### Structure of SOAP XML



<envelope>  
  <headers>  
    <pwd> 123 </pwd> } helper data  
  <header> 345  
  <body>  
    <register>  
      <acctName> maja </acctName>  
      <acctNo> ac1 </acctNo>  
      <fun> mafamoy </fun>  
    <pwd> abc </pwd>  
  </register>  
  </body>

</envelope>

carrier of actual payload.  
SOAP wraps XML so it's

called binding protocol.

(App Specific Protocol)

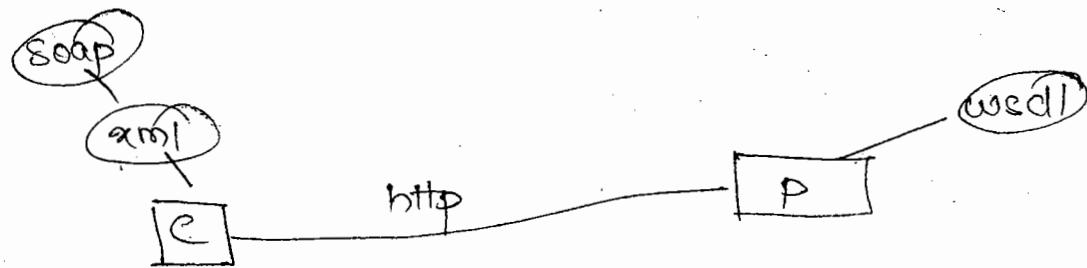
SOAP tells from which section  
to read what data

Soap is also called "carrier" of actual payload.  
(Info)

Web Service client → consumer. (C)

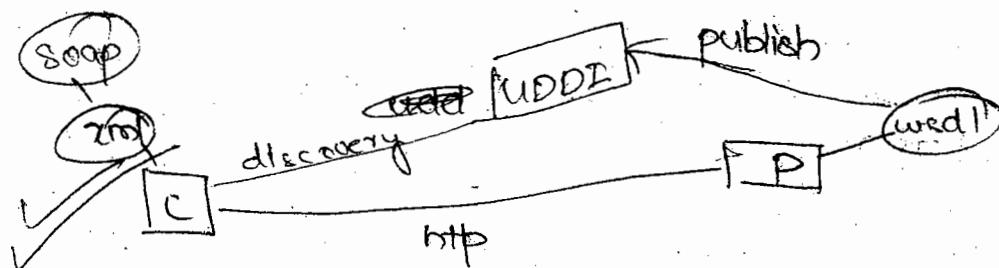
P → Web Service Provider documents the Info. abt services.

→ Documentation should be Interoperable too.



wsdl → Web Service Description Language (It's in xml)

Provider/Service/Web Service are interchangeable.



UDDI to place wsdl info.  
[classname, method, param, return types]  
registry

Discovery: process of connecting to UDDI & performs  
matching

UDDI : Universal Description Data Integration Registry

Global registry

UDDI helps to get details ~~befor~~ + of contract  
btw 2 parties.

UDDI → Development time repository  
or

Design time repository

## Interoperable

→ UDDI aren't mandatory, consumer can still communicate with providers b'coz UDDI provides governance

i.e. ~~it~~ provides info.

→ ~~UDDI~~ are well known but less used. (S/w providers provide it)

public (IDEE)

→ Global (Commercial / open source)  
~~Eg Reliance~~

private UDDI

Eg Reliance

(Belongs to Specific org)

Microsoft

Sun

IBM

Gave own standards

solution.

→ They tried to build interoperability with incompatible standards

WS-I (Web Service Interoperable Organization)

→ Non-profit organization.

→ BP 1.0 specification (Basic Profile 1.0)  
(first specification) (white paper)

[Open Paper = white paper] [Any I can ~~give~~ recommend]  
(Recommends set of standards to make Interoperable applications)

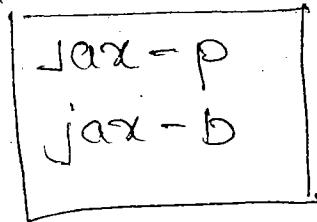
→ BP 1.1

Some extra standards

→ Adapting to BP 1.0 Java released JAX-RPC API.

→ Adapting to BP 1.1 Java released JAX-RS API.

→ To work with XML Java released 2 API.



→ saaJ API released by Java to work with SOAP. (SOAP with Attachment API for Java)

→ Port discovery from UDDI Java released

(jax-r & jax-registry)

→ Any prog. adhering obeying BP1.0 / BP1.1 spec then we can call the prog. & web service.

XML

→ Extensible Markup Language.

→ Language comprising of ~~a~~ set of tags.

→ Governed & owned by W3C. org.

→ Version → 1.0

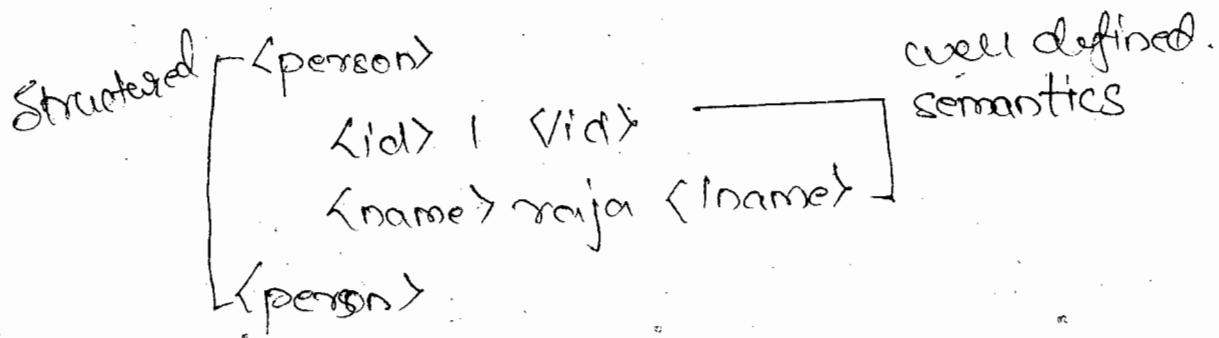
→ Used to represent some info in an "xml" format.

→ Well defined semantics & rigid fixed

- Data represented by XML is portable & interoperable
- De facto standard for exchange info b/w 2 computers.
- Every language has some key words (grammars)
- XML has 0 grammars.
- Every language has some syntactic rule (syntax)

### Rules

- Every XML should start with prolog processing instruction  
`<?xml version="1.0" encoding="utf-8" ?>`
- Parser: Reads content of XML, using some parsing technique.
- Info. should be enclosed b/w starting tag & ending tag
- Structured: knowing where it starts & ends.



- The level at which we started the start tag the end tag should be at same level.
  - Every start tag has an end tag.
- E.g.: Peculiar student writing notes in constructed way.  
well formed

### XML IDE'S

→ Altova XMLSpy, XMLSpy.

→ Jdk 1.6

① Format OS.

② Install Jdk1.6

③ Don't install weblogic, JDeveloper (for Weblogic)

④ Oracle Virtual ~~Machine~~.Box

~~⑤~~ ~~⑥~~ ~~⑦~~

⑥ Create a virtual hard drive now. → Create

⑦ VDI → Next

⑧ Fixed size (20.00GB) (Don't install on primary drive)

→ Install all languages.

⑨ Win XP.

Insert Guest Additions CD Image . . .

Right side ctrl key (Host key) [ctrl + C]

full screen mode

-- JDK 1.5 - 32bit

→ Install Altova

~~Ex:~~

Supplier & Retailer/Producers

Ex:

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

<purchaseOrder> ↗ Purchase Order contains

<orderItems> ↙ another element.

<item>

<itemCode> IC1001 </itemCode>

<quantity> 29 </quantity>

<item>

<itemCode> IC1002 </itemCode>

<quantity> 2 </quantity>

</item>

</orderItems>

<shippingAddress>

<addressLine1> Maitrivanam </addressLine2>

<addressLine2> Ameerpet </>

<city> Hyd </>

<state> AP </>

<country> India </>

</shippingAddress>

data

Well formedness guarantees running our on correctness.

E.g : Examination paper.

<Quantity> abc </Quantity> → well formed but not correct.

E.g : D.L. by RTO office



we need one more Doc

(Defines structure of Doc in another Doc)

Dtd or xsd

↓ Validation process

① What is well formedness & validity.

DTD (Document Type Definition Document)

→ Defines validity criteria / structure of XML.



< purchaseOrder >

<

2 types of elements

① Compound elements:

Contains child  
elements only

directly contains data inside it.

② Simple element:

contains anything  
compound, data

Declaring an simple element of my XML DTD

<!ELEMENT ELEMENT-NM CONTENT-TYPE>  
CONTENT-MODEL>

CONTENT-MODELS (Datatype) of DTD,

PCDATA {Parseable character data}  
CDATA ( " " )

<!ELEMENT item (#PCDATA)>

Ex: PCDATA

< > ' ' → Special characters  
symbols

Special character are parsed.

CDATA: ! Special characters are considered  
symbols

characters aren't as part of data. Treated as text:

IN ← Tells it's normal IN.

2 places to write data

① Attribute

② Element

`<book type="kids">` → CDATA

`<isbn> <a> </a> </isbn>` → PCDATA.

`</book>`

`<Book>`

`<html>`

`<head>`

`</head>`

`<body>`

`<p> </p>`

`</body>`

`</html>`

} should have head &  
body part only.

↳ Body may contain  
characters & any other tag.

PCDATA → Elements

CDATA → Attributes

Declaring Compound Elements

→ Bottom to Top

## Syntax

<!ELEMENT ELEMENT-NAME (CH1, CH2; CH3)>

comma is sequence separator.

<!ELEMENT item (itemCode, quantity)>

case-sensitive.

<!ELEMENT itemCode (#PCDATA)>

<!ELEMENT quantity (#PCDATA)>

→ write dtd as caps b'coz it's syntax.

<item>

<itemCode>      <itemCode>      Order should  
                        be match to

<Quantity>      <Quantity>      dt.d.

</item>

Occurrences of an element inside another

<a>                      + → 1 - n

<b> </b>                      \* → 0 - n

<b> </b>                      ? → 0 - 1

<b> </b>                      N/S → 1 - 1

</a>

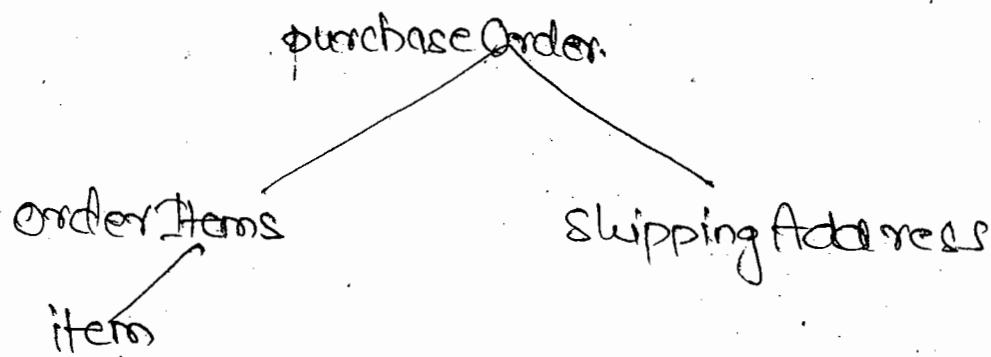
<!ELEMENT a (bt)>

<!ELEMENT b (#PCDATA)>

### Floating Conventions

→ Servlet-Name ✓  
(recommended)

servleflName X  
(not rec--)



<!ELEMENT purchaseOrder (orderItems,  
 shippingAddress) >

<!ELEMENT orderItems (items+) >

<purchaseOrder>

<orderItems>

<item>

<ItemCode>242</>

<Quantity>2452</>

</item>

</orderItems>

<shippingA >

Java class ← Instance of class obj  
if follows structure.

DTD/XSD ← if XML follows elated  
its called instance  
(created first) of the DTD.

⇒ DTD/XSD : XML Instance

Linking XML to DTD (validate)

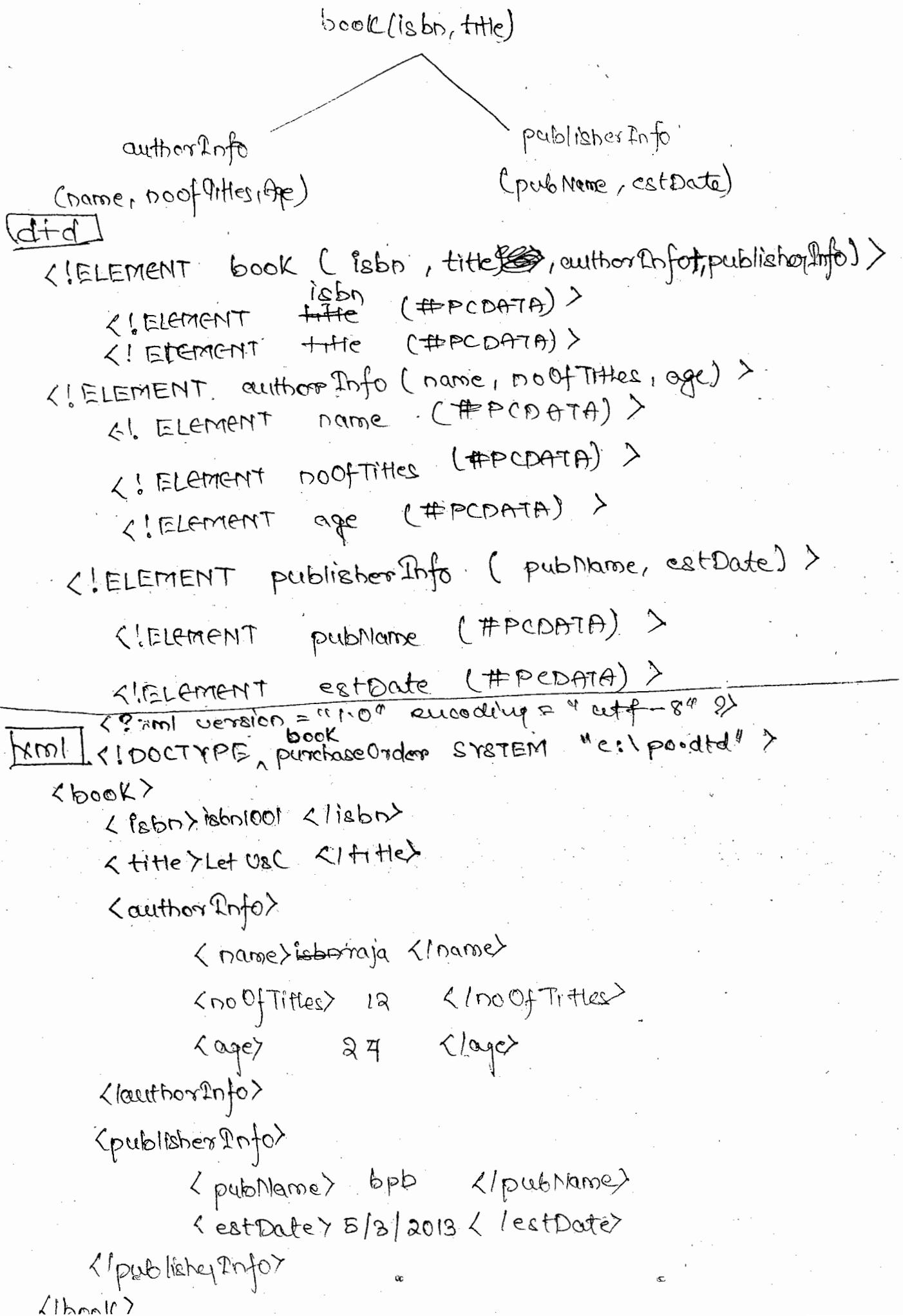
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE purchaseOrder SYSTEM/PUBLIC  
"C:\po.dtd">

! ⇒ stands for declaration.

SYSTEM ⇒ If DTD is in local machine

PUBLIC ⇒ A → a "remote".



## (Determining child)

<item>

  <itemCode> 123 </itemCode>

  <quantity> ABC </quantity>

} Valid

</item>

→ ~~But, XML~~ DTD can do only structural validation.

DTD is loosely typed language.

→ DTD's can never perform Data type validation.

## Limitations of DTD

<isbn> <isbn>                    } we require one

<serialNo> <serialNo>            } of them.

      |  
      | pipe

<!ELEMENT book ((isbn | serialNo), - - - )

Either or Separator

<!ELEMENT a (x|y) >

<a>

<!ELEMENT x (#PCDATA)>

<x> </x>

" " y ( " " )

<y> </y>

</a>

<!ELEMENT a (x|y)+ >

\* x or y may come occur

any no. of times in any order.

Occurrences of a group.

## ELEMENT

content group of elements any element can come  
any no. of times in any order.

E.g.: Purchasing veg or fruits.

<order>

<id> </id>

<mobile> </mobile>

<fruit>

<name> </name>

<pieces> </pieces>

<vegetable>

<type>

<quantity>

<vegetable>

Almonds

} fruit may come any no  
of times, vegetable may  
come any no. of times,  
both may come in any  
order.

④ (fruit; vegetable) +

## Mixed Content Model

<mail>

<to>

<from>

<msg>

<body>

</mail>

<mail>

or

ABC

</mail>

<!ELEMENT mail (PCDATA | to | from | msg | body)

```
<a>  
<x> </x>      <!ELEMENT a (#PCDATA | x | y)*>  
<y> </y>  
afdaf  
</a>
```

<!

## Working with ANY content type

```
<mail>          <!ELEMENT mail (to,frm,body)>  
<to>            <!ELEMENT to (#PCDATA)>  
<frm>           <!ELEMENT frm (#PCDATA)>  
<body>          <!ELEMENT body ANY>  
</mail>
```

Eq.  
<

<!!

→ can contain any parseable content.

- All are mandatory to write.

## Working with Attributes [contains only CDATA]

```
<book type="programm">    <book>  
<isbn> 1234567890  
<title> Let us C  
</book>          <type>  
                  <isbn>  
                  <title>  
                  </book>
```

→ Attributes provides supplementary info. of the element

→ Data doesn't expand. Parser should take it as text.

→ Data content should be less.

```
<!ELEMENT ELEMENT-NM ANY>
```

Syntax:

<!ATTLIST ELEMENT\_NM ATTR\_NM ((CONTENT\_TYPE)  
(ATTRIBUTE VALUE) )>

Fig:

```
<book type="">  
  <isbn>  
  <title>  
</book>
```

CONTENT TYPE (MODEL)

CDATA

ID

IDREF

NMTOKEN

EN1 | EN2 (Enumeration)

ATTRIBUTE VALUE

DEFAULT VALUE

#REQUIRED

#IMPLIED

#FIXED

<!ELEMENT book (isbn, title)>

<!ELEMENT isbn "">

<!ELEMENT title (#PCDATA)>

<!ATTLIST book type CDATA "fiction">

→ type is default value is fiction

CDATA #IMPLIED

→ type is default ~~other~~ value null

→ type CDATA #REQUIRED →

mandatory

CDATA #FIXED "kids" >

Only "kids" allowed.

→ type CDATA

→ type (fiction | thriller)

Encapsulation of values

→ type NMTOKEN (R & D)

Value has to be Element name only

→ type ID.

<fruit itemCode = " " > } no duplicates

<veg itemCode = " " > }  
unique.

→ type IDREF

<fruit itemCode = " " > } referring to

<veg itemCode = " " > } one of id  
in XML.

<cancellation itemCode = " " >

should be existing in my

~~Ex~~

book - category

E.g ① Book category (optional default = "fruits")

② " " (optional null)

③

### CONTENT-TYPE

① order-type (fruits / veg / mixed)

DTD

↳ Supports structural not data validations.

**XSD**

(XML Schema Definition) (Strongly typed language)

→ Used for validating contents of XML.

→ Defines structure of XML.

→ Governed by W3C.

→ Initial ver = 1.0 , curr ver = 1.1

→ XSD is an XML which is used to validate any other XML.

→ XSD has predefined set of elements. ↳ fixed element  
structure. It's special XML.

→ Starts with prolog. Has 1 root element.

<? version = "1.0" encoding = "utf-8" ?>

<xsi:schema>

</xsi:schema>

Simple compound

## Syntax for declaring simple elements

{xs:  
<element name="itemCode" type="xs:string"/>

<xs:element name="itemCode" type="xs:string" />  
intedit data type.

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

<xs:schema elementFormDefault="qualified/unqualified"  
attributeFormDefault="qualified/  
unqualified"/>

</xs:schema>

= "itemCode" = "xs:string" />

<xs:element name="quantity" type="xs:int"/>

= "city" = xs:string

= "state" = xs:string

= "zip" = "int" />

<xs:element name="country" type="xs:string"/>

<xs:complexType> } Represents structure of  
<xs:complexType> data inside an element.

<xsd:complexType name="itemType">

<xsd:sequence>

<xsd:element name="itemCode" type="xsd:string" />

<xsd:element name="quantity" type="xsd:int" />

<xsd:sequence>

<xsd:complexType>

↓

<xsd:element name="itemCode" type="xsd:itemType" />

Indicate occurrences

minOccurs = " " maxOccurs = " "

Linking element (xm) to xsd

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

<purchaseOrder noNamespaceSchemaLocation="file:///c:/po.xsd" >  
    <sup>FTP protocol</sup>

</purchaseOrder>

Advanced XSD

1 yr → Core + SCJP + Struts / Spring + ADO Java +  
Design Patterns

`<x> </x>`

`<y> </y>`

`(ia)`

~~✓~~ `xS:sequence → xS:all`  
any order

## Working with Inheritance

→ Primitive aren't able to represent complex data after elements.

## 2<sup>nd</sup> way of writing xsd

`<xS:schema>`

`<xS:element name="book">`

`<xS:complexType>` — Anonymous inner type.

`<xS:sequence>`

`<xS:element name="isbn" type="xS:string">`

`<!-- " " --> "title" " "`

`<xS:element name="authorInfo">`

`<xS:complexType>` —

`<xS:sequence>`

`<name>`

`<date>`

`<noOfTitles>`

`<xS:sequence>`

`<xS:complexType>`

`<xS:element>`

`<xS:element name="publisherInfo">`

`<xS:complexType>`

`<xS:sequence>`

`</xS:sequence>`

## Inheritance in xsd

```
<xsd:complexType name="USShippingAddressType">
  <xsd:complexContent>
    <xsd:extension base="ShippingAddressType">
      <xsd:sequence>
        <xsd:element name="country" type="xsd:int"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Inheriting one complex type  
another complex type.

## Working with either or content

→ Either p.o. or usp.o.

```
<purchaseOrder>
  <orderItems>
    </orderItems>
    <shippingAddress>
      <shippingAddress>
        <usShippingAddress>
          <usShippingAddress>
        </usShippingAddress>
      </usShippingAddress>
    </purchaseOrder>
```

OR

```
<xsd:complexType name="p0Type">
  <xsd:sequence>
    <xsd:element name="orderItems"/>
    <xsd:choice>
      <xsd:element name="shippingAddress"/>
      <xsd:element name="usShippingAddress"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
```

Overwritten choice

## Working with references

Ex

<book>

<isbn>

<title>

<authorInfo>

<name>

<dob>

<noOfTitles>

<public>

<authorInfo>

<publisherInfo>

<name>

<estDate>

<publisherInfo>

</book>

booklog

Global elements are at schema  
levels.

<xss:element name="name"  
type="xss:string"/>

<xss:element ref="name"/>

## Working with Groups

<xss:group name="grp1">

<xss:sequence>

<xss:element name="name" />

<xss:element name="language" />

<xss:sequence>

</xss:group>

selection

<xss:group ref="grp1" />

<xs:all>

<xs:element name="vegetable" type="vegetableType" />  
minOccurs = "1" maxOccurs = "unbounded" />

doesn't work

</xs:all>



<xs:all>

XSD version 1.0

p6lm

↳ child

minOccurs = 0 / 1

maxOccurs = 1

→ <xs:all>

→ can't use groups

<xs:all minOccurs = "1" maxOccurs = "unbounded" />

<xs:element name="vegetable" type="vegetableType" />

<xs:choice minOccurs = "1" maxOccurs = "1" />

~~xs:group~~

</xs:all>

minOccurs = 0 / 1

maxOccurs = 1

→ Use <xs:choice minOccurs = "1" />

maxOccurs = "unbounded" />

~~fillable~~ ~~minOccurs~~ ~~maxOccurs~~

fillable → Can content of element be  
empty or not.

↳ Presence of data inside element.

minOccurs,  
maxOccurs → presence of element.

By default minOccurs = 1  
maxOccurs = 1 & data is  
mandatory.

<xss:element name="" type="xs:string"

fillable = "true" />

For Int

& ~~minOccurs~~

< . . . fillable = "true" default = "1" />

## Working with Restrictions (Imposing Data Restrictions)

<zip> </zip>

↳ should contain int and must be of not be more than 5 digits.

→ xs:int: we need to define our own simple type.

<xs:simpleType name="zipType">

<xs:restriction base="xs:int">

<xs:totalDigits value="5"/>

<xs:restriction>

<xs:simpleType>

<xs:element name="zip" type="zipType"/>

<city>

<city> Working with Enumeration

↳ Hyd/Sc value only if it should contain

<xs:simpleType name="cityType">

<xs:restriction base="xs:string">

<xs:enumeration value="Secunderabad"/>

<xs:enumeration value="Hyderabad"/>

</xs:restric-

</xs:->

<xs:element name="city" type="cityType"/>

# Defining Attributes

Compound elements.

Simple Elements

Complex Elements

```
<shippingAddress type="permanent/temporary">  
    deliveryType="express/normal" </>
```

<xsd:schema>

<xsd:complexType name=

<xsd:attribute>

```
<xsd:complexType name="shippingAddress">  
    type=" - - " </>
```

<xsd:sequence>

<(xsd:sequence)>

```
<xsd:attribute name="type" type="xsd:string">
```

</xsd:complexType>

+ type = "permanent / temporary"

<xs:simpleType name = "addressType">

<xs:restriction base = "xs:string">

<xs:enumeration value = "permanent">

<"> "temporary" </>

</xs:restriction>

<xs:simpleType>

<shippingAddress type = "permanent">

(—xm)

<xs:attribute name = "type" type = "addressType"

use = "required" />

Simple elements

→ simple element can't have attributes.

<xs:complexType name = "addressLine2Type"

<xs:simpleContent>

<xs:extension base = "xs:string">

<xs:attribute name = "landmark" type = "xs:string" />

</xs:extension>

</xs:simpleContent>

</xs:complexType>

addressLine2 shouldn't exceed 50 chars

<xsd:simpleType name="addressLine2RestType">

<xsd:restriction base="xsd:string">

<xsd:minLength value="10"/>

<xsd:maxLength value="50"/>

</xsd:restriction>

</xsd:simpleType>

↳ <xsd:extension base="addressLine2RestType" />

## XSD vs DTD

Cow story:

→ Document Type Definition

Document.

→ Not XML type docs

→ XML Schema Document.

→ XML progs. need to learn it

→ XML docs.

→ Not type safe. (no int.)

→ Type safe.

→ No user def datatypes

→ Allows to define user def datatypes

→ No reusability

→ Reusability

→ Structured validations

→ Structured + Datatype

DTD

→ +, \*, ?: for occurrences

XSD

→ minOccurs & maxOccurs  
(more granular level control)

<xsd:simpleType />

No elements

<xsd:complexType>

No restriction

~~xsd namespace~~

→ Part of xsd itself.

→ Package in java resolves naming collision across several

data types

→ If there are complex type elements def by multiple  
prog. then chances exists that "name" will be same.

package com.xyz.abc; → import com.xyz.abc.A;

class A {

}

Bind into package

class B {

{ A a;

→ Declaring namespace → import namespace  
(XSD) (XML)

## Declaring namespaces in xsd

Sc

→ Name of package's package should be as unique as possible.

### Convention in naming package

Package: com . ibm . wsa . ui . dao  
company name      productname      module      name of class  
                        functionality

<xs:declare namespace="http://www.w3.org/2001/XMLSchema#"/>

<xs:schema targetNamespace="namespace\_label">

</xs:schema>

→ It's recommended to name our namespace label using uri. (Universal Resource Indicator)

uri → Gives physical location

domain name who exist nature of

uri - http://ebay.in/sales/types

Logical sequence of chapters

SchemaLocation = "namespace of uri"

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

<xsd:schema>

<xsd:element name="bool" type="booleanType"/>

<xsd:complexType name="boolType"/>

<xsd:sequence>

<xsd:element name="title" type="string"/>

<xsd:element name="isbn" type="string"/>

<xsd:element name="authorInfo" type="string"/>

<xsd:element name="publisherInfo" type="string"/>

<xsd:sequence>

</xsd:complexType>

<xsd:schema>

↓ with namespace

<xsd:schema targetNamespace="http://flipkart.com/sales/online">

xmldns:fo = "http://flipkart.com/sales/online">

<xsd:element name="bool" type="xsd:booleanType" elementFormDefault="unqualified"/>

xml

<book schemaLocation="http://flipkart.com/sales/online  
file:///c:/books/xsd">

xmldns:fo = "http://flipkart.com/sales/online" #

→ if parent is qualified - it's child needn't  
to be qualified.

namespace.

```
<book schemaLocation=" ... " "
```

xmlns="http://... " />

prefix is default.

Any namespace without prefix is considered as default namespace.

→ Here, element acts as qualified due to xml & unqualified due to xsd.

→ When we use element from Default "unqualified"  
never use default name space.

Namespaces are diff. if written

by diff. people or  
diff. purpose

## Import vs Include

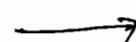
```
package pkg1;  
class A {  
}  
} → A class shouldn't exceed no. of  
than 2000 lines LOC. (sometimes  
4000 LOC)
```

① How many no. of packages in ur project & their

name ? package pkg1;

② class A1 // we divide class for ease of  
}

```
package pkg1;  
class A1
```



```
package sm1;  
class B1
```

} // written by others  
we use B in A class

part-1 (Working with more than 1 xsd)  
→ xsd can be as many no. of complex type & elements.

### part-1.xsd

```
<?xml version="1.0" encoding="utf-8"?>  
<xsi:schema targetNamespace="http://www.xmlext.com/xsd">  
  <xsi:complexType name="ShippingAddressType">  
    <xsi:sequence>  
      <xsi:element name="streetAddress" type="string"/>  
      <xsi:element name="city" type="string"/>  
      <xsi:element name="state" type="string"/>  
      <xsi:element name="zipCode" type="string"/>
```

### part-2.xsd

```
<?xml version="1.0" encoding="utf-8" ?>  
<xsi:schema targetNamespace="http://ebay.in/shoponline">  
  <xsi:complexType name="PurchaseOrderType">  
    <xsi:sequence>  
      <xsi:element name="orderItems" type="xs:complexType">  
        <xsi:sequence>  
          <xsi:element name="item" type="xs:complexType">  
            <xsi:sequence>  
              <xsi:element name="name" type="string"/>  
              <xsi:element name="quantity" type="int"/>  
              <xsi:element name="unitPrice" type="float"/>  
            </xsi:sequence>  
          </xsi:element>  
        </xsi:sequence>  
      </xsi:element>  
    </xsi:sequence>  
  </xsi:complexType>  
<xsi:element ref="xs:complexType" />  
→ we need to link xsd b'coz they are part  
of same people / purpose.
```

→ If first ~~same~~ xsd & 2<sup>nd</sup> xsd namespace are  
same, then use include.

```
<xsi:include schemaLocation=" " />
```

~~<targetNamespace = "eb"~~

PO2.xsd (my xsd)

→ Other xsd contains other namespace.

PO2.xsd

<xsi:schema --> xmlns:xs="http://--"

<xsi:import namespace="http://(ebay).in/schemas/line  
schemaLocation="file:///c:/App1/xsd/1/

<xsi:element ref = "e&1:shippingAddress" />

W3C-1A

██████████

→ xsd is also xml. It can be validated.

→ W3C gave a default namespace so we  
prefix elements with xsi:element

W3C - xsd → targetNamespace

<http://w3c.org/2001/XMLSchema>.

<xsi:schema xmlns:xs="http://--/xsd"

19-03-19 (Tuesday)

<purchaseOrder>

xsi:schemaLocation="ns1 ... xsdLoc">

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"



"instance"

Defines attributes to be used in xsd (Contains elements / attributes to be used in xsd)

19-03-14

JAX-P

- Java API for XML processing / parsing.
- Used for reading contents of XML in XML format.
- Earlier we have only file IO but it was unable to read in XML format.

Factory Design Pattern

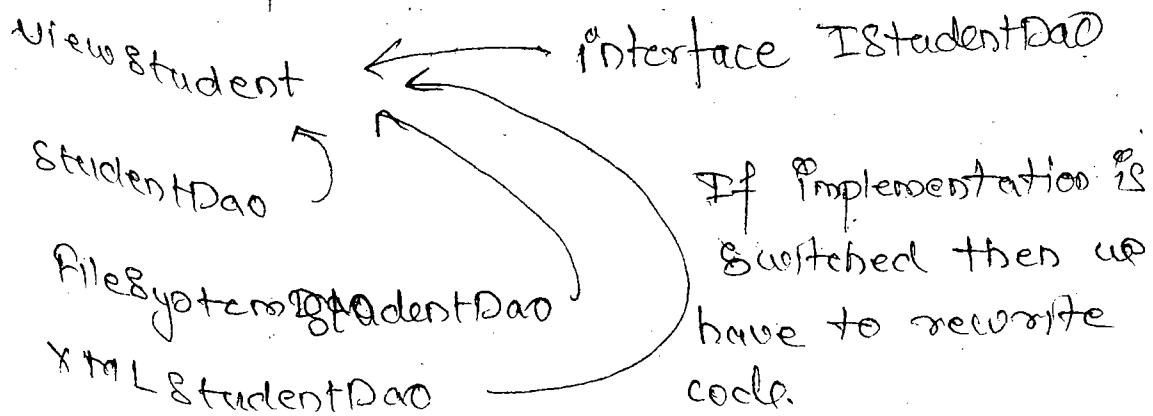
```
class ViewStudent  
{  
    class StudentDao  
}
```

```
class StudentDao
```

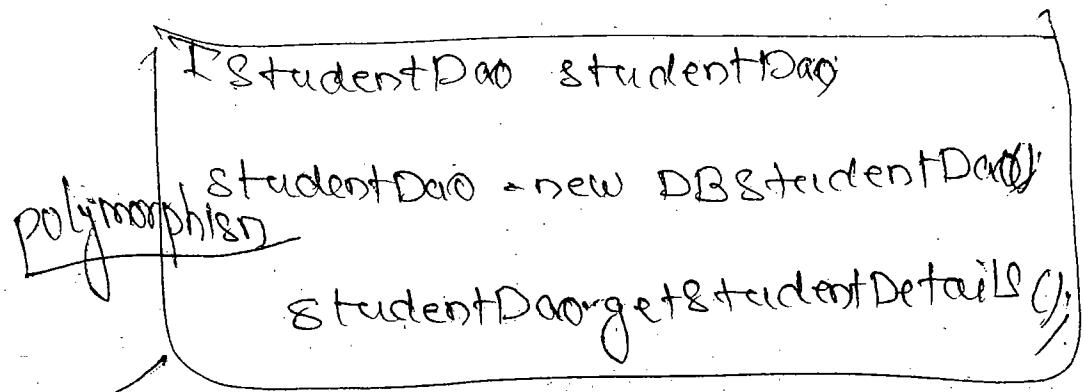
```
public StudentInfo getStudentDetails()
```

```
class ViewStudent  
{  
    public void render()
```

- ViewStudent is dependent on StudentDAO
- High level of coupling b'coz ViewStudent is directly trying to talk to class StudentDAO
- Always design to interface never designs to concrete implementation.



- IStudentDAO should be used now



- Calling is done based on obj to which it's pointing to
- Bt talking to interface minimal amt. of coupling is still left. & the class should know the complete instantiation process
- Every class in Java can be created using new
- The above pblm ① & ② can be solved by P.

Student DaoFactory {

```
    static  
    public IStudentDao getStudentDao(String type) {
```

```
        if (type.equals("clb"))
```

```
            return new DBStudentDao();
```

```
        "
```

```
}
```

→ As the above method contains logic to create another class they're called factory methods. As, the nature of method is static, it's called static factory method.

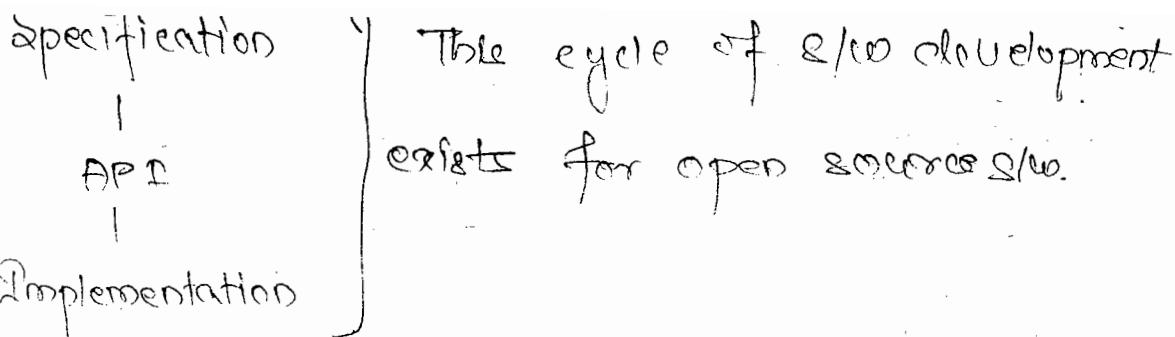
→ API ~~contains~~ interfaces & abstract classes mostly. (E.g : Connection, Statement, ResultSet)

Connection conn = DriverManager.getConnection()  
API Vendor

→ we don't directly talk to vendor specific classes this is the advantage of API.

Connection conn = new OracleConnImp(); — X

→ Factory classes are meant for providing abstraction on creating the obj of another class.



→ Specification is white paper / open paper. I want to present idea to world b'coz the initial thought & with help I can get my idea more matured. Any1 can write a spec.

→ first spec doc. is called draft / initial spec. doc.

→ ~~Re~~ when repeated exchange of idea, & juncture point is reached & I can have final specification ~~doc.~~ doc.

→ Using this final spec. doc we will design API.

E.g.: Building House:

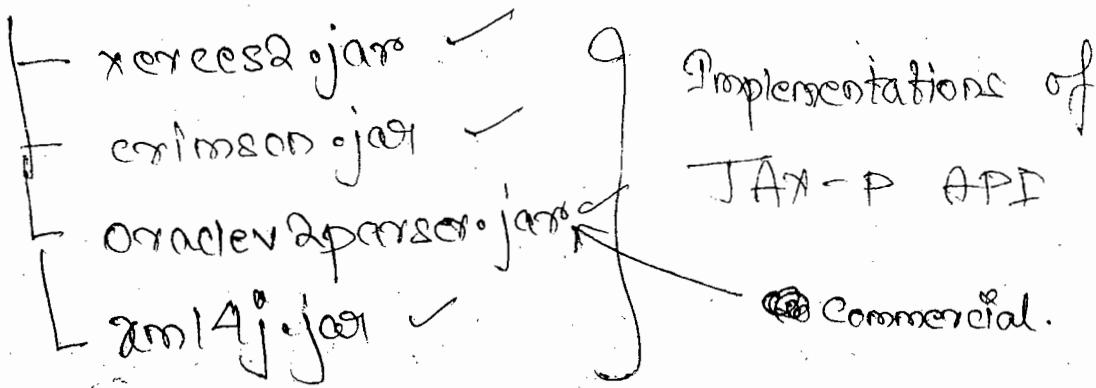
JAX-P API

→ Third party vendors gave some S/W on existing ~~the~~ Java S/W like DOM4J, XML4J, SAX4J. before run gave API.

E.g.: Developing a app to read XML using Java

→ If API isn't there then we have to change the total implementation switching from one tech to other.

## JAX-P API



→ Prior to JAX-P, `xml4j.jar` was owned by IBM.  
Then, IBM gave it to Apache.

→ `OracleX2Parser.jar` comes with weblogic server

→ ~~No commercial~~ Non commercial org gets ~~prof~~ revenue by giving online trainings, support, consultants.

→ From developer's viewpoint there're many impl for API.

### jdk1.4

→ No support for JAX-P API.

Prog should depend on JAX-P API + Impl jars

### jdk1.5+

→ JDBC is shipped with it.

→ JAX-P API will be shipped as part of `xt.jar` or `tools.jar`.

→ With JAX-P API `xerces2.jar` implied default impl is been provided.

says to read contents off XML tree

→ (e)

- 2 (ways) technique to read XML (famous)

SAX } Universal access methodology  
DOM } (any lang. can use it)

To

repr.

Sour.

[Use]

<..

<pc

### SAX (in general)' Procedure to read XML

Simple Access for XML API.

For reading contents / parsing contents of XML document

Sequential Access Model. [Top → Bottom processing]

SAX engine places a pointer on top & reads element & increments pointer till bottom

SAX uses event based processing model.

Sou.

### Event Based Processing Mechanism

- AWT/Swing's event bubbling tech.

'f

- 3 actors

→ x

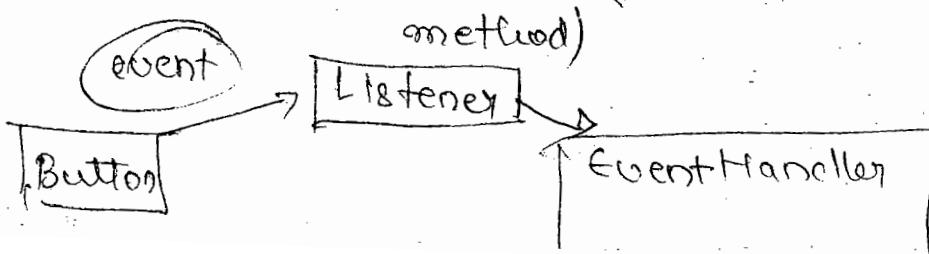
① - source (originator / triggerer of events)  
(buttons)

→ SI

② Listener (knows whom to call to process the events)

or

③ Event Handler (class having event handling method)



→ event contains one internal obj called source.

To know which event happens - Event obj

represents state of activity that happened on source.

### Using SAX Event Based Processing Mechanism

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

<p> — starting element — startElement() event is triggered  
</p>

<i>

<i> I1000 </i>

<i> i7 </i>

</i>

Parser

Source

<i> Input

<i>

startDocument()

startElement()

endElement()

endDocument()

putData()

(Handling)

</p> - ending element - endElementEvent is triggered.

→ XML acts as a source

→ SAX consumes very less memory & it's faster

in processing cmp to DOM

## Working with SAX In JAX-P

class POHandler

→ If you write your own methods in Handler  
the parser won't know which method to  
invoke for events.

→ Handler class should extend DefaultHandler. [A]  
(Normal class)

class POHandler {

    class POHandler extends DefaultHandler {

        void startDocument() {

    }

        void endDocument() {

    }

        void startElement() {

    }

        void endElement() {

    }

        void characters() {

    }

SAX is read only

API

startElement( - - - )

namespace uri, local\_element\_name,

qualified\_name, attribute\_list

characters( - - - )

char[] data, offset, length

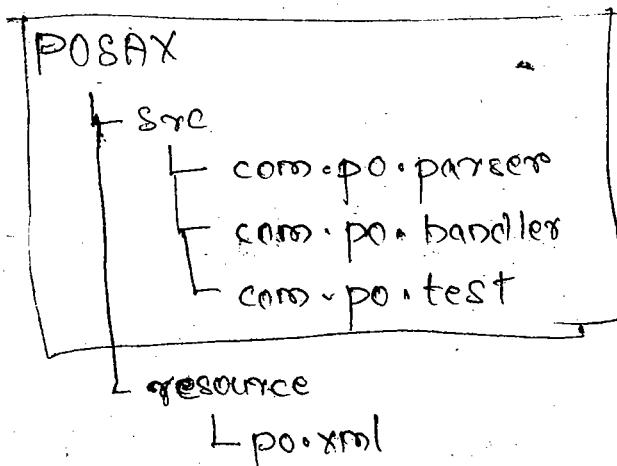
length will always be fixed or short answer.

## SAX Parser [~~Interface~~ ID: SAX-P API]

→ parse(XmlDoc, Obj)

→ SAXParser takes XML & handler obj as input

SAXParserFactory → Factory to create SAXParser obj.



SAX parser by default  
won't consider namespaces

→ Alt + Shift + S + V → Overrule super class methods

class ViewPOParser {

public void print(@final String path) {

SAXParserFactory factory = SAXParserFactory.  
new  
getInstance();

SAXParser parser = factory.newSAXParser();

parser.parse(new File(path), new ViewPOHandler());

factory.setNamespaceAware(true);

④ class POCTest

psum(~) {

ViewPOParser poParser = new — ()

poParser.print(FILE\_PATH);

po.xml

alt-enter

↳ Resource → Location

### ItemCountHandler

class ItemCountHandler extends DefaultHandler {

int count;

startElement (namespaceURI, localName,  
qName, attribute) {

→ Always write initialization logic in

startDocument(). If multiple doc are there the  
cons. will be executed once & so count will

be initialized to 0 only once.

private int total;

↳ String lastProcessedElement;

startElement (— — — —)

lastProcessedElement - qName

}

characters (-, -, -) } check for null

If (!lastProcessedElement.equals("Quantity")) {

} total = - - -

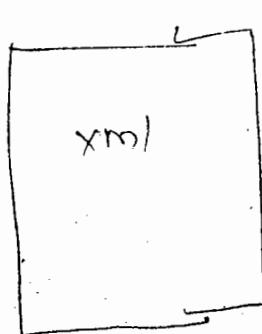
Alt+Shift+S — Alt+G — Getter

DOM — W3C

④ → Document Object Model.

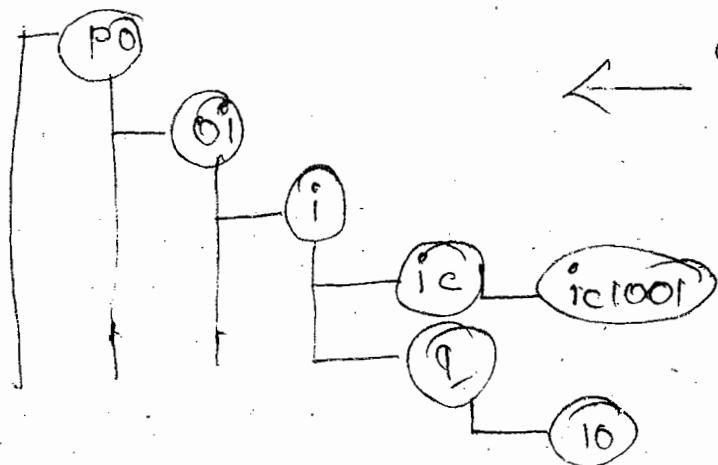
→ Processes XML randomly

→ Hierarchical based processing model.



DOM ~~processes~~ loads entire XML ~~to~~ memory in 1 single shot. Checks for well-formedness before loading & creates tree structure.

document node, element node, text node



Content tree

(Linked List)

Entire XML is loaded  
into memory  
foot print.

[1kb XML → 10kb of JVM to process it]

## Working with DOM using JAXP

→ In Document object the XML will be stored.

Interface

→ DocumentBuilder — A.C



DOM can read,

modify & create

new document.

→ DocumentBuilderFactory

DocumentBuilderFactory factory =

DocumentBuilderFactory.newInstance()

DocumentBuilder builder = //supports reading & modifying

factory.newDocumentBuilder()

Document doc = builder.parse(  
new File(path));

→ All nodes are impl of Node (I).

→ DOM places a pointer before start of  
node.

+ com.po.dom.parser  
- com.po.dom.test

class PODOMParser

public void printNode (final String path) {

DocumentBuilder --

-- --  
Sax (doc.getFirstChild().getNodeName());

→ Total element → TO.

✓ Min qty allowed to PO should be 100. Write  
one sax parser to validate.

✓ Calc. invoice amt. for PO by assuming 10rs as  
price per piece.

✓ fraud zip. codes detection.

Logic to traverse in DOM

- Inorder, PostOrder, Preorder.
- Recursive tech / Stack

```
<root>
  <i> 1</i><1>
    <i> 10</i>
  </1>
</root>
```

Alg0:

~~function~~

FUNCTION traverse (Node node) THEN

IF node.type == DOCUMENT\_NODE THEN

traverse (node.getFirstChild);

ENDIF;

IF node.type == ELEMENT\_NODE THEN

print nodeName;

children = node.getChildNodes();

FOR child : children LOOP

traverse child;

END FOR;

~~else~~ / ENDIF;

~~else~~; IF node.type == TEXT\_NODE THEN

print node.value;

ENDIF;

END;

## XML Traverser

```
class XMLTraverser {
```

```
    public void parse(final String path) {
```

```
        DocumentBuilderFactory factory =
```

```
            DocumentBuilderFactory.newInstance();
```

```
        DocumentBuilder builder =
```

```
            factory.newDocumentBuilder();
```

```
        Document doc =
```

```
            builder.parse(new File(path));
```

```
        processNode(doc);
```

```
    void processNode(Node node)
```

```
} void processNode(Node node) {
```

```
    switch (node.getNodeType()) {
```

```
        case Node.DOCUMENT_NODE:
```

```
            processNode(node.getFirstChild());
```

```
            break;
```

```
        case Node.ELEMENT_NODE {
```

```
            System.out.println(node.getNodeName());
```

```
            " < " + _____ + " > "
```

NodeList children = node.getchildNode();

for (int i=0; i < children.getLength(); i++) {

    Node child = children.item(i);

    processNode(child);

}

System.out.println("< " + node.getNodeName() + ">");

break;

case 3: Node.TEXT-NODE

System.out.println(node.getNodeValue());

break;

① Do you worked on web service? Experience?

Are u good at XML, xsd, dtd? Parsing ways

of XML? Which API used? SAX vs DOM?

Which parser do you prefer? (Analysing power)

→ It's not what I know to work on it. I can't  
decide at one single shot until I have reqd.

→ size of XML [Intra. can't guarantee upper limit]

→ reading only or modify & extract too.

→ Partial values of XML or all values

~~Less size~~

Fixed size - DOM

SAX

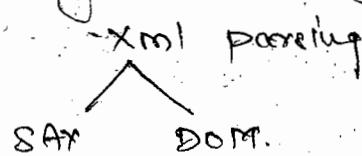
- Simple Access for XML API
- Sequential
- Less ~~useful~~ memory.
- Faster
- Event based processing model
- Read only API.

DOM

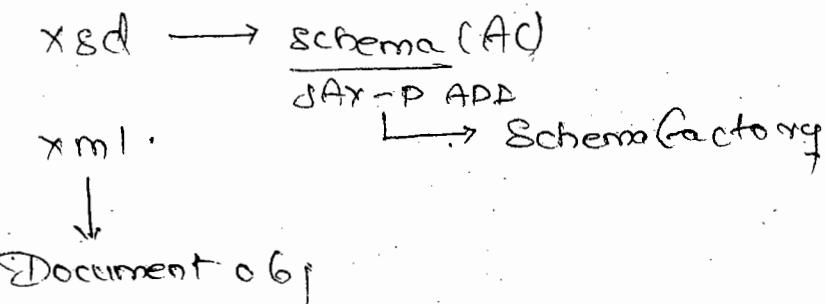
- Document obj model
- Random.
- More
- Slower.
- Hierarchical.
- Read + modify + manipulate.

- SAX & DOM parse only checks for well formedness
- JAX-P API contains classes for reading XML R
- checking against xsd too

### Validating XML through JAX-P



- ⑥ Prg to read XML & validate it.



For 10 XML instead of loading 10 xsd ~~write~~ load it into one obj & validate

```
SchemaFactory sfactory = SchemaFactory.newInstance("nsuri");
Schema schema = sfactory.newSchema("po.xsd");
Validator validator = schema.newValidator();
validator.validate(xml);
// write parsing log.
```

Prgm:

```
parse (String path) {
    // validation logic
```

```
SchemaFactory f = SchemaFactory.newInstance(
    XMLConstants.W3C_XML_SCHEMA_NS_URI);
```

```
Schema poschema = f.newSchema(new File("xsdpath"));
```

```
Validator validator = poschema.newValidator();
```

```
validator.validate (new StreamSource (new File (path)))
```

// parsing logic

f

## JAX-B API

SAX (Streaming API for XML)

↳ By IBM

→ Better than SAX & DOM.

→ It's a technique.

→ Pull based event parser.

→ Uses less memory.

→ JDK 1.6 f

## Java Architecture for XML Binding API

→ JAX-B provides XML to gives an object so that B-logic can easily be managed.

### History

- Introduced to reduce complexity of SAX & DOM.
- Binds XML to obj & obj back to XML.
- XML binding is an universal technique given for any language.

02-04-14

- JAX-B extracts entire XML & represents into obj to develop B-logic so, java developer is comfortable.
- Third party vendor specific libraries:

+ xml beans (3)

+ castor (2)

+ jibx (1)

+ ADB (apache data binding api)

+ jaxbolution

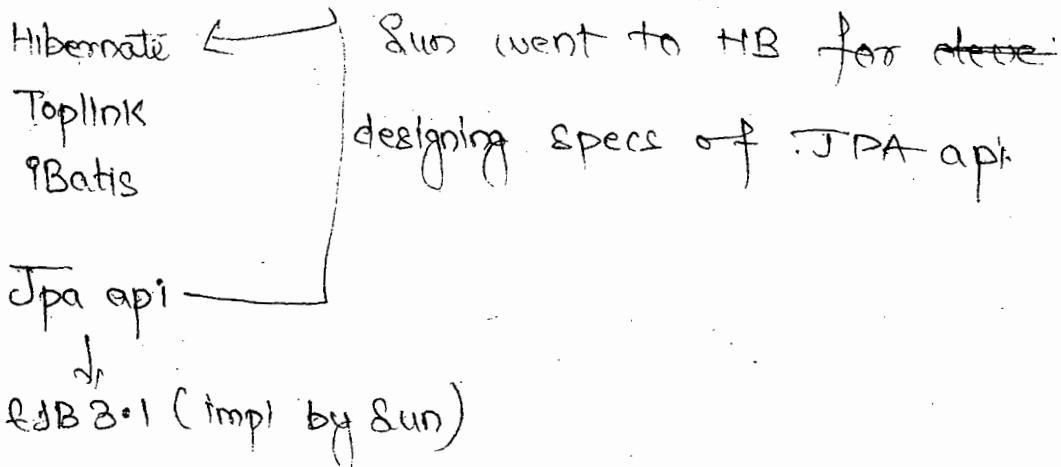
To write your own Jax-b api.

→ Jax-b api implementation

+ Jaxb-ri (reference implementation)

→ No vendor gave impl for Jax-b api so sun went to every vendor. B'coz sun didn't contacted anyone while designing Jax-b api. Sun learnt + learnt a lesson.

### ORM

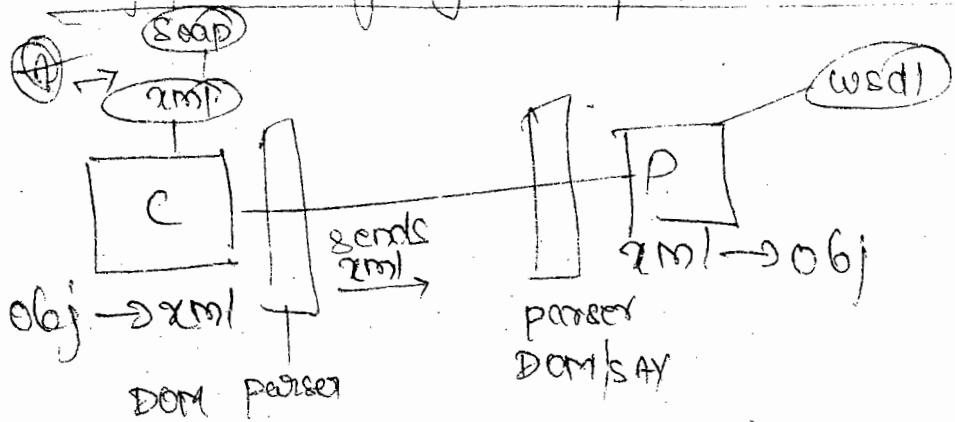


→ One more impl of Jax-b api

↳ Jaxme

→ Sun promotes Jax-b api throughout its products.

① Significance of Jax-b api in Web services?

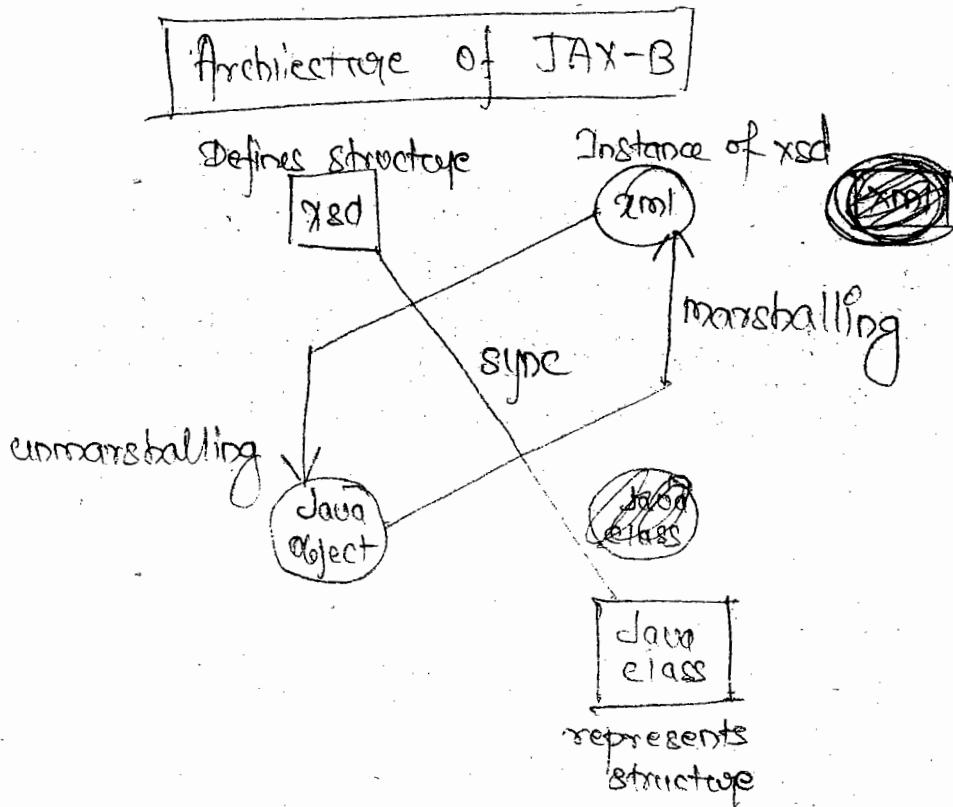


- Consumer can send multiple obj as inputs.
- If there are 2 obj then on consumer side we need 2 parser & 2 parser on provider side.
- Instead of using parsing used binding  
jax-b api

Q) Where did u use jax-b in ur project?

A) We're using web services as part of our proj. We don't send serializable obj but we need to convert in provider side & vice-versa in java.

To automate the process of converting obj to xml & xml to obj we use jax-b api.



If java class can represent a structure of xsd then we can represent ~~instance of~~ contents of xml into

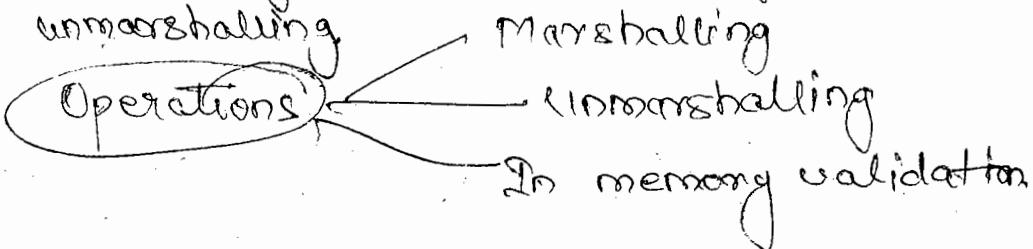
Process of converting an XML obj data into its associated class represented structure is called unmarshalling.

B-4-14

→ Unless the obj holding data has structure we can't do marshalling & unmarshalling.

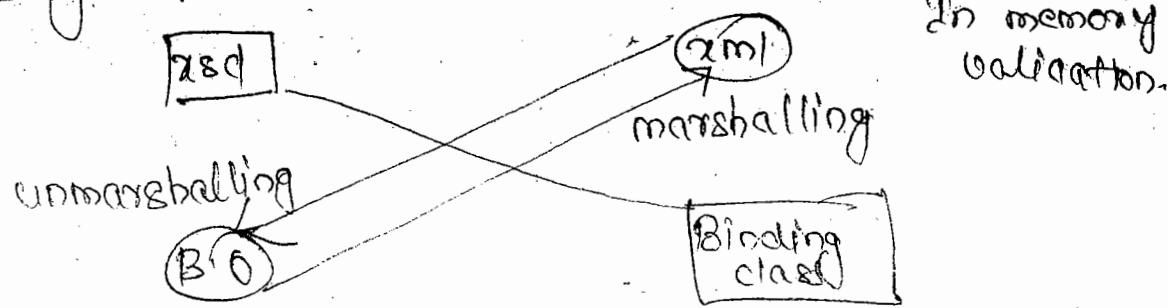
Inmemory Validation:

Performing validation during marshalling or unmarshalling.



Binding class: Class i.e. used to convert XML into java obj & vice-versa. & it's obj eye called

Binding obj:



In memory validation.

→ 1 complex type in xsd = 1 java class

All complex types are interlinked.  
using def data type.

<xsd:complexType name="itemType">

<xsd:sequence>

<xsd:element name="itemCode" /> B

<xsd:element name="quantity" />

<!--

class ItemType {

    String itemCode;

    int quantity;

}

For PO xsd

→ 1 Binding class

class ShippingAddressType {

    String addressLine1;

    String addressLine2;

    String city;

    String state;

    String zip;

    String country;

// setters & getters

→ 2nd Binding class

class OrderItemsType {

    List<ItemType> item;

}

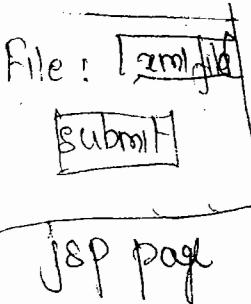
- 3rd Binding class

class PurchaseOrderType {

    OrderItemsType orderItems;

}

    ShippingAddressType shippingAddress;



```
HttpServlet {
    void service(....) {
        // Get contents of xml file
        // parsing or binding
    }
}
```

unmarshalling

→ Every time when we get new data we need to do

marshalling or unmarshalling ①

Before it we need to perform inmemory validation ②

So, these are called.

## Runtime Ops (③ ops)

→ If there are 2000+ complex types in xsd we have to write 2000+ classes which may be complex & we have to study the composition. [Time Taking process]

→ So, JAX-B provided 2 tools  
one time      ↗ xjc [xml schema document to java compiler]  
ops            ↗ schemagen [schema generator]

→ xjc is complex.

xsd → xjc → ~~java~~ Binding class.

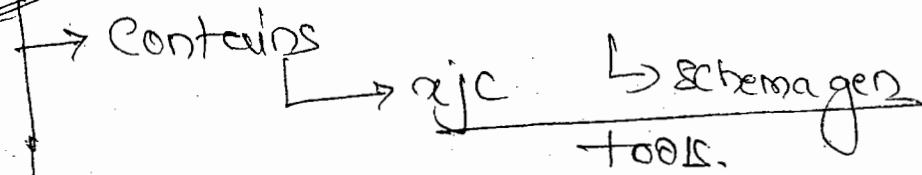
B.C → schemagen → xsd.

→ In JAX-B 2 types of ops are there:  
    ↗ runtime

    ↗ annotation

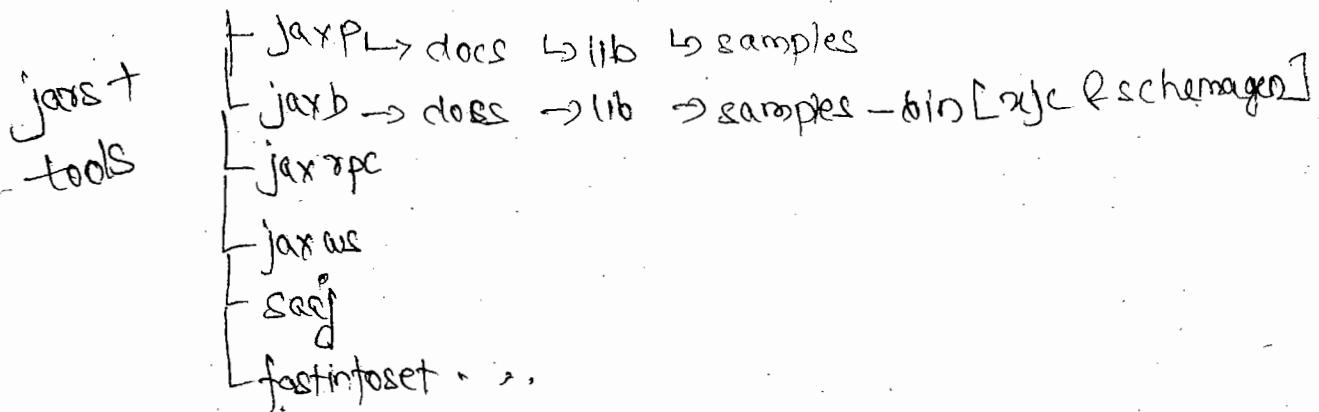
One-time ops: For every XML we needn't create new binding class.

~~jwsdp~~ [java web service developer pack] [by sun]



↳ Single stop for all the things related to web services.

↳ c:\sun\jwsdp-2.0 folder [default]



# Installation of JWE&DP

## CLIENT OS

- ① JDK 1.5 - 22.
- ② Tomcat-6wspip → zip file.
- ③ Program files → Apache Software Foundation → paste zip file
- ④ Extract to here → Delete zip file.
- ⑤ Jwe&dp folder → Double click on .exe file
- ⑥ Accept terms
- ⑦ Select path
- ⑧ Installation folder "C:\sun\jwe&dp-2.0"
- ⑨ Install Jwe&dp SW with
  - Download Sunone app servers
  - No servers

Browse

Browse to Tomcat Apache SW Foundation

◦ Install with tomcat

## ⑩ How to run xjc tool?

⑪

xsd

|  
xjc → binding classes.

POJAXB

```
  +-- src  
    +-- bin  
      +-- resource  
        +-- po.xsd
```

javac -d . xjc  
Additional switch

cmd:

```
set path=%path%; C:\sun\jwe&dp-2.0\jaxb\bin
```

```
POJAXB> xjc resource\po.xsd
```

How to display date in Locale specific format (MLG)

ResourceBundleMessageSource      id = messageSource  
    └ baseName  
    └ baseNames.

(I) Beanfactory, MessageSource (I)

    ↑  
ApplicationContext (I)

    └ getMessage(&string Key, Object[] params, Locale)

⇒ Place properties file directly into src folder

    └ getMessage(&string Key, Object[], "Default msg", Locale);

Q1

~~Lookup Method Injection~~      ↪ Dependency Lookup

```
class XServlet extends HttpServlet  
private Connection con;  
void service (req, res){}
```

    |  
    |  
    |

web.xml

/x

Container creates one obj for servlet class but one  
thread per request

Q1

> xjc -d src resource\po.xsd

-verbose [Additional info about how tools execute]

07-04-14  
Monday

> xjc -d src -verbose resource\po.xsd

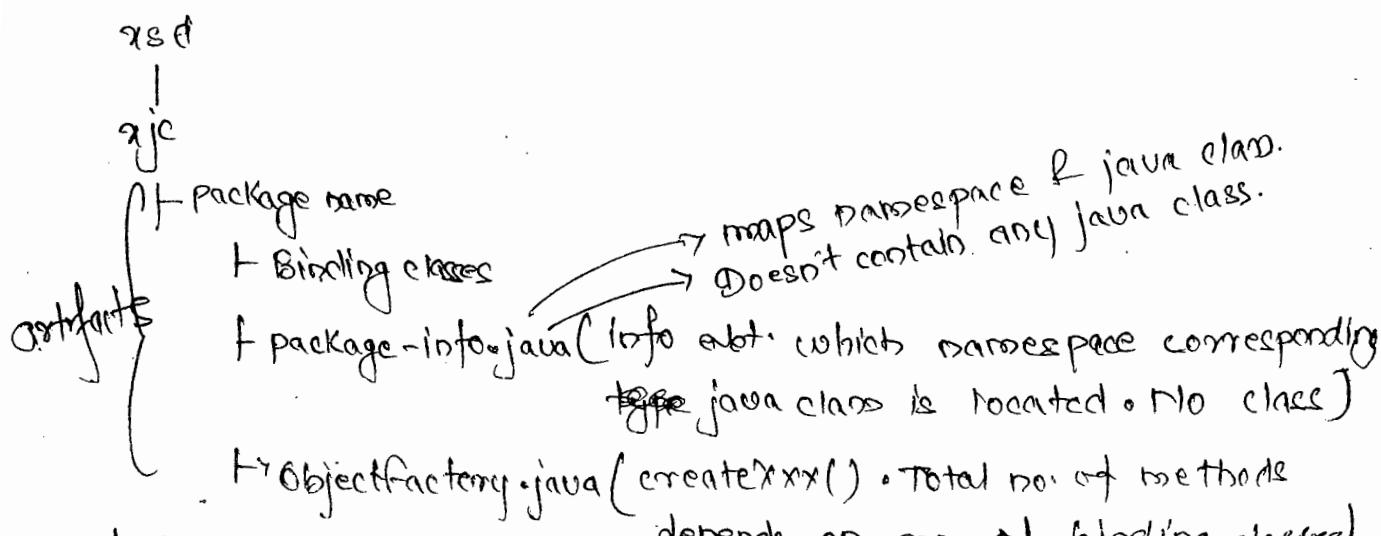
① What will get generated when we run xjc ?

A) http://ebay.in/sales/types

↓  
truncated ↓

ebay.in/sales/types

in.ebay.sales.types



xjc

Switches

## package-info.java

- It's reqd. b'coz if we specify explicit package then, to know abt. binding classes location & respective package. JAXB lets us to customize the package name.

↳ xs:schema

```

targetNamespace = "http://ebay.in/sales/types"
 xmlns:et = "http://ebay.in/sales/types"
 <xst:complexType name="orderItemstype">
   <xst:complexType name="itemType">
     <xst:sequence>
       <xst:element name="itemCode"
         type="xs:string">
         <-- quantity
       </xst:sequence>
     </xst:complexType>
   </xst:complexType>
 </xs:schema>
  
```

Left: orderItem  
 OrderItem  
 schemaLocation =  
 http://ebay.in/sales/type  
 file:///C:/po.xsd  
 xmlns:est =  
 http://ebay.in/sales/types  
 <items>  
 <itemCode>1C1001</>  
 <quantity>17.</>

package in.ebay.sales.types

class ItemType {

String itemCode;

int quantity;

class OrderItemstype {

ItemType item;

ObjectFactory

createXxx()  
classNames

package-info.java [Namespac]  
 @xmlSchema(name = "")  
 package in.ebay.sales.types;

→ XML as input

→ mappings.xml

~~class~~ → ~~entity~~

< mappings >

< class name = "OrderItem" >

element = "orderItem" >

< element name = "itemType" type = "itemType" >

< element name = "itemCode" alt = "itemCode" />

< item >

< class >

→ At that time  
this mapping XML was  
generated.

< mappings >

→ Now Tax-P 2.0 is redesigned to

work with Tax 1.5.

→ Now it uses annotations instead of

mappings.xml.

Annotations are called source code metadata.

@XMLType("itemType")

@XMLAccessorType(AccessType.FIELD) ↴  
class ItemType {  
 • METHOD How to access data  
 from XML.

@XMLElement(namespace = "")

String itemCode;

@XMLElement(namespace = "")

int quantity;

108-09-14  
Tuesday

- Jax-p uses STAX to parse the contents of XML
- STAX starts parsing p0.xml, picks up namespace goes to xsd, picks up namespace, looks for binding class with `@XMLType(" -- ")` & creates an obj, then again goes to xsd and so then it goes to binding class to locate attributes.

### JAX-RPC API

- ↳ JAX-P (equivalent parsing methodology)

### JAX-WS API

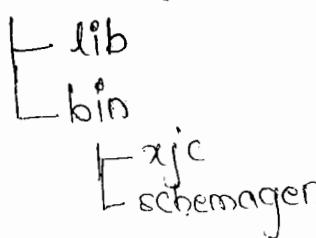
- JAX-B (recommended)  
Binding methodology.

### JAX-B API

- ↳ STAX parsing tech.

- - JDK 1.5 (No support for Jax-b) (download separately)  
↳ Jwsdp-2.0

- JDK 1.6+ (Direct support for JAX-B)



} for compilation jdk ~~1.6~~/lib is used, for execution jre/lib is used.

- JDK 1.4  
JAX 1.5  
JAX 1.6

} API  
versions  
(Backward compatibility issues)

JAX-B API  
1.x  
2.x  
[ Backward compatibility issues ]  
[ classes in 1.x API have been removed in 2.x API ]

> JDK 1.5  
  └ JaxB API 1.0  
    └ Jax-B API 1.0

└ ~~JaxB API~~  
  └ Jax-B API 2.0

POJAXB  
  └ src  
    └ lib  
      └ resources  
        └ pooxml  
          └ pooxed.

Run

cmd command  
(buffer) (no prev cmd buffering)

Font → Lucida Console

layout → Buffer

alt + space [+] ↵ + P

paste ↑  
in cmd.

POJAXB > xjc -d src -verbose resources\pooxed

How to perform Runtime Ops

## ① Unmarshalling

↳ Input : XML

↳ UnMarshaller (I)

  └ unmarshal (xml) [ Object unmarshal (xml) ]

JAXB API

↳ JAXB RI is vendor

## JAXBContext (factory ~~class~~)

↳ Acts as factory for unmarshaller.

↳ Creates In memory from memory space to hold JAXB classes only. (like class loader)  
(Acts as a container)

09-04-14  
Wednesday

↳ Loads all the classes in a package into Context

JAXBContext jContext = JAXBContext.newInstance()

(packageName);

or

↳ All the classes in package are loaded, sometimes can be performance problem.

JAXBContext jContext = JAXBContext.newInstance  
(classType);

Unmarshaller unmarshaller = jContext.createUnmarshaller();

Object unmarshal (new File(xml));

→ There can be more than one classes in

JAXBContext with same name as complex type

name in two diff xsd can be same. So, at that time the context will be searched & two classes with same name will be ~~found~~ there. Then, the Unmarshaller will check for package to



target namespace and find the particular package for locating the `rend`: class.

### Jaxb1.x

→ `@XmlAccessorType(XmlAccessType.FIELD)` → `② - - (XmlAccessType.FIELD,`  
→ `JAXBElement jElement = (JAXBElement) → PurchaseOrderType pot =`  
~~unmarshaller.unmarshal(xml);~~  
~~return type is Object~~  
~~PurchaseOrderType pot =~~  
`jElement.getValue();`

### Jaxb2.x

`② - - (XmlAccessType.FIELD,`  
`unmarshaller.unmarshal(xml);`

```
class A {
    List l;
    List getL() {
        if (l == null) {
            l = new ArrayList();
        }
        return l;
    }
}
A a = new A();
a.getL().add("1");
```

} never expose setter,  
Then chances are there to  
lose existing values

```
class OrderItemsType {
    List<OrderItem> items;
    if (item == null) {
        item = new ItemType();
    }
}
```

10-04-17  
Thursday

## ② Marshalling

Marshaller marshaller = jContext.createMarshaller();

marshaller.marshal(po, System.out);

ShippingAddress Type      ShippingAddress ;  
class = complex type      attribute = element of XML.

## ③ XmlRootElement(name = "purchaseOrder")

→ Since, we nowhere declared purchaseOrder attr

so, how the Marshaller will know that the

root element is purchaseOrder. So, we need to use

the above annotation.

## ④ In-Memory Validation

→ In Unmarshalling Unmarshaller, marshalling Marshaller

will only check well-formedness, so we have to

tell it to validate.

→ unmarshaller.setSchemaLocation(" ");

→ If we set path of xsd, then we have to read xsd for 10 times for 10 xml doc.

→ If we use Schema obj only one time xsd will be read.

```
// create SchemaFactory  
// create Schema  
// create Unmarshaller  
unmarshaller.setSchema(poSchema); }  
// Do unmarshalling  
} Enabling in-memory validation.
```

## JAX-RPC Web Services

### Basic Terminology around Web Services (Pre-requisite)

Consumer = Web Service Client/Consumer.

Provider = Web Service.

→ Now we will work to develop Provider.

① Are you working on provider or consumer?

② Which type of Web Services are you developing in your project?

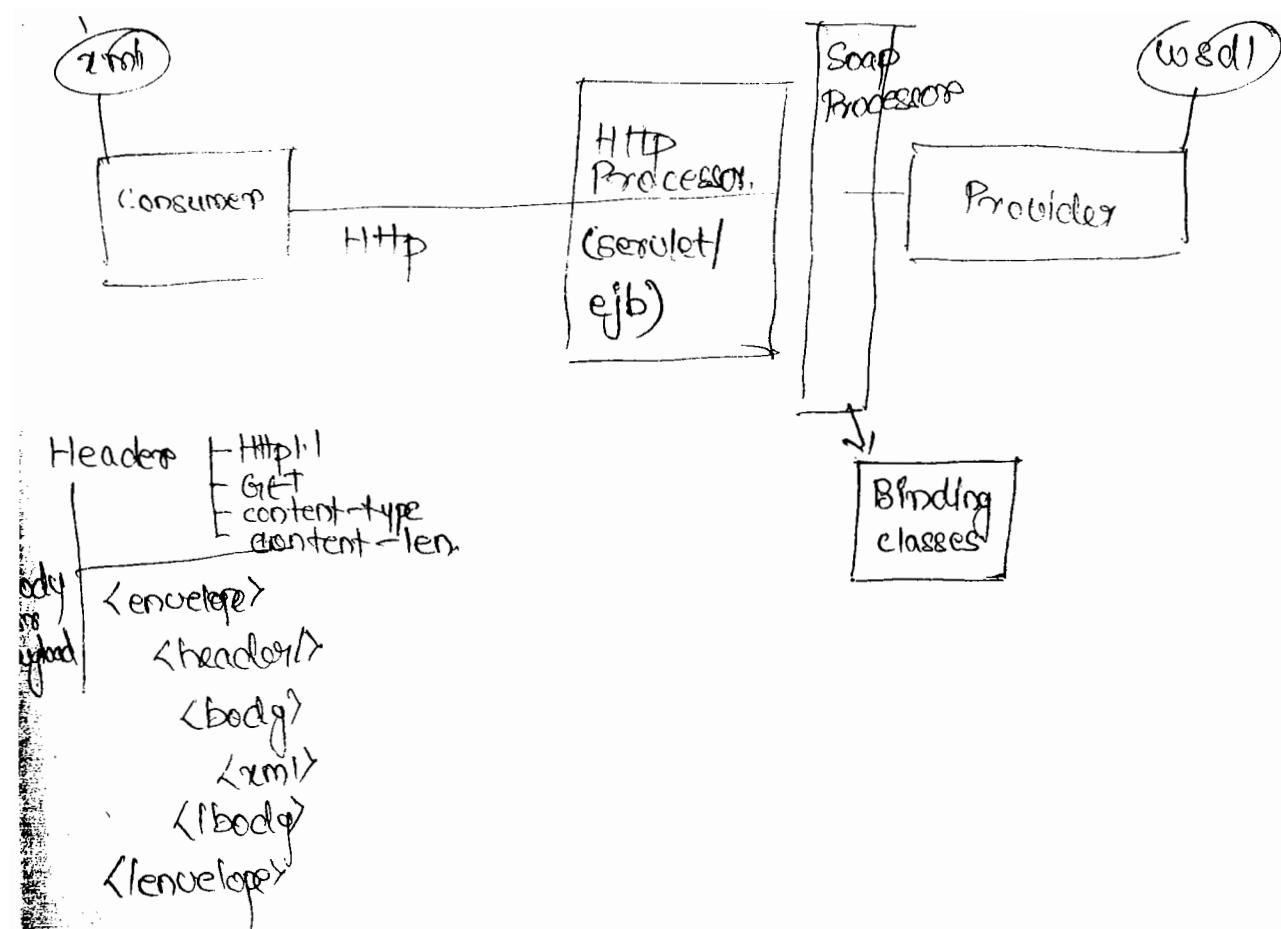
③ In your project where are you using Web Services?

- ↳ WSDL 1.0 specification or JAX-RPC API Apache Axis or
- ↳ BP 1.1 JAX-RPC or NS 1 + Impl ]

④ Which endpoint based web service you're developing?

    └ Servlet endpoint web service,

    └ EJB    "    "    "



Endpoint refers to provider. (target endpoint)

why people go for servlet as ~~not~~ end point rather than EJB.

EJB containers are commercial.

EJB is complex tech.

EJB are heavy weight comp.

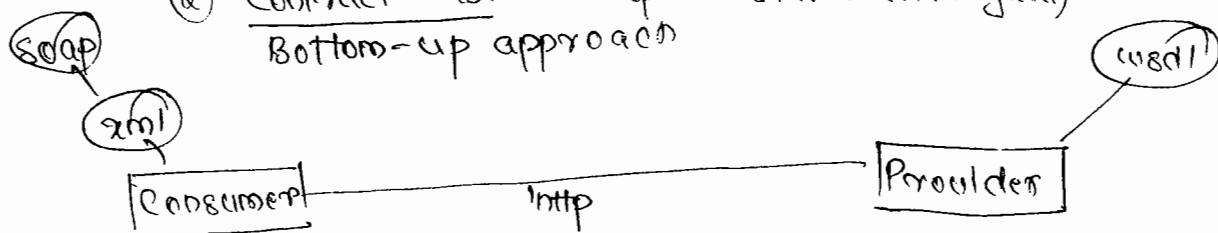
11-04-14

Friday

⑥ Which approach have you used in developing web service

(A) → ① Contract first, (development starts with wsdl)  
Top-down approach

② Contract last, (development starts with java)  
Bottom-up approach



→ wsdl acts as a contract b/w consumer & provider  
in web services.

⑦ Which message exchanging pattern you're using? (MEP)

(A) → How Info is exchanged b/w consumer & provider or  
how ecomm is done b/w consumer & provider.

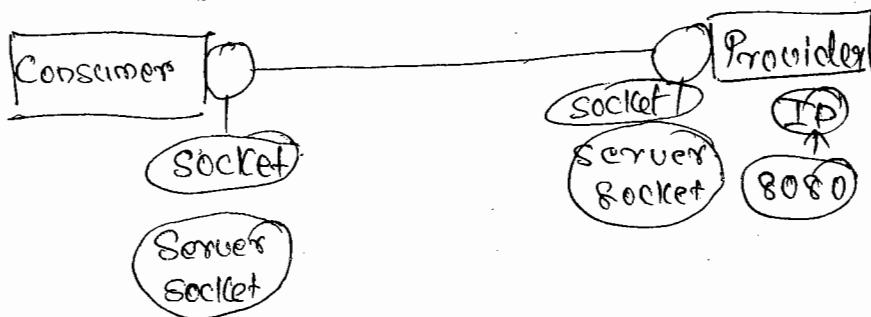
→ 3 MEP :-

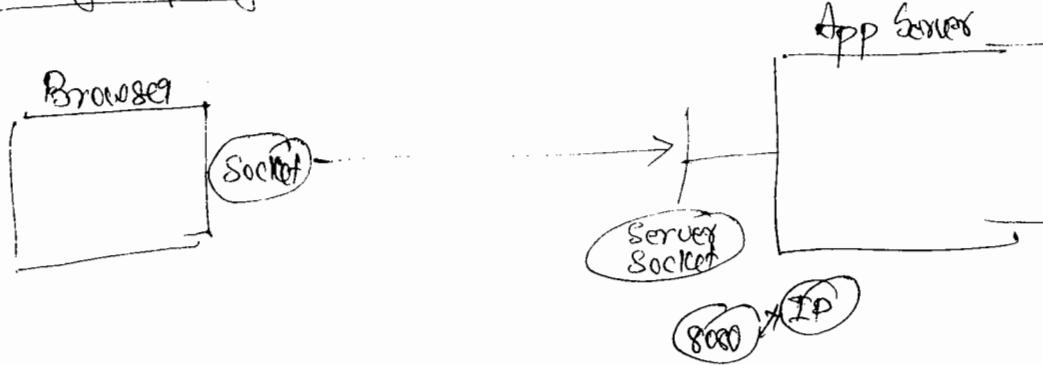
① Synchronous req|reply. [Consumer is blocked on req|reply]

② async req|reply or delayed response [non-blocking req|reply]  
Eg: Postman, Mobile Balance enquiry.

③ One way invoke or fire & forget. [DTH provider]

Sync. prog in case of Core Java





12-04-14  
Saturday

Q) What is Message Exchanging Format (MEF)?  
Used in developing web service?

A) → MEF is indicated by 2 people.

| Style    | use     | [Based on style & use we decide MEF] |
|----------|---------|--------------------------------------|
| rpc      | literal |                                      |
| document | encoded | [ 4 MEF ]                            |

→ Style represents web service method call.

Eg. Ticket reserve(Passenger pinfo, Travel tinfo){--}

<reserve>

<pinfo> --> </pinfo>

<tinfo> --> </tinfo>

</reserve>

Style = structure of xml      use = way the data is written in xml.

→ The way the data is being formatted to send from provider/consumer is MEF.

→ JAX-RPC by default uses rpc - encoded.

→ JAX-WS    "    "    "    document - literal.

Using Broulet end point using contract last way  
Sync rep reply using by default rpc connection

→ Service Endpoint Interface [SEI]. It acts as contract. In contract last approach development starts from java side. Interface acts as contract from java perspective.

→ There're rules to write SEI :-

① Must extend from Remote interface marker

Diff b/w local & SEI

methods declared in SEI are accessible remotely.

interface BookInfo extends Remote {

② Declare only web service methods

getBookPrice method shouldn't be final, static, private, may or mayn't return values & have parameters

Recommended to be Serializable.

Must declare throws RemoteException

float getBookPrice(String isbn) throws RemoteException;

③ Why I should throw RemoteException and why I should declare it to throw RemoteException ?

(A)

- consumer

```
String isbn;  
//connecting db  
if(isbn)  
try{  
    BookInfo bi = ...();  
    price = bi.getBookPrice(isbn);  
    connecting db  
    price + service tax  
vat  
insert into db.  
}  
catch(f){  
    if catch(SQLEception sqe){  
        }  
    catch(FileNotFoundException fe){  
        }  
    catch( RemoteException re){  
        }  
}
```

// Any exception other than RemoteException  
is due to code of consumer.  
B'coz provider methods throws only  
RemoteException only, then only consumer  
throws knows why which code is  
generating exceptions  
↑

④ Why SEI should throw  
only RemoteException ?

→ Provider should never process  
exception, it should always  
throw it back to consumer.

E.g : try {  
 catch(SQLEception sqe){  
 }  
 throw new RemoteException(sqe);  
} checked Exception

→ RemoteException is  
checked exception. It  
has to be handled.

So, consumer can catch it.

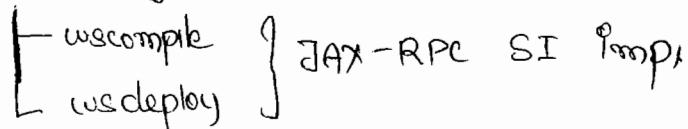
14-04-14  
Monday

→ In JAX-RPC SI impl we need to implement SEI.

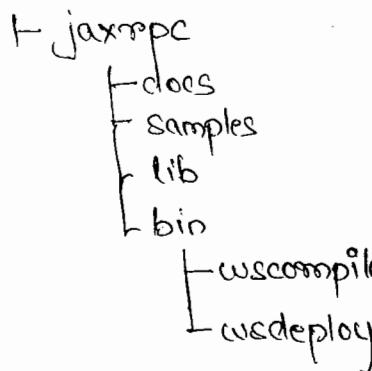
Step 1: Write SEI.

Step 2: Write impl class for SEI. { Only Biologic Methods may/mayn't throw Remote Exception.

Step 3: Create Binding classes.



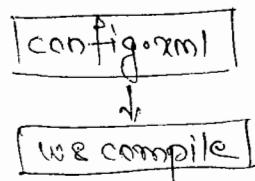
C:\Sun\jwsdp-2.0



18-04-14  
Tuesday

→ wscompile can't take SEI directly as input. We should pass config.xml.

config.xml



targetNamespace  
typeNamespace  
name  
interfaceName  
servantName  
packageName

attributes

Q) What is the development procedure you used for developing Web Service?

A) → Refer voice notes.

Consumer

```
--  
String isbn;  
//connecting db  
//isbn  
try {  
    BookInfo bi = ...();  
    price = bi.getBookPrice(isbn);  
    connecting db  
    price + service tax  
    vat  
    insert into db.  
}  
catch (Exception e) {  
    if (e instanceof SQLException) {  
        catch (SQLException sqle) {  
            }  
        catch (FileNotFoundException fe) {  
            }  
        catch (RemoteException re) {  
            }  
    }  
}
```

Any exception other than RemoteException  
is due to code of consumer.  
B'coz provider methods throws only  
RemoteException only, then only consumer  
thrusts know which code is  
generating exceptions  
↑

④ Why SEI should throw  
only RemoteException?

→ Provider should never process  
exception, it should always  
throw it back to consumer.

E.g.: try {  
 catch (SQLException sqle) {  
 throw new RemoteException(sqle);  
 }  
}

→ RemoteException is  
checked exception. It  
has to be handled.

So, consumer can catch it.

14-04-14  
Monday

→ In JAX-RPC SI impl we need to implement SEI.

Step 1: Write SEI.

Step 2: Write impl class for SEI. [Only Business logic  
Methods may/mayn't throw  
Remote Exception.]

Step 3: Create Binding classes.

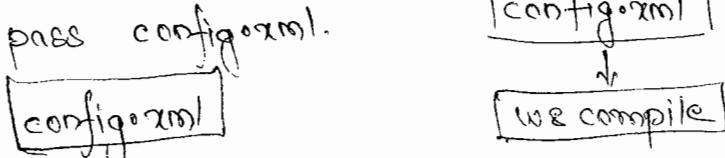
[ wscompile ] JAX-RPC SI Impl  
[ wsdeploy ]

c:\Sun\jwsdp-2.0

+ jaxrpc  
  + docs  
  + samples  
  + lib  
  + bin  
    + wscompile (generate binding classes from method)  
    + wsdeploy

15-04-14  
Tuesday

→ wscompile can't take SEI directly as input. We should  
pass config.xml.



targetNamespace  
typeNamespace  
name  
qName  
interfaceName  
servantName  
packageName

attributes

Q) What is the development procedure  
you used for developing Web Service?

A) → Refer voice notes.

→ config.xml should be placed in WEB-INF directory.

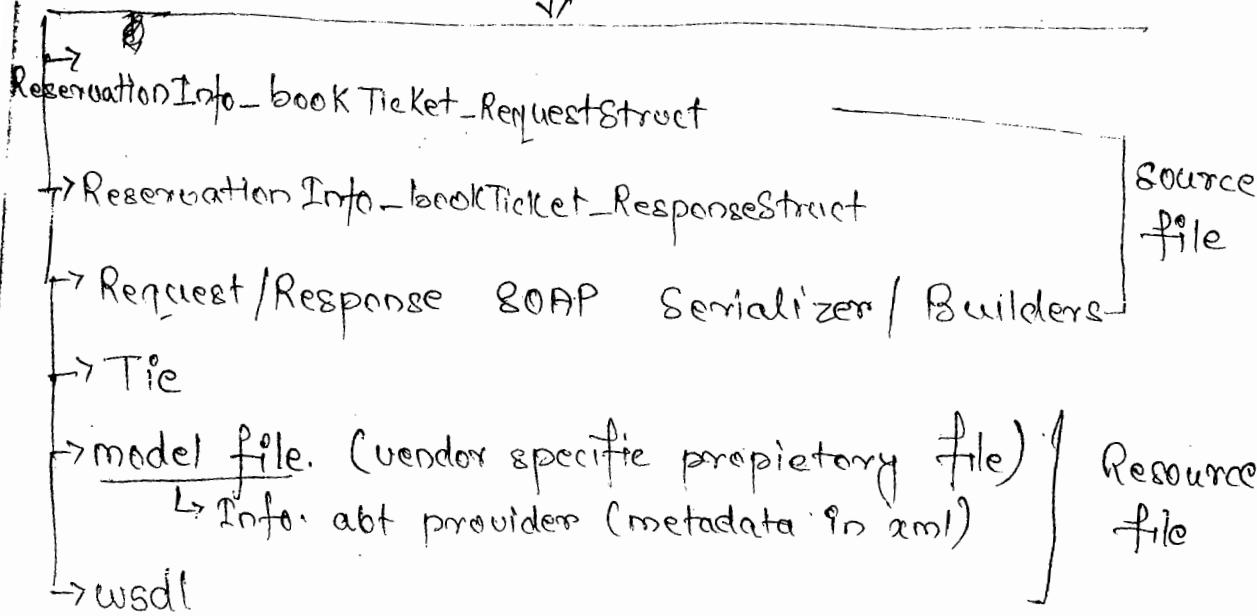
## ① How to run wscompile tool?

- A) → set path to bin. If pbm exists set path to stayed\bin.

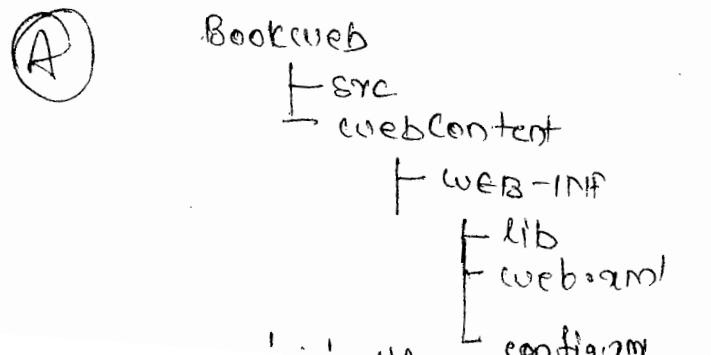
16-04-14  
Wednesday

→ we need to pass both SAI & impl to check for validity.

config.xml  
↓  
wscompile      ② What is gen when  
                  we run \$wscompile  
                  ↓



## ② How to run tool?



set path = %path%; c:\sun\jwsdpa\0\jaxrpc\bin

BookWeb:/> wscompile -el src -gen:server  
-ep build\classes -keep -verbose -model model-rpc-encxml  
-model model-rpc-encxml.g3 !webcontent\WEB-INF\config.xml.

↳ jwscompile takes .class file b'coz its free from compilation errors.

-gen:server → Generate provider stub classes

→ Drag & drop web & model in WEB-INF

Step 4: Map web service with URL pattern

17-04-14  
Thursday

jaxrpc-si.xml - vendor specific

Step 5: Run wsdeploy

wsdeploy -o target.war -verbose BookWeb.war  
↳ open using winzip (Shift + Right click)

Step 6: Copy paste cfg file into WEB-INF.

5-09-14

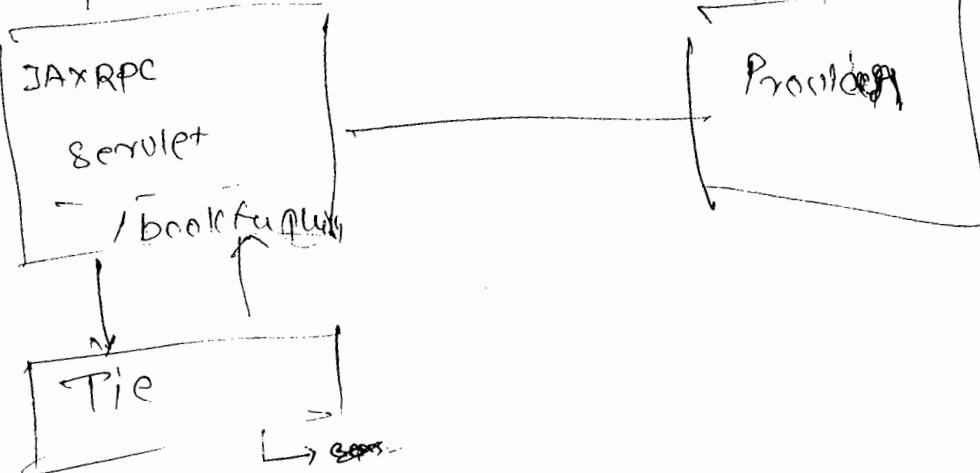
Friday

9-09-14

Saturday

## Request Processing

`http://localhost:8080/Bookweb/bookinfo`



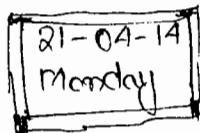
- JAXRPC servlet can't process the req, so it uses Tie class
- In model file we compile writes info. used by JAXRPC servlet to locate Tie class.

jaxrpc-min.xml                                  jaxrpc-ri-runtime.xml  
 minimal web service      Complete web service descriptor file.  
 descriptor file.

④ Tie  
 invoice-> getBookPrice (StreamingHandler state state)  
 Http Obj

⑨ Why are we running webservice tool?

A) So that we don't need to know client classes that are cfg'd in jaxws-ri-runtime.xml.

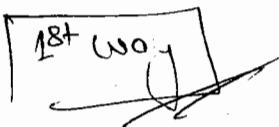


→ When you don't use model ~~you~~ the wrong file name will be generated.

→ Sometimes it's called skeleton/helper.

⑩ Suppose there're 2 methods but how does the servlet ~~knows~~ knows which method to call?

A) There're total 3 ways:



→ JAX-RPC API specific technique.

→ Request-to Method Resolution

Consumer code goes to wsdl & tries to find method using operation code. (ops code = order of declaration starts with 0)      Httpref header  
[opcode Value].

→ Opcode value is resolved from wsdl document.

# WSDL | Web Service Description Language

<definitions> ← root element  
 types  
 messages  
 portType  
 binding  
 service

} sections of wSDL

Interface TrainInfo  
 } should be in complexType so that consumer  
 can call methods -  
 Ticket bookTicket(PassengerInfo pInfo, JourneyInfo jInfo)

<?xml version="1.0" encoding="utf-8" ?>  
 <definitions xmlns:tns = "http://traintech.com/multirequest/types" >

<types> → Represent parameter types & return types of  
 web service methods. 1 per param, 1 per return type  
 (separated) → called TypeNamespace

<xsi:schema targetNamespace = "http://traintech.com/multirequest/types">

<xsi:complexType name = "PassengerInfo">

<xsi:sequence>

<xsi:element name = "ssn" type = "xsd:string"/>

</xsi:sequence>

</xsi:complexType>

</types>

you can import external xsd here

} inline xsd

## Messages

one single input & one single output (a msg)

```
<message name = "TrainInfo-bookTicket">
```

```
  <part name = "pInfo" type = "xsd:string" />  
    ↳ "int : PassengerInfo"
```

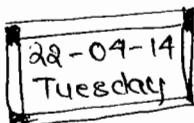
```
  <part name = "jInfo" type = "xsd:int" />  
    ↳ from types section
```

```
</message>
```

```
<message name = "TrainInfo-bookTicket Response">
```

```
  <part name = "result" type = "xsd:int" />
```

```
</message>
```



portType : Represents SEI interface.

Skeleton structure of service

what is there in service

```
<portType name = "SEI-Interface-Name"> [ "bookInfo" ]
```

parameterOrder = "pInfo jInfo"

```
  <operation name = "method-Name"> [ "getBookPrice" ]
```

```
    <input message = "BookInput: TrainInfo-bookTicket" />
```

```
    <output message = "BookOutput: TrainInfo-bookTicketResponse" />
```

```
  </operation>
```

```
</portType>
```

targetNamespace = "http://intra.co.in/rail/reservation/facil" } In model  
and ns:input = " " ↳ level

What is an abstract wsdl & what is an

concrete wsdl?

→ doesn't provide details using which  
protocol we need to access service

types, messages, portType section.

spec of  
binding protocol

abstract wsdl doc. ↳ No format info  
about input.

types, messages, portType, binding, service section

concrete wsdl doc

Abstract

F.

Learn

Spe

bindings

- How to access particular service info.
- Tells which binding protocol to use [SOAP]
- Tells abt. transport protocol & MGF.

Cl

<binding name="TrainInfoSOAPBinding" [seI Binding/SEISoAPBinding]  
    type = "tns:TrainInfo" > ["portType name"]

<soap:binding transport = "----soap/http" ↪  
    style = "rpc" />

<operation name = "bookTicket">

PC

<input>

<soap:body use = "encoded" />

<input>

<output>

<soap:body use = "encoded" />

<output>

</operation>

</binding>

> <soap:operation soapAction = "http://traininfo.in/rail/reservation/wSDL#BookTicket" />

} Input & Output go  
& see in portType.  
Here, we specify  
how to send/receive

23-09-14  
Wednesday

## Service

It's not actual webservice  
Factory for creating obj for post  
(How to get service to access)  
(Where is the service located)

```
<service> name="TrainInfoService"
  <port name="TrainInfoSOAPPort"
    > Actual web service
      binding="tns:TrainInfoSOAPBinding"
      <soap:address location="url">
        </port>
</service>
```

## Contract First Approach

interface Insurance extends Remote {  
public MembershipId enroll(MemberInfo mInfo, PolicyInfo pInfo);  
}  
wsdl  
<?xml version="1.0" encoding="utf-8" ?>  
<definitions>  
<types>  
<xss:schema targetNamespace="http://lic.org/insurance/types">  
<xss:complexType name="MemberInfo">  
</xss:complexType>

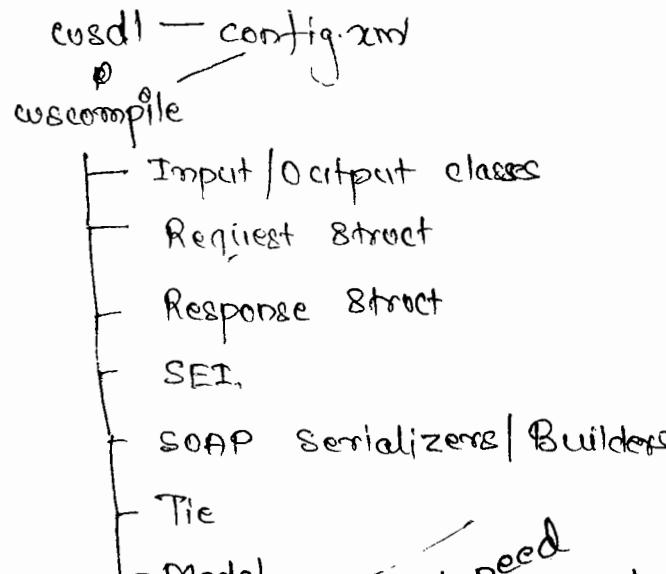
24-04-14

Thursday

## Interface-Generator

Step 1: Write wsdl.

Step 2: Run wscompile tool.



*we just need  
to write impl for SEI.*

config.xml in WEB-INF

sun\jwsdp\Jaxme\samples\HelloWorld\etc

wscompile -d sun -gen:server -keep -verbose  
-model model-rpc-en.xmlogz WebContent\WEB-INF\config.xml

25-04-14  
Friday

Step 3: Write implementation class.

Step 4: Write jaxrpc-ri.xml.

Step 5: Export war file.

Step 6: Run wdddeploy tool.

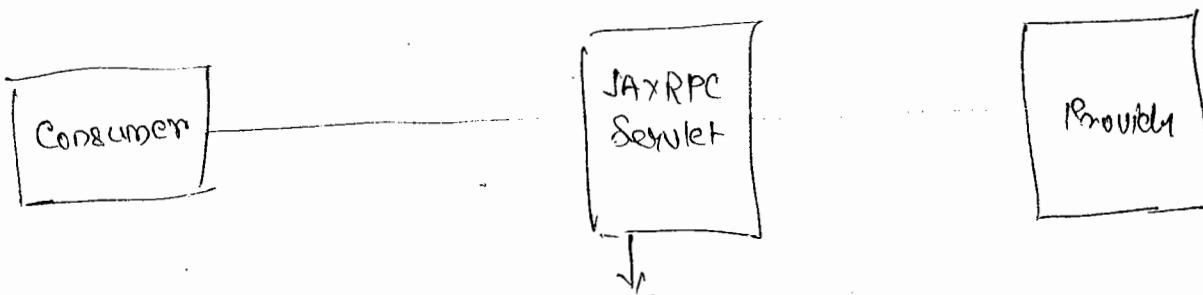
Step 7: Add jaxrpc-ri-runtime.xml & web.xml  
to WEB-INF folder of project

Step 8: Use SOAP to access service.

Purpose of soapaction in <binding>

http://localhost:8080/PolicyWeb/Insurance ] endpoint url

→ Another way of resolving methods of a service is  
using soapaction.



Works with JAXRPC RI - Looks for opcode first.

WSS-I recommended - Looks for soapaction second.

⑩ Did you face any pblm working with Web Service?

⑪ Have you worked on cross platform services?  
How?

26-04-14

Saturday

→ Concrete wsdl be available at runtime only

wsdl shooting

from all

from drive

from CD

url to dynamic req. for wsdl. (JAX-RPC 2.0)

Dynamic wsdl generation

→ we place wsdl in WEB-INF b'coz it's abstract.

wsdl → wsdl Handler

⑩ Have you developed contract-first or contract-last?

⑪ You can say both.

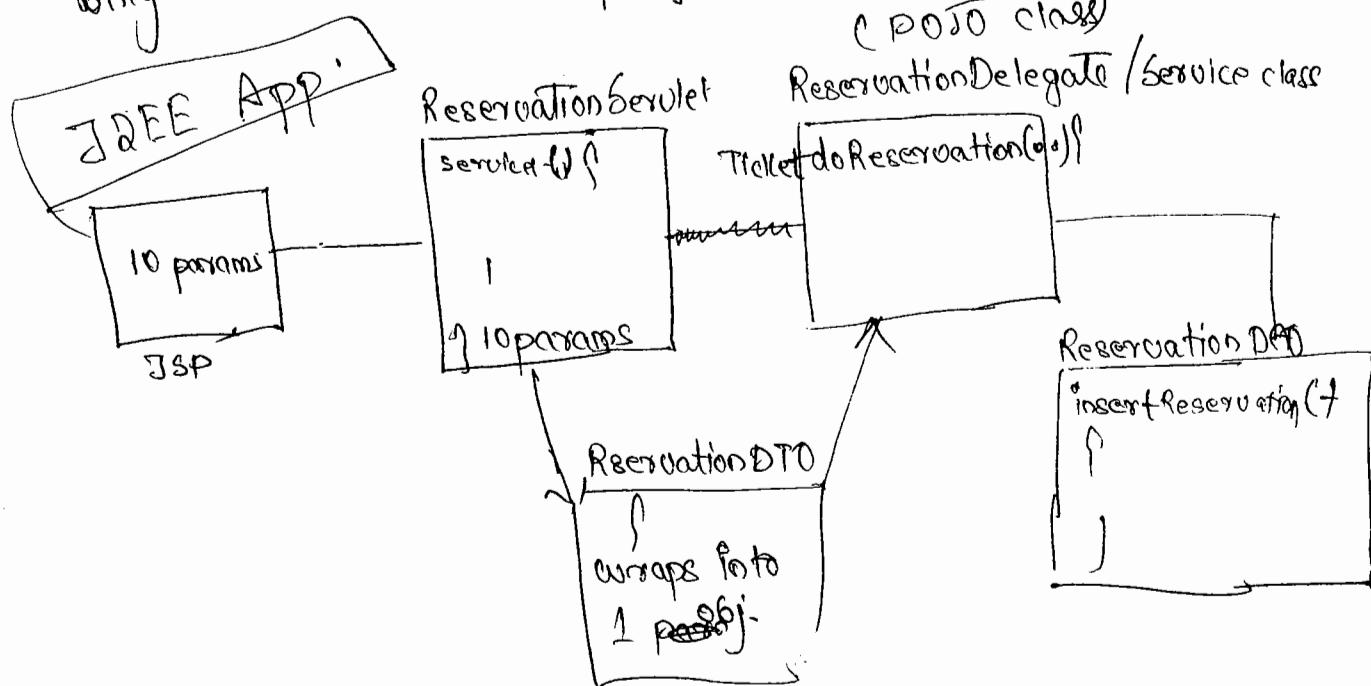
Contract first → Explain @ wsdl & xsd, namespace.

Contract last →

⑩ Can you explain your project architecture &  
which classes have u exposed as web services.

## Project Architecture / Project Flow

→ Mostly in existing project some ~~are~~ some services are highlighted as web services rather than creating only web service ~~part~~ project.



→ Whenever your project is up & running you need to use contract - last approach.

④ When to use contract first?

Ⓐ Project is fresh or service is new one that will be provided.

→ For web service there is web service Req'd doc. given by client to & used by B-analyst.

B-analyst based on the reqd. will analyze what will be input & output.

E.g. Search Service in Flipkart.

B-analyst consults with Tech. Architect for feasibility negotiation & then

and then, HL use case, Low Level use case,

abstract web design / functional doc will be written  
MS Word doc

→ Develop web services.

### Developing Consumer

→ JAX-RPC API provides set of classes to develop provider and consumer.

#### JAX-RPC API

use  
(70%)   | Stub based consumer

(20%)   | Dynamic Proxy (DP)

(10%)   | Dynamic Invocation Interface (DII)

3  
} Types of  
Consumers

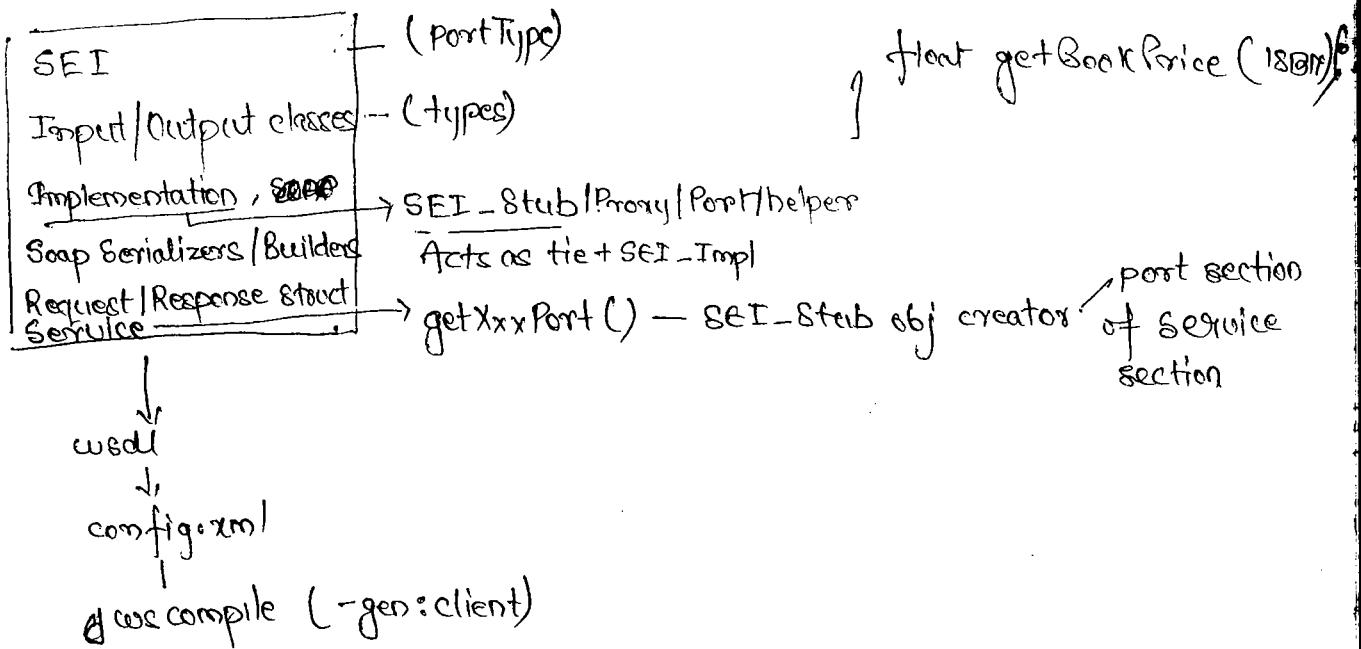
27-04-14  
Sunday

Helper class to talk to provider

## Stub Based Endpoints



interface BookInfo {



```
SEI sei = new SEI-Stub(); // getXXXPort();
```

```
sei float price = sei.getBookPrice ("ISBN1001");
```

→ Stub class is internal class so, we don't know the name.  
So, service class is generated by wecompile tool to  
create stub class obj.

~~Service service = new Service();~~

- Put jars in lib → deployment assembly
- config.xml with wsdl
- wscompile -d src -gen:client -keep
  - verbose ~~listens~~ config/config.xml
- Create a new ~~per~~ package.

public class BookInfoClientTest {

    @Svr (-)throws ServiceException, RemoteException {

        BookInfoService service = new BookInfoServiceImpl();

        BookInfo port = service.getBookInfoPort();

        float price = port.getBookPrice("ISBN1001");

        System.out.println(price);

}

## PurchaseOrder.wsdl

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

```
<wsdl:definitions
```

```
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
```

```
    xmlns:tns="http://ebay.in/sales/wsdl"
```

```
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
```

```
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
    name="PurchaseOrderService"
```

```
    targetNamespace="http://ebay.in/sales/wsdl"
```

```
    xmlns:et="http://ebay.in/sales/types">
```

```
<wsdl:types>
```

```
    <xs:schema targetNamespace="http://ebay.in/sales/types"
```

```
        elementFormDefault="qualified">
```

```
        <xs:complexType name="purchaseOrderType">
```

input parameter

```
            <xs:sequence>
```

```
                <xs:element name="itemCode" type="xs:string"/>
```

```
                <xs:element name="quantity" type="xs:int" default="1"/>
```

```
            </xs:sequence>
```

```
        </xs:complexType>
```

```
        <xs:complexType name="orderStatusType"
```

return value

```
            <xs:sequence>
```

```
                <xs:element name="orderId" type="xs:string"/>
```

```
                <xs:element name="status" type="xs:int" default="1"/>
```

```
            </xs:sequence>
```

```
        </xs:complexType>
```

```
    </xs:schema>
```

```
<wsdl:message name="Order-placeOrder">
  <wsdl:part name="purchaseOrder" type="et:purchaseOrderType"/>
</wsdl:message>          Order-placeOrder-ResponseStruct
<wsdl:message name="Order-placeOrderResponse">
  <wsdl:part name="result" type="et:orderStatusType"/>
</wsdl:message>
```

11

&lt;/w

```
      ↗ SEI Interface name
<wsdl:portType name="Order">           ↗ SEI method name
  <wsdl:operation name="placeOrder">
    <wsdl:input message="tns: Order-placeOrder" />   ↗ from <wsdl:message>
    <wsdl:output message="tns: Order-placeOrder Response" />   ↗ for req & res
  </wsdl:operation>
</wsdl:portType>
```

— c1

```
      ↗ PortType name
<wsdl:binding name="Order SoapBinding" type="tns:Order">
  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="placeOrder">           ↗ SEI method
  <soap:operation soapaction="http://ebay.in/sales/wsdl/placeOrder" />   ↗ To identify method
  <wsdl:input>                                (method resolution)
    <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" 
      namespace="http://ebay.in/sales/wsdl" use="encoded" />
  </wsdl:input>
  <wsdl:output>
    <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" 
      namespace="http://ebay.in/sales/wsdl" use="encoded" />
  </wsdl:output>
</wsdl:operation>
```

1

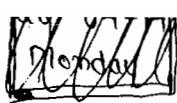
```
<wsdl:service name="OrderService">  
  <wsdl:port binding="tns:OrderSOAPBinding" name="OrderSOAPPort">  
    <soap:address location="REPLACE-WITH-ACTUAL-URL"/>  
  </wsdl:port>  
</wsdl:service>  
</wsdl:definition>
```

Classes Generated by JAXRPC SI

usage)  
package com.store.service;  
public interface Order extends javax.rmi.Remote {  
 public com.store.service.OrderStatusType  
 <sup>return type</sup>  
 placeOrder(com.store.service.PurchaseOrderType purchaseOrder)  
 <sup>parameter</sup>  
 throws javax.rmi.RemoteException;

[config.xml] → Input for wscompile tool

```
<?xml version="1.0" encoding="UTF-8"?>  
<configuration  
  xmlns="http://java.sun.com/xml/ns/jax-rpc/rmi/config">  
  <wsdl  
    location="WebContent/WEB-INF/PurchaseOrders.wsdl"  
    packageName="com.store.service" />  
  </wsdl>  
</configuration>
```



Mo

Req

Order-placeOrder-RequestStruct

- default constructor
- param constructors
- getters
- setters

Order-placeOrder-ResponseStruct

- default constructor
- param constructors
- getters
- setters

→  
2  
→  
→  
→  
→

Front

interf...xx

1.  $\text{order}^B = \text{order}^A$ 

Thay

order = order

order = order

Thay

cotton

28-04-14  
Monday

## Dynamic Proxy Based Consumers (DP)

- Implementation class for SEI will be generated at runtime during execution of program
- we don't need tools.
- We have to use low level API of Jax-RPC.

~~Step~~ → Write I/O classes

→ write SEI

→ Impl of SEI

→ ServiceFactory

→ javax.xml.rpc.Service (JAX-RPC API) (I)  
└ getPort();

ServiceFactory factory = ServiceFactory.newInstance();

Service bookInfoService = factory.createService("bookInfoService");  
└ (new QName(tns, "bookInfoService"))

bookInfoService.getPort("BookInfoPort");

↓      └ Port refers to bindings. To know the MEF.

(new QName(tns, "BookInfoPort"));

└ To identify SEI.

<service name="xservice">

  <port name="p" binding="" />

</service>

<service name="yservice">

  <port name="p" binding="" />

sfactory.createService(wsd1.getLocation(), new QName(tns, "bookInfoService"));

SEI sei = bookInfoService.getPort(new QName(tns, "BookInfoPort"),  
SEI.class);

public interface BookInfo extends Remote {

} float getBookPrice(String String-1) throws RemoteException;

public class DPCClientTest {

private static final String WSDL\_URL = "";  
" " " " TARGET-NAMESPACE = "";  
" " " " SERVICE-NM = "";  
" " " " PORT-NM = "";

psum(-) {

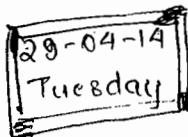
Servicefactory sfactory - - -

## Stub Based

- Stub created at development time → Stub created at runtime
- Tools reqd.
- Vendor specific classes are generated b'coz of tools.
- No such impact.

## DP

- No tools reqd.  
(No Infrastructure/vendor specific tools)
- API based classes
- It has to decode model & generate a class to run & hence performance impact.



## (DII) Dynamic Invocation Interface Based Consumed

- Most complicated consumer.
- Rare chances of working.
- Calling a method dynamically on an interface.

```
class A{  
    void m1();  
}  
A a = new A();  
a.m1();
```

compile  
time  
resolution  
of  
method  
call.

→ We need to call method &  
class of method at runtime  
taking input at runtime.

↳ JAX-RPC

```
public void invoke(String className, String methodName)
```

```
Object obj = Class.forName(className).newInstance();
```

```
Method m = new Method(methodName);
```

```
m.invoke(obj, args);
```

|  
|      ↳ Invoking method on obj.  
|  
|      obj isn't invoking method.

}  
} Reflection API

→ we're creating a method to be invoked on an

obj.

→ Provider & method will be decided at runtime.

→ We don't need SEI b'coz we're not fixing it.

→ Only webd is enough.

→ No generation of binding classes as interface is not fixed.

→ No vendor specific tools.

→ JAX-RPC API classes are reqd.

→ As port isn't there marshalling & unmarshalling we have to have

to do by ourself only.

↳ JAX-RPC API interface

→ Call → class that represents methods to be called  
on a provider.

```
↳ invoke(args);
```

Servicefactory

Service (I)

call (I) → represents method of any provider chosen at runtime.  
— — invoke(args);

Servicefactory factory = ServiceFactory.newInstance()

Service bookInfoService = factory.createService(new QName(TGT\_NNSP,  
"IBookInfoService"))

Call e call\_getBookPrice = bookInfoService.createCall(new QName(TGT\_NNSP,  
"BookInfoPort"));

e call\_getBookPrice.setOperationName(new QName(TGT\_NNSP, "getBookPrice"));

sfactory.createService (wsdlLoc, QNameService); } Overloaded  
sfactory.createService (QNameService); forms

↳ we don't need bindings so  
we don't need wsdlLoc here

createCall();

createCall (QName);

createCall (QName, QName); } Overloaded forms

01-05-14  
Thursday

→ We will use low level JAX-RPC API.

→ Method on an obj is called.

call.getBookPrice.setTargetAddress("new QName  
http://localhost:5050/- - - ");

→ create Call points to method.

call.getBookPrice.addParameter("String-1", "xs:string",  
ParameterMode.IN);

call.getBookPrice.setReturnType("xs:float");

call.getBookPrice.setProperty("use-SOAPAction", true);

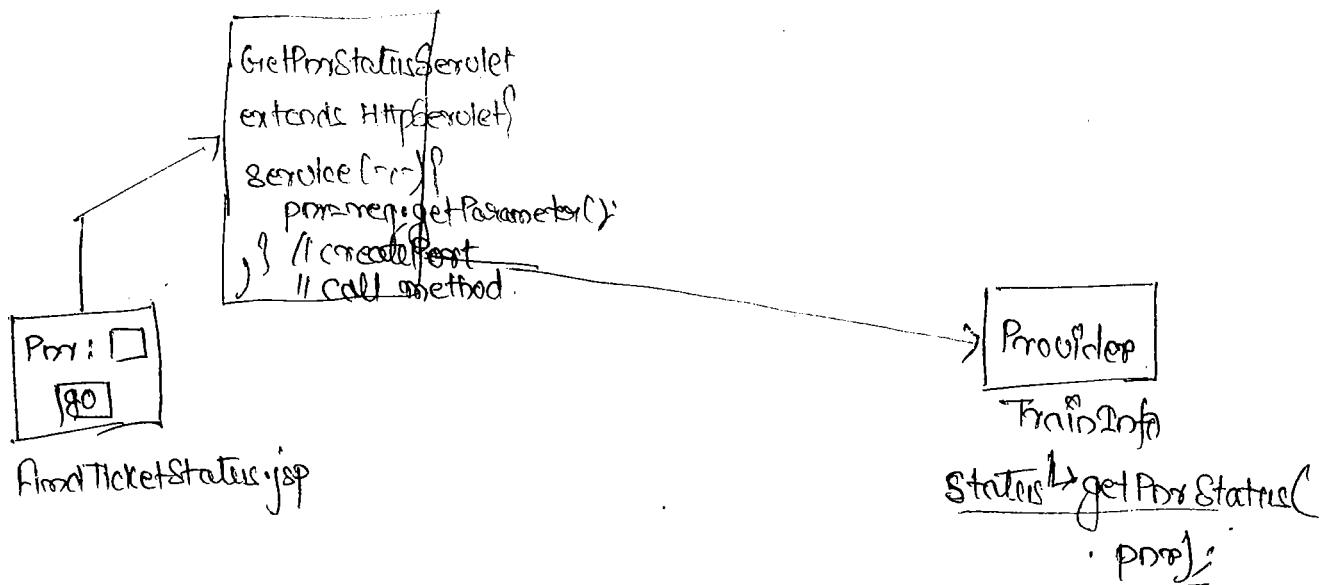
call.getBookPrice.setProperty("SOAPAction", "");

call.getBookPrice.setProperty("encoding-style-uri", "NMSPE");

float price =

call.getBookPrice.invoke(new Object[] {"ISBN1001"});

# Architecture of Consumer Side Project



- rule should never place binding classes under 'src' folder bcoz it's not written by developer & also we have to maintain in Source Control Version (SCM) so, we should keep jar files of binding classes.
- If we're keeping source code then we have to track the versioning also. We have to compile them. So, package it into jar & place it into classpath.
- Never at consumer side we place src files we use jars.

02-05-14  
Friday

(1) S

Completely written in Java.

~~Apache Axis~~

→ JAX-RPC API impl.

→ Provider + Consumer can be developed.

→ Apache Extensible Interaction System

→ Initially called Apache SOAP.

→ Can work with FTP, SMTP, HTTP . . . protocols

→ Internally uses SOAP engine. \*

→ Convenient tools support.

→ SOAP monitor ~~tool~~ Applet tool to monitor msgs

exchanged b/w consumer & producer.

→ Open source impl for JAX-RPC API. (First one)

→ It's impl of SOAP so called as SOAP Engine.

→ Flexible Handlers.

→ C++ support was planned but never implemented.

How to work with Apache Axis

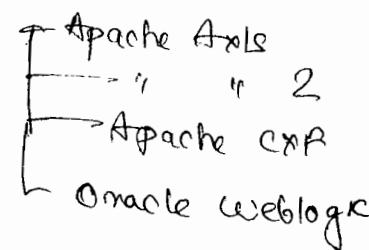
→ Apache Axis ships one zip file (binary distribution)

having jars & tools.

→ current ver 1.4.1  
No future release

Deployment Jar Bundles → ~~axis-bin-1.4~~  
axis-bin-1.4

(2)



(1)

etc

8en

## ① Setup Development Environment

→ Keep tools at a place to run. e.g. JDK + jars

## ② Setup Web Project

→ Normal web project + additional cfg files.

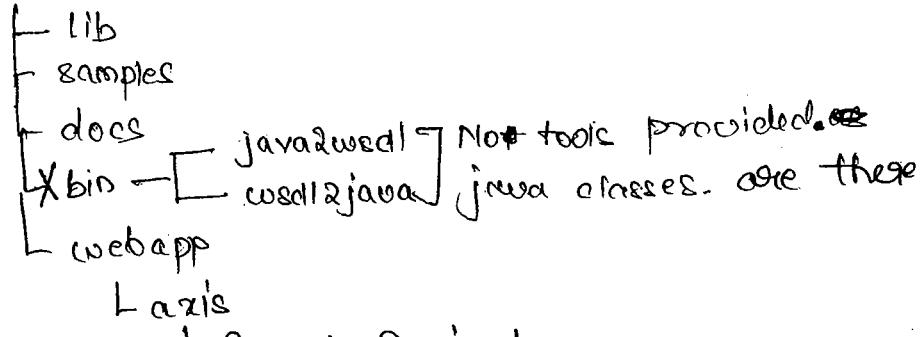
### ① Setup Dev Env

ⓐ Copy zip & paste into C:\.

ⓑ Extract into C:\ directly. [C:\axis-1.4]. Delete zip file.

ⓒ Create lib folder & paste activation.jar & mail.jar.

~~extracts~~ C:\axis-1.4



Run JWedp/ → activation.jar, mail.jar

3-05-14

Saturday

e  
→ Apa

cfg files in sample project is reqd. to develop

→ S

Axis web services.

New → Dynamic Web Project → TrainWeb → next →

Cal

→ Cal

Finish → Copy jars not properties file from

U  
↓  
↓

Axis-1-9\lib & paste in project.

→ -

→ Go to sample <sup>proj</sup> → copy all except WEB-INF &

• java file to "WebContent" folder.

→ copy .java file to src folder.



→ 1

→ Go to WEB-INF users.list and web.xml to project

WEB-INF.

→ Go to classes folder copy i18n.properties, i18n-javaproperties

copy to src.

[http://localhost:5050/TrainWeb]

Happy Home Page

Validator → Validates if the resources of project  
can build Axis web services

## List

→ Apache Axis provide two default web services :

- Admin Service (webd)

- Version (webd)

→ Drop at project.

## Call

→ calls a web service with no input.

## Visit

→ Apache Axis Homepage.

→ Therefore two types of web services in Apache Axis

- + JWS style of web services (Easy)

- POJO web service (Standard)

JWS Style (Java Web Service)

→ Create java class in default package only

```
public class TrainStatus
```

```
    public String getPNRStatus(String pnr){}
```

```
        return "PNR ---";
```

] Should be free  
of compilation  
error.

→ Development is over.

→ Drag & Drop this Java file into ~~the workspace~~ → ✓

F2 → Rename to TireInStatus.jws.

→ Ver

http://localhost:8080/BreadWeb/TireInStatus.jws?wsdl. ↴

500 Internal Server

Windows → Preferences → Installed JARs → Click on

directory → point to IdIC. → ok.

→ IDE points to IdIC now → Restart Server

→ Click wsdl

→ Open SoapUI ✓ Done

→ In this approach parameters can only be  
Primitives

→ Namespace concept is also not there

① ✓

→ It expresses Java class as web services

② ↗

→ Good for developing dummy service.

③ ↗

## POJO Web Service

→ Very strong support is there in Apache Axis for contract first approach.

interface TrainInfo {

~~Book~~ Ticket bookTicket (PassengerInfo pInfo, JourneyInfo jInfo)

}

→ Write wsdl first.

→ By looking at portType & messages we can diff. blo

forget & fire & Async/Sync

→ Generate Binding classes.

JAX-RPC 8I (Contract first) Apache Axis

① Create proj

① Create & config.

② Write wsdl

② wsdl.

③ Gen Binding classes.

③ Binding classes

  |  
  | wscompile  
  |  
  | redeploy.

  |  
  | wsdl2java (contract first)  
  | java2wsdl (contract last)  
  | AdminClient.

→ In Apache Axis we don't need vendor specific tools.

cfg file. b'coz there're two separate tools.

05-06-14

Monday

### How to run wsdl2java

javar w...

java org.apache.axis.wsdl.WSDL2Java

Go to eclipse F12 or 1+Shift+T → type WSDL2Java →

You can see package name → See at Bottom you will

find jar file name.

c:\axis-1.4\lib\axis.jar

class uses all jars  
of lib

java -cp path-to-axis.jar org.apache.axis.wsdl.WSDL2Java

append other jars using ;)

→ Better to create an environment variable.

Set My\_VAR = c:\axis-1.4\lib\axis.jar

echo %My\_VAR%

`java -cp %MY_VAR% org.apache.axis.wsdl.WSDL2Java`

axisenv.bat → create in C & delete only.

cmd → edit axisenv.bat → file save → file exit.

→ Right click on file → Open with notepad.

`set AXIS_HOME = C:\axis-1.4\lib`

No spaces ←

`set AXIS_CP = %AXIS_HOME%\activation.jar;%AXIS_HOME%\axis-ant.jar;`

- - -

→ close bat file

C:\> axisenv.bat

① Create env var pointing to all jars in lib directory.  
(AXIS-CP)

② `java %AXIS-CP% org.apache.axis.wsdl.WSDL2Java`



Switches that has to be passed as input while  
running tool

-v → verbose

-s or --server-side

-S or --skeletonDeploy <args> [true/false]

-N or --NSToPKg <arg>=<value>

-O Svc

wsdl

Q) What will be generated if we run this tool?

A) → Input / Output

→ SEI

→ RequestStruct / ResponseStruct

→ SOAP Serializers / Builders

→ Tie.

Internally used. So,  
these are also not there  
as no model file is  
there that has the info  
about generated classes.

→ Every Input / Output class will have Serialization &

Deserialization.

WSDL2Java

Input / Output

- SEI

- SOAPSerializers

- SOAPDeserializers

- SEI Impl

- deploy.wsdd

- undeploy.wsdd

Generic classes. Inbuilt with axis2.

config file (Autogenerated).

wsdd → web service Deployment

descriptor.

Generated files

Input / Output

SEI

SEI Impl

deploy / undeploy (.wsdd)

com files

After running it at command

C:\Project-Directory>java -cp "%AXIS-CP%\org\apache\axis.wadl  
WSPL2Java -D sync -V --server-side  
--skeletonDeploy false webContent\WEB-INF\trainInfo

06-05-14  
Tuesday

→ Stub here uses DII. But, here we're working with provider. So, we should delete it.

→ Write Impl class.

~~Dynamic Deployment Model~~

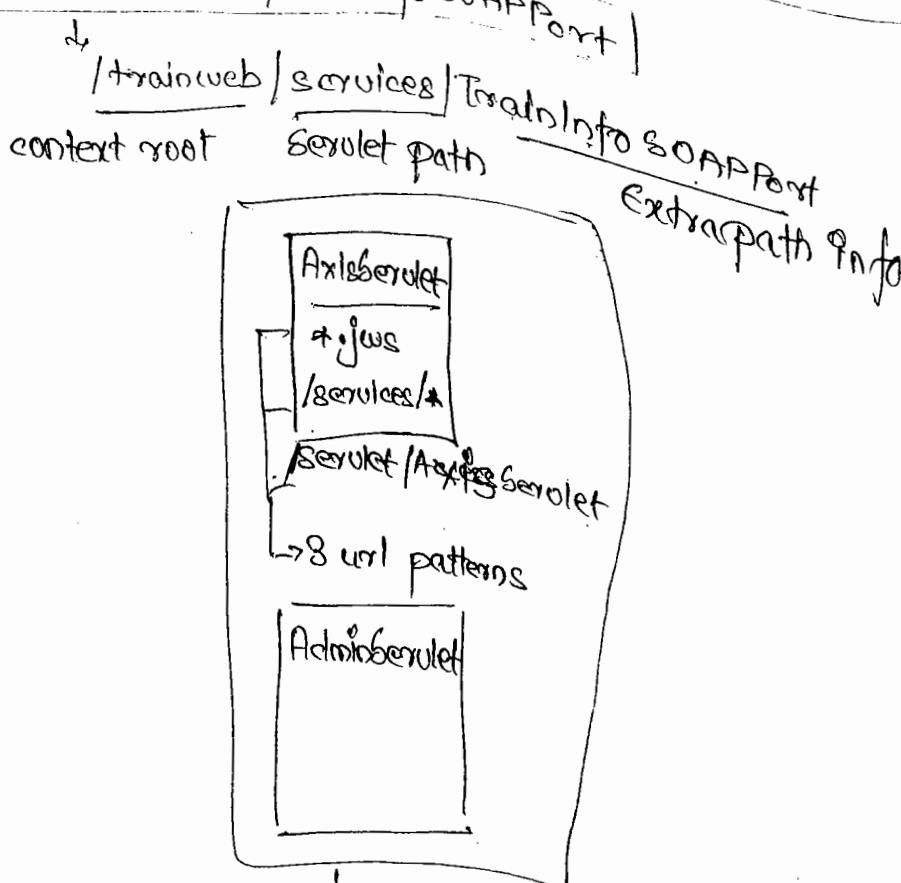
→ In JAXRPC SI if we want to turn off a service we have to modify jaxrpc-runtime.xml, comment url in web.xml after stopping server.

→ In Apache Axis we can independently activate & retire any service at runtime

contd—

Req. Processing flow of Apache Axis based Web Services

→ Web



Server

defa

our e

clepla

we

Di

→ deploy.wsdd server-config.wsdd

→ Request comes to AxisServlet, it looks to act as tie. deploy.wsdd file to identify impl class <sup>serializers</sup>.

& deserializers as there is no Tie.

→ deploy.wsdd contains only specific service partial web service deployment descriptor file developed by us. It's service specific cfg file.

→ server-config.wsdd is complete web service deployment descriptor file. It will have all the service info.

" "

Axis

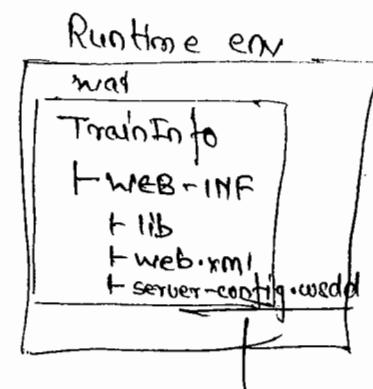
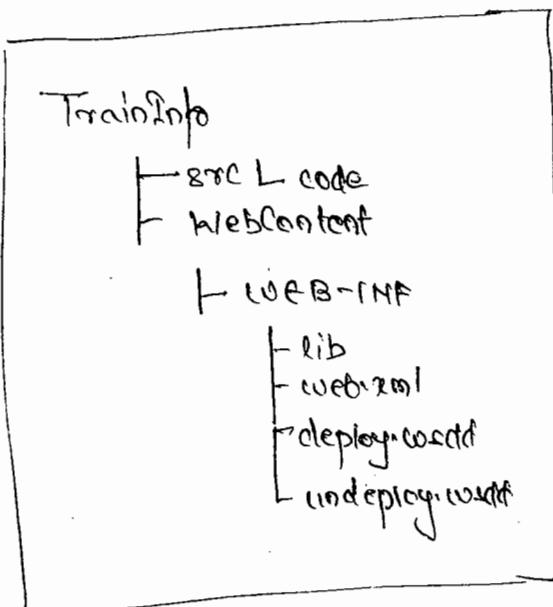
use

Ar

it

→ whenever server starts AdminServlet checks for server-config.xml, if it's not found so it creates a default cfg with AdminService & WebService. For our specific service new AdminServlet checks in deploy.wudd & writes that info into server-config.xml. We have to append it.

### Development env



"AxisServlet" will now get the request from AdminServlet. OR AdminTool will be used to pass the deploy.wudd to AxisServlet.

As, AxisServlet can see & modify server-config.wudd, it will append the our service cfg to server-config.wudd.

java -cp %.AXIS-CP%. org.apache.axis.client.AdminClient

-l http://localhost:5050/Tranxweb/servlet/AxisServlet

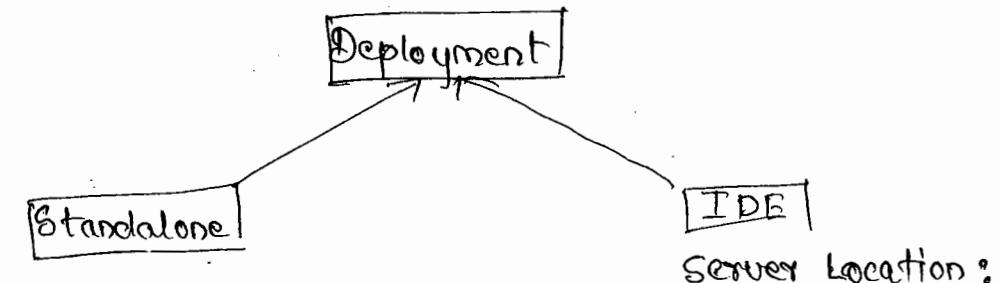
WebContent\WEB-INF\deploy.wadd

→ To retire a service we need to use

undeploy.wadd.

- └ Activating service.
- └ Retiring service.

bou  
sm  
dee  
de



tomcat\webapps

Server Location:

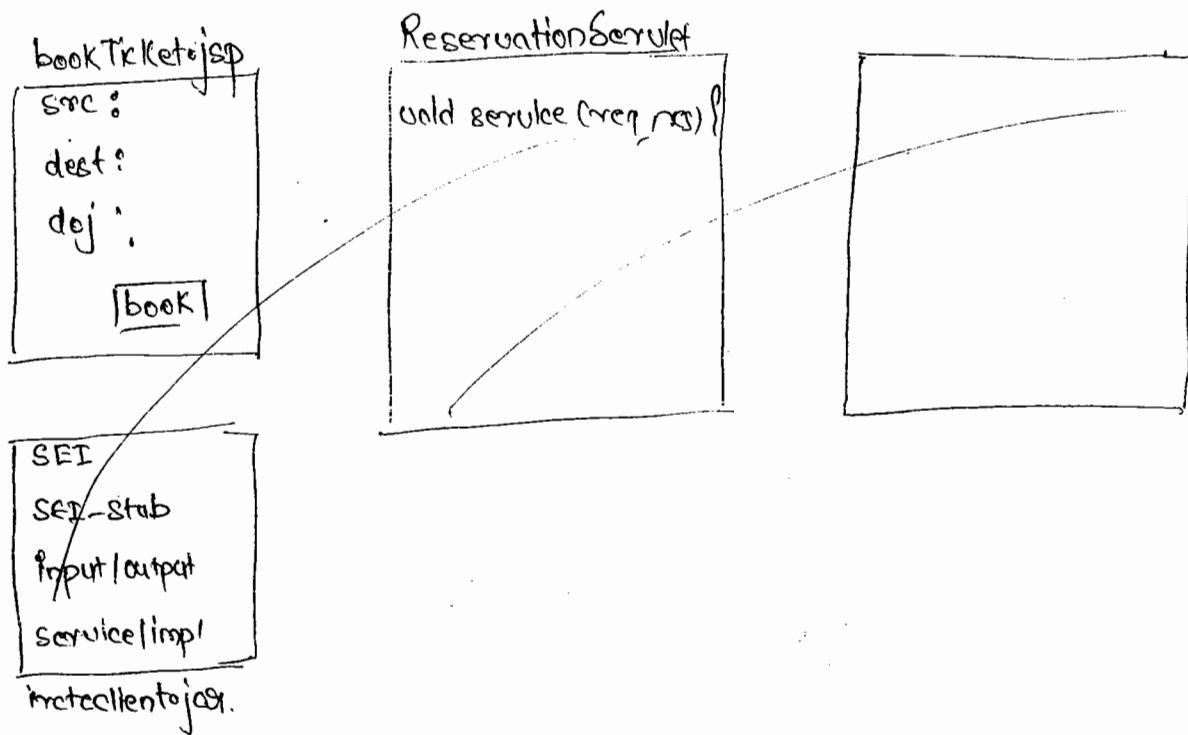
- └ relative workspace or metadata (default)
- └ server Location
- └ custom Location

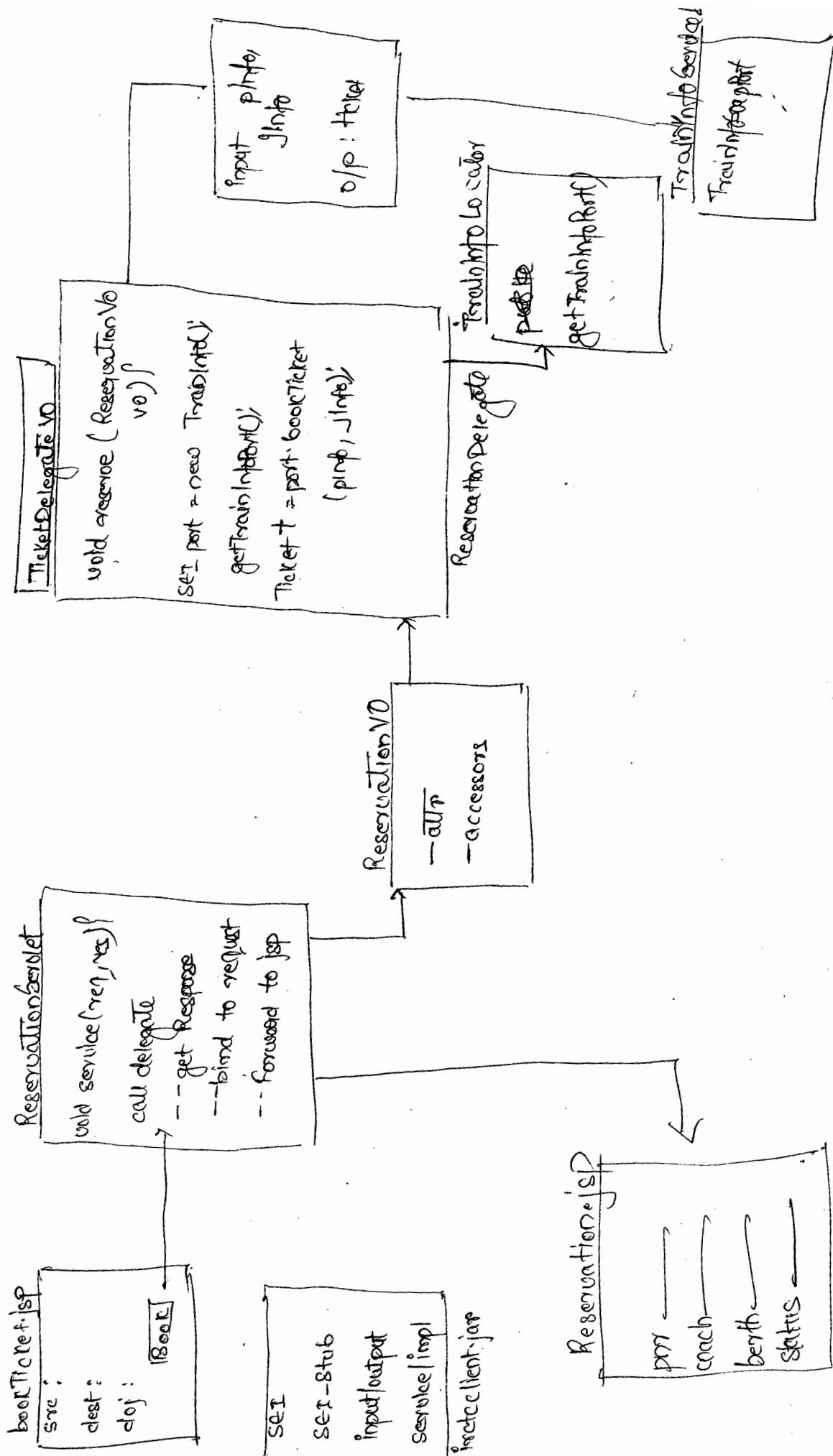
SE  
Sky  
g.  
II  
E  
prnt

08-05-19  
Thursday

## Consumer Web App

### Architecture from Consumer Perspective





- Start the Preulder services.
- Run again the tools as servers-config.wadl  $\oplus$   
generated at runtime.
- Generating binding classes.
  - Create TrainClient project.
  - new folder → paste jars.
    - Right click on proj → deploy  $\oplus$ .Assembly.
  - Run wsdl to java [no --servers-add]  $\oplus$  [with Skeleton Deploy tree]
  - Delete Deploy.wadl & Undeploy.wadl.
  - Delete Impl class.
  - Delete Skeleton class
  - Export Java → jar → next → only  src.
    - Export generated class files & resources.
- Create Dynamic web Project → TicketWeb
  - Copy all axis jars
  - Copy clientjars
  - Right click on webcontent → new jsp  $\Rightarrow$  BookTicket.jsp  
TicketDetails.jsp

```
<body>
<form> action = "url" method = "post" >
<table>
  <tr>
    <th> Journey Details </th>
    <th> Passenger Details </th>
  </tr>
  <tr> <!-- Journey table -->
    <td>
      <table> <tr> <td> src! </td>
        <td> <input type = "text" name = "src" /> -->
          dest
          doj
        <!-- Passenger table -->
        <td>
          ssp
          name gender
          dob
    </td>
  </tr>
```

```
<tr>
  <td colspan = "2" align = "center">
    → Write VO - Input/Op classes
```

com.yatra.web.vo  
com.yw.servlet  
com.yw.delegate  
com.yw.util (ServiceLocales)

```
class ReservationVO implements Serializable {
```

```
private static src;
  " " dest,
  " " date; doj;
  " " passenger;
  " " gender;
  " " dob;
```

// setters & getters.

}

class TicketDetail VO impl

String port,

beauty)

(coach;

Stages;

// setters & getters

}

gives the port of TrainInfo Service

class TrainInfoLocator

public TrainInfo getTrainInfoPort()

TrainInfoPort port = null;

TrainInfoService service = null;

service = new TrainInfoServiceLocator();

port = service.getTrainInfoSoapPort();

return port;

}

class ReservationDelegate

public TicketDetail VO receive(ReservationVO vo)

TrainInfo port = null;

TrainInfoLocator trainInfoLocator = null;

PassengerInfo pinfo = null;

JourneyInfo jinfo = null;

Ticket ticket = null;

TicketDetail VO ticketDetail VO = null;

SimpleDateFormat sdf = new

plInfo = new PassengerInfo();  
plInfo.set Ssn.  
" Gandy  
" Dob (sdf.parse (vo.getDob ()));

No. 1  
Final

pul

jInfo.setSrc (vo.getSrc ());  
" Dest (" " Dest ()"  
Dobj (" ~ Dobj()); sdf.parse (vo.getDobj ());

\*trainInfoLocator = new TrainInfoLocator();

port = trainInfoLocator.getTrainInfoPort();  
Ticket = port.bookTicket (plInfo, jInfo);

TicketDetailsVO = new TicketDetailVO();

TicketDetailsVO.setPnr (Ticket.getPNR ());  
getSeatNo()  
getCoach ()  
getStoles ()

return TicketDetailVO;

}

}

08-05-14

Friday

public class ReservationServlet extends HttpServlet {

service { - } }

TicketDetails VO ticket<sup>DetailVO</sup> = null;  
Reservation VO vo = null;

" Delegate & delegate = null;

vo = new ReservationVO();

vo.setSrc(req.getParam("src"));

vo.setFest(- - -)

Do |

SSN |

Gender |

Dob |

delegate = new ReservationDelegate;

TicketDetailVO = delegate.reserve(vo);

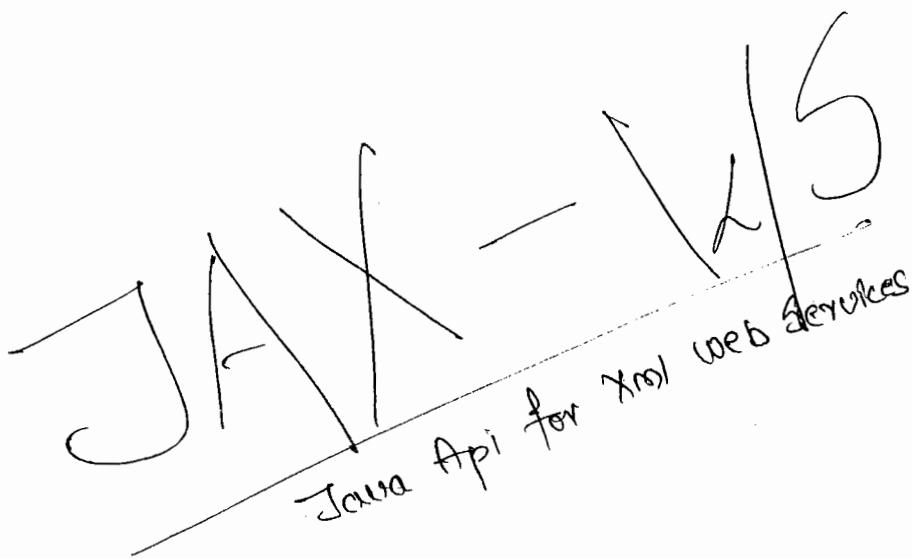
->req.setAttribute("ticket", ticketDetailVO);

ResDisp vd = req.getAttribute(- - -);

->vd.forward(req, res);

String src = req.getParameter("src");  
vo.setSrc(src);

# Diff. b/w JAX-RPC and JAX-WS

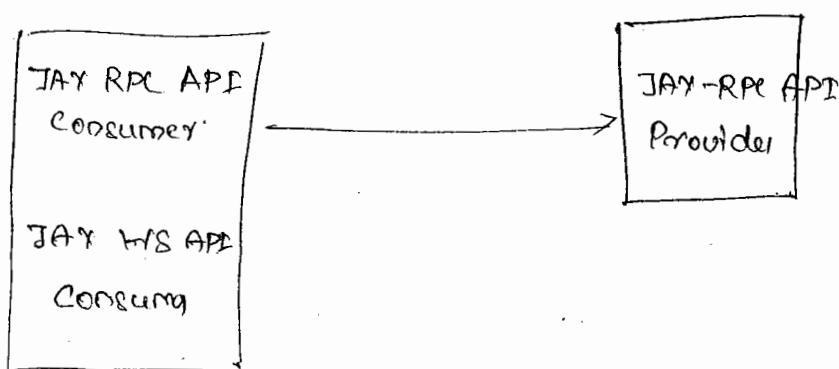


→ RPC means synchronous

Common b/w JAX-RPC & JAX-WS

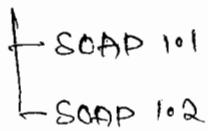
- Soap 1.1 over Http 1.1
- WSDL 1.1
- JAX-B
- SOAP 1.1 / 1.2 → JAX-WS

SOAP  
version



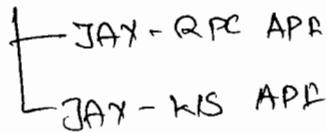
Both blocks can access.

→ Web Service is always developed for 2 bindings for



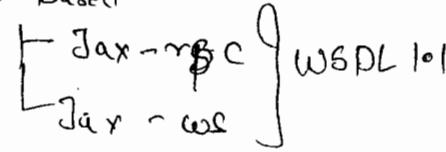
## Types of Web Services

Old Age



New Age

SOAP Based



WSDL 1.0

Non-SOAP Based

Jax-RS API (restful services)

Jersey (Impl. by Sun)

WSDL

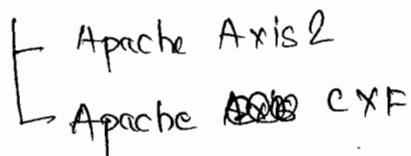
WADL → Web App Desc Language.

Proprietary by Sun

→ WSDL 2.0 is in the market but most of the vendors

haven't upgraded.

WSDL 2.0 Impl



Jax-RPC API — WSDL 1.0

Jax-WS API — WSDL 1.0 / WSDL 2.0

```

<binding name="TrainInfoSOAPBinding"
  type="tns:TrainInfo">
  <soap:binding transport="soap/http" style="rpc"/>
  <http:binding -->
</binding>

```

→ In JAX-RPC <http:binding> was avoided ↗

• In JAX-WS both are included (Soap:binding)

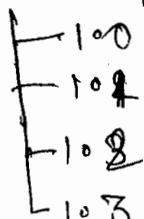
### soap based

jax-rpc  
jax-ws

### non-soap based

jax-rs  
jax-ws

### Basic Profiles



We're going to learn JAX-WS if it follows 1.0.3.

Jax-rpc - Java 1.4+

Jax-WS → JDK 1.5+

uses anno, autoboxing/unboxing, generics --

## Data Mapping Model — JAX-RPC

How to map Java data types to XML types

int -- xs:int

float -- xs:float

90% of java data types are mapped with schema types

→ WSE

— BP 1.0

JAX WS API → JAX-B ~~is mapping model~~

→ Only 90% are mapped, so in JAX-RPC if you take params as Serializable high chances are there that it will be mapped in the 90%.

11-05-14  
Sunday

→ JAX-RPC supports only sync req/rep.

→ JAX-RPC has its own data mapping model

## Interface Mapping Model

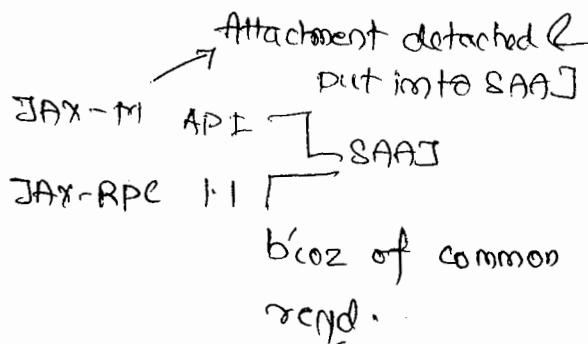
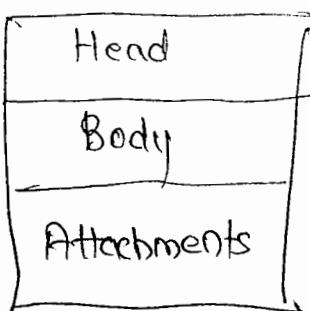
↳ Supports Java 5 features

↳ ~~async~~ func

# MTOM (Message Transmission Optimization Mechanism)

→ In

Soap Structure



M

→ Ver

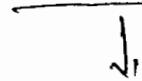
→ E

→ Initially (Java API for XML Messaging)

JAX-M API was there before web services over any protocol.

┌───┐  
 | XML msgs | can be sent.  
 └───┘  
 Attachments

JAX-RPC 1.0 API (attachments can't be sent)



JAX-RPC 1.1 API (SAAJ API) (Soap with Attachment API)  
 by Sun for Java

→ Before SAAJ by Sun, Microsoft released one specification MTOM for interoperability in attachments. MTOM became standard.

→ So, SAAJ wasn't accepted & hence not interoperable

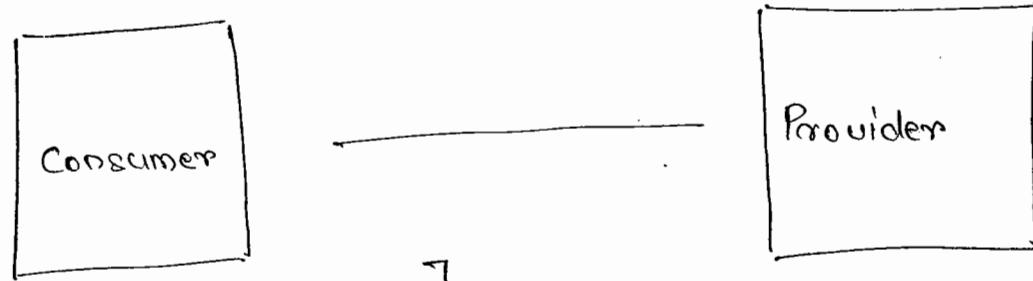
→ But, JAX-RPC used SAAJ API only.

→ Income Tax of India developed Web Services using SAAJ API.

- JAX-WS API supports MTOM, and SAAJ.
- In JAX-WS we can send attachments in two ways.  
MTOM and SAAJ.
- Very few web services will have attachments reqd.
- JAX-WS via JAXB adds support for MTOM.

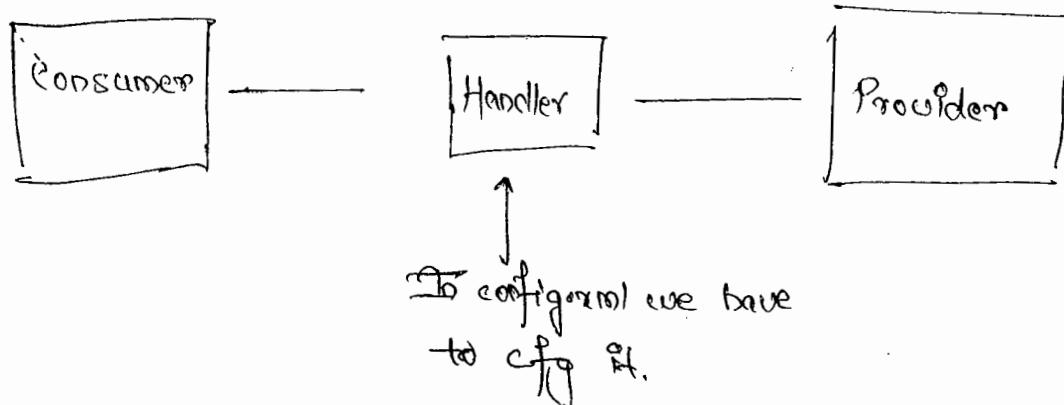
### The Handler Model

- Like filters of servlets & JSPs.
- Preprocessing & Postprocessing the request.



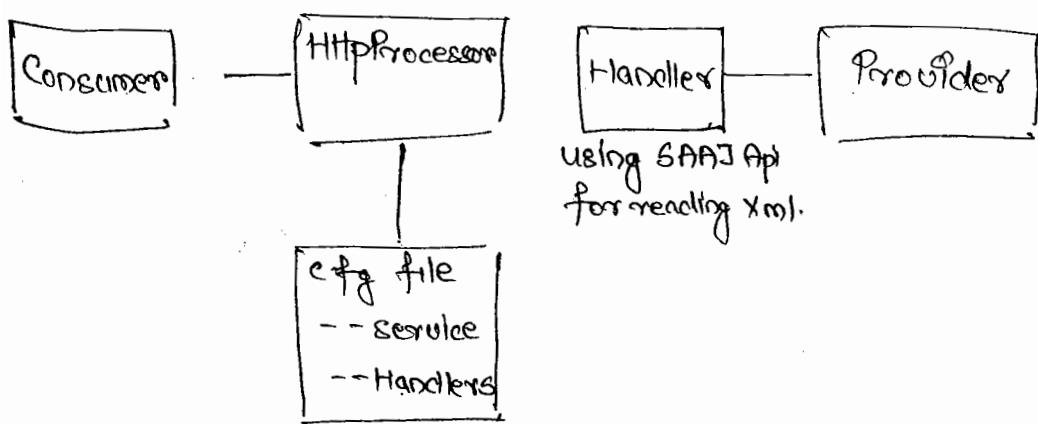
[Only B-partners should access provider.]

- For privileged access we need to authenticate. (Security)
- In SOAP header the username & pwd of consumer are present sent to provider. } (web service security)
- For in-house web services we don't need WS security as it's costly, we'll use custom security.



JAXRPC calls Tie, Tie finds Handler,

If Handler is present after processing it calls provider.



In Apache Axis we need to write one more cfg file to cfg the Handlers.

To Handler a SOAP XML will be passed as if username & password are invalid we'll waste time for conversion from XML to Object for calling Impl class.

-> Col

-> fr

-> fc

-> L

If

Con

-> D

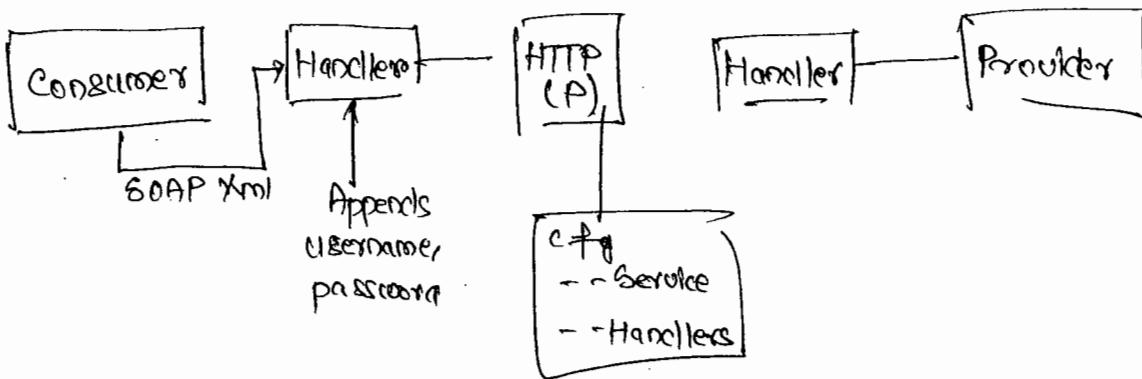
ui

-> J

RE

- > Compression & Decompression
  - > Encryption & Decryption
  - > Security
- } Use cases for  
Handlers.

- > Consumer side post calls stub obj . Stub checks if there're handlers & it will call provider.



- > Both Handlers are present in both consumer & provider unlike Filters.
- > In JAX-RPC you can't develop web service without SOAP.

## JAX-WS

- Handlers are of two types
  - > SOAP XML Handlers (Protocol Specific)
  - > Logical XML Handler (no soap xml)

→ JAX-RPC uses SAAJ 1.3 API

JAX-WS Handler uses SAAJ 1.3 API

## ⑩ When to use JAX-RPC / JAX-WS?

A →

### JAX-RPC

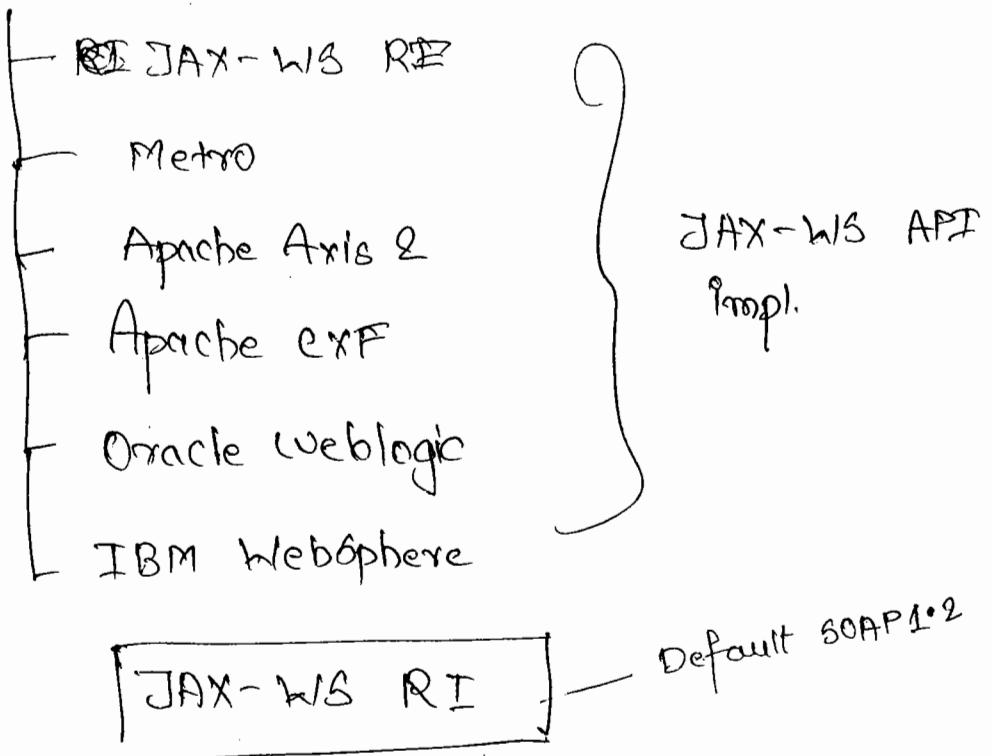
- Already running web services
- If you don't want to step up to Java5
- Creating RPC/encoded style WSDL & send SOAP encoded messages.

### JAX-WS

- New message-oriented API
- MTOM
- JAXB
- Sync prog. model.
- SOAP 1.2 messages can be handled by clients/services
- Eliminate SOAP & just use XML/HTTP binding

JAX-RPC is deprecated API.

# JAX-WS API Based Web Service



→ WS-BP 1:1 JAX-WS API, JAX-WS RI impl

using servlet endpoint using contract first using  
based.  
asynchronous req/rep using document literal web service.

not required

```
interface Bank {  
    extends Remote {  
        . . .  
    }  
}
```

@WebService

```
interface Bank {
```

// declare any methods By default all methods  
will become web service methods.

④ @WebMethod

m1()

m2()

→ If you annotate atleast one method with

④ @WebMethod then the annotated method(s) will be exposed as web service methods.

→ Parameters & Return types are JAX-WS data mapped to mapping so it's not recommended to be Serializable.

→ We needn't throw RemoteException. JAX-WS is independent of RMI API. JAX-WS has its own dedicated API.

### Jax-ws api

— WebServiceException (unchecked exception)

→ If catching Exception also o/p is end of prg  
then we use un-checked exception

→ If provider isn't able to process req then  
what why to use checked exception. You don't  
need to throw it & catch it.

④ WebService  
interface Bank {

    m1()

    m2()

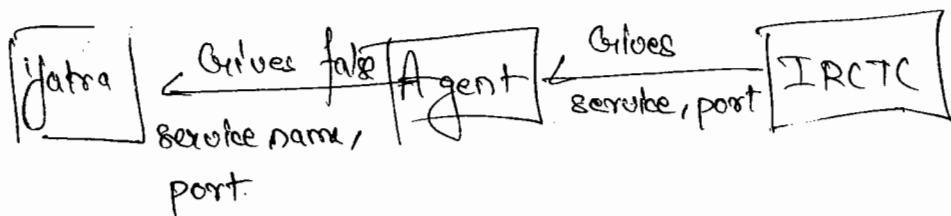
    @webMethod(exclude=true)  
    m3()

} only this m100() will not  
be exposed as web service.

⑤ WebService(targetNamespace = " " . . . )  
interface Bank {

    Receipt withdraw(AccountInfo aInfo, TransactionInfo tInfo)

}



→ If namespace is there Yatra can authenticate  
if it's correct port & service.

⑥ WebService(targetNamespace = "http://cici.com/transaction/card",

    name = "BOOK")

    Port name

→ In JAX-RPC a SF impl contract last approach

we can't use SOAP action.

⑥ `@WebMethod(action = "http://www.example.com/transaction/web#withdrawl")`

operationName = "fundDraw")

appears in UML for usability  
(by default name of operation)

withdrawl (-, -)

appears in Java code

Momo

→ [ ]

]

①

Receipt withdrawal (@WebParam(targetNamespace = " -- /types",

~~name = "aInfo") AccountInfo,~~  
else string-1 like bad naming

@WebParam(targetNamespace = " -- /types", -- )

→ @WebResult(targetNamespace = " -- /types",

name = "result")

}

→ Service name & Port name will be in

concrete i.e. Impl class.

→ Apro avoids vendor specific cfg file.

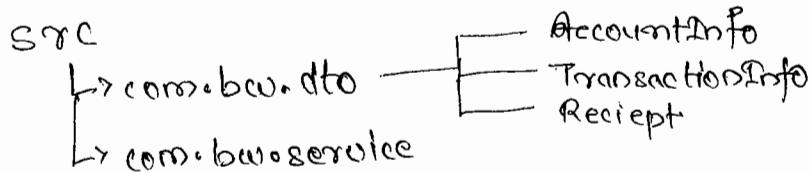
→ Annotations are also called as source code metadata.

12-08-14  
Monday

→ BankWeb → JAX-WS First project

JAR bundles → Jax-WS → copy & paste jars.

① Write SEI Interface Create



class AccountInfo {  
 string accNo;  
 accountHolderName;  
 branch;  
 // setters & getters  
}  
  
public class TransactionInfo {  
 int atmId;  
 float amount;  
 // setters & getters  
}  
  
class Receipt {  
 long transactionId;  
 float balance;  
 string statusCode;  
}

Input class                          Output class

Interface

pub

press

esc → ctrl+space [IDE bug] for @WebService

@WebService(targetNamespace="http://icibank.com/trans/websvc",  
name="bank")

public interface Bank {

@WebMethod(action="http://icibank.com/trans/websvc#withdraw")  
Receipt withdraw(@AccountInfo accountInfo,  
TransactionInfo transactionInfo)

@WebParam(targetNamespace="--/types" name="accountInfo") } write at  
method level

@WebParam(targetNamespace="--/types" name="transactionInfo") } write at  
method level

@WebParam(name="receipt") } cut paste  
to extra parameters

not required  
class BankImpl implements SEF

@WebService(endpointInterface="com.bco.service.Bank",  
serviceName="BankService",  
portName="BankSOAPPort") } for JAX-RPC we  
have to use  
SOAP 1:1 port &  
for WS SOAP 1:2  
ports. 2 ports  
are exposed.

Implementation

@SOAPBinding(bindingType=SOAP11Binding)

class BankImpl {

```
public Receipt withdrawl { AccountInfo, transactionInfo } }
```

}

//For 2nd binding for SOAP 1.2.

```
@WebService(----- portName = "BankSOAP12Port" )  
@SOAPBinding(bindingType = SOAP12Binding)  
class BankImpl12 {
```

    }

③ How to manage versioning ( old → new )

A. → namespace → ~~types1.0~~ types1.0  
→ wsdl1.0

④ How can you maintain version in xsd / ?wsdl ?

A) Based on namespace & binding.

⑩ How to maintain version if already running web service.

⑪ Web

⑪ URL → append "new" to URL.

⑫ Bio

SLA → Service Level Agreement period.

public

↳ In how many days to move from old → new service

(Rolling out versions)

→ In case of JAX-WS web service,  
SEI interface is optional.

→ If there's no SEI then anno written in SEI have to be written to SEIImpl.

→ But, if multiple Impl classes are there then better to write SEI as the abstract part will be common b/w all Impl classes.

→ Methods shouldn't be private, static & final so that they can be exposed as web service method.

→ RI ~~for~~ only supports SOAP 1.1

13-

⑬ (:

-71

to

err!

→ Tc

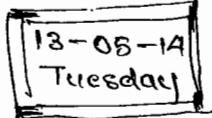
ws

ws

⑥ WebService Endpoint Interface = "com.bcm.service.Bank"  
serviceName = "BankService"  
portName = "BankPort")

⑦ Binding Type (SOAPBinding, SOAP11HTTP-BINDING)

```
public class BankImpl {  
    // override  
    public Receipt withdraw ( AccountInfo, TransactionInfo ) {  
        Receipt receipt = null;  
        if ( receipt == null )  
            receipt = new Receipt();  
        receipt.setTransactionId(121);  
        receipt.setBalance( );  
        receipt.setStatusCode(1);  
    }  
}
```



### ⑧ Generating Binding classes

→ If BEI is mandatory, then for any changes we have to modify BEI, directly or in JAX-WB. We can expose impl classes as web services

→ Tools

wsgen (c:\sun\jewdp-2.0\jaxws\bin)  
wsimport (like WSDL2Java) (consumer side)

wsgen (web service generator) like Java2WSDL (provider side)

wsgen

- d src
- keep
- + -cp build/classes
- + -verbose

How to run

What will be generated while running wsgen tool?

→ I

impl

wsgen

B'coz of JAX-B

- + wsdl (generated at runtime only) ↓
- + RequestStruct / ResponseStruct X → withdraw /
- + SOAPserializers / Builders X      withdrawResponse
- + Tie X
- + model X

(A) He

with.

→ RequestStruct / ResponseStruct are generated

as withdraw / withdrawResponse b'coz of

JAX-B binding.

→ I

→ Latest API generates runtime wsdl. Unless you request for it there'll be no physical wsdl.

on

8

C:\> set path=%path%; C:\Sun\jwsdp-2.0\jaxws\bin

L01

C:\> wsgen -help

C:\> set JAVA\_HOME=C:\Program Files\Java\JDK- -

re

C:\> wsgen -help

~~skip~~  
C:\Proj-dir> wsigen -d src -keep -cp build\classes  
-verbose com.bws.service.BankImpl

→ In the pkg which contains Impl class one subpkg will be generated.

com.bws.service.jaxws

④ Here, we need to ~~cfg~~ write cfg file to map url with service. (B'coz no internal classes are being generated)

sun-jaxws.xml → complete web service deployment descriptor

{  
  | Name of endpoint     | As, we know everything that we're  
  | url pattern.         | writing in this xml file so we  
  | Name of Impl         | don't need tool.

→ JAX-WS - RI provides WSServlet. We need to config it in web.xml.

sun-jaxws.xml (Vendor specific)

<endpoint name="Bank"

implementation="com.bws.service.BankImpl"

url-pattern="/fidi/bank" />

web.xml

--servlet

--name Bank

--class WSServlet

--servlet-mapping

--name Bank

--url-pattern /fidi/bank

Should match.

new app

{servlet}

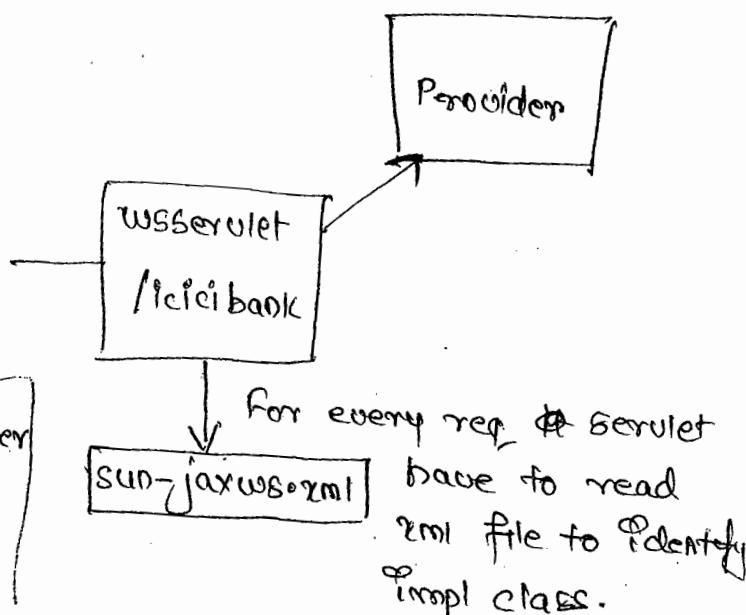
{servlet-name} Bank < / -->  
< / - class > Ctrl+Shift+F < / -->  
or  
write WSServlet in impl import it  
copy file name & paste  
Enter copy name  
& paste.

{/web-app}

→ We have to cfg one Listener also given by ~~jaxws~~ <sup>WSServletContextListener</sup> Prop.  
↓  
JAXWS - ~~R~~ <sup>WSServletContextListener</sup> Prop.

~~Ctrl+Shift+E~~ (Ctrl+Shift+E) on Listener class.

Request Processing Flow



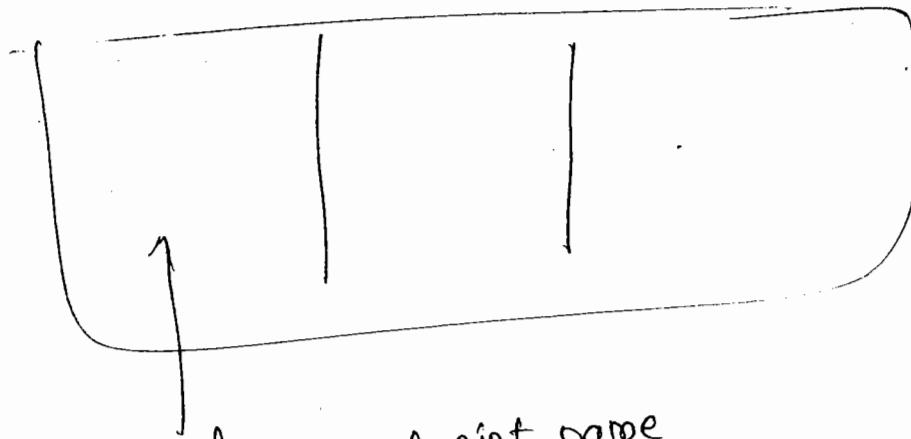
Listener working !!

→ ~~jaxws~~  
→ we  
bank  
So,  
Serv..  
→ co  
&  
→ F..  
→ In  
spec.  
→ M..

→ ~~the~~ web service

→ WebServletContext listener creates WebServiceContent obj  
having all the info. for the web service in the app  
So, that for each incoming req. we needn't to the  
Servlet has to read the xml file.

## Web Services



comes from endpoint name

- copy wsdl url from dynamically generated wsdl & paste it in address bar, it'll give abstract wsdl.
- From SOAP ui send request.
- In contract last the data is optional b'coz you can't specify minOccurs & maxOccurs
- Members variables should be private.

SOAP

- Simple Object Access Protocol

Working

→ Body

→ SOAP

→ Content

L

+

→ Context

/

→ BFF

→ Service

IS.

→ S

NE

→ C

Pmt

→ D

→ SOAP 1.2 is the standard messaging protocol used by

J2EE Web Services. De Facto standard.

→ SOAP is another XML Markup language. Used for exchanging data over m/w. It's m/w app protocol.

→ A SOAP XML document instance, which is called SOAP message, is usually carried as payload of some other m/w protocol.

→ SOAP XML document is also called SOAP envelop.

→ A lightweight protocol for exchange of info in a decentralised, distributed env.

→ Web Services can use One-way messaging or Req/Res messaging.

→ SOAP has its own schema, namespaces, processing rules

&lt;envelope&gt;

&lt;header/&gt; → optional

&lt;body&gt;

&lt;fault/&gt;

&lt;/body&gt;

&lt;/envelope&gt;

→ Header contains XML elements describing Info present in Body message.

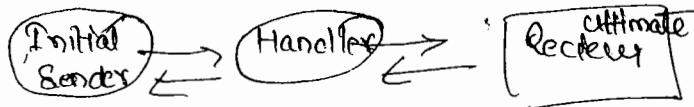
→ Body contains any arbitrary XML.

### → SOAP Namespaces

→ Code Modules with SOAP Namespaces.

↳ Modularity enables you to use diff code libraries to process diff parts of SOAP message.

### → SOAP Message Path



→ BP designed SOAP XML & there can be only one body.

### → SOAP Messaging Modes

- └ RPC Encoded
- └ RPC Literal
- └ Document Literal

### SOAP Faults

→ ⚡ Mechanism by which SOAP app report errors "upstream" to nodes earlier in the message path

→ SOAP faults are generated by receivers, either at intermediary or ultimate receiver.

→ In fire & forget no faults are received.

message contains a Fault msg in body  
is called Fault message.

<envelope>

<header/>

<body>

any XML / <fault> → In body either only

</body>

<envelope>

<fault> is there or XML is present.

→ Det.

→ <fault>

process

<faultCode>

① pointer

<faultString>

<faultactor> ] optional.

<detail/>

↓

</fault>

class

shape

class AppExpData {

String modelName; } Detail section  
" exp message;" contain this data  
" friendly error;"

}

class myException {

private AppExpData expData;

~~fault codes to identify error~~

SOAP standard (consumer & provider use diff SOAP)

Fault codes

Client Server Version Mismatch.

Must understand (sent by consumer in header). The first one to receive soap has to process it b/c security lie there

→ Detail element should be included in exceptn while processing body not header.

① Interface SEF {

public float getBookPrice (String isbn) {

}

class SEFStubImpl SEF {

public float getBookPrice (String isbn) {

SOAP

// converts to xml

// Soap xml

// open connection (HTTP)

// sends provider if receives SOAP fault & throws exception

+ // response → if stub receives success resp  
builds obj & returns

class MyClient {  
    PSOM {

        service SEF-port = service.getPort();

        SEF-port.getBookPrice (J);

→ In webd we'll have Input, output & fault message also.

→ Provider will send normal resp / ~~fault~~ or normal response only. Internally state has to classify based on webd.

15-08-14  
Thursday

to  
→ 80.

④ Identity

Q18

① <

## Message Exchanging Format

→ 6c

① RPC/Encoded    ③ Document/encoded

80.

② RPC/Literal    ④ Document/literal

are

⑤ document/wrapped

interface MemberRegistration {

MemberCap enroll (MemberInfo mInfo, PolicyInfo pInfo);

}

→ (part) differs for MEF.

RPC/Encoded

Adv!

② 1

→ Soap Action differentiates method call

In service Ids sent as part of ~~http~~ SOAP

header which is plaintext. Cipher can be

used to back it & compare to webd if he has it.

case

③ 0

→ Portaudio can manipulate SOAP action / operation & a user  
to provider.

- ① WSDL is self explanatory.
- ② Operation name appears as root of msg

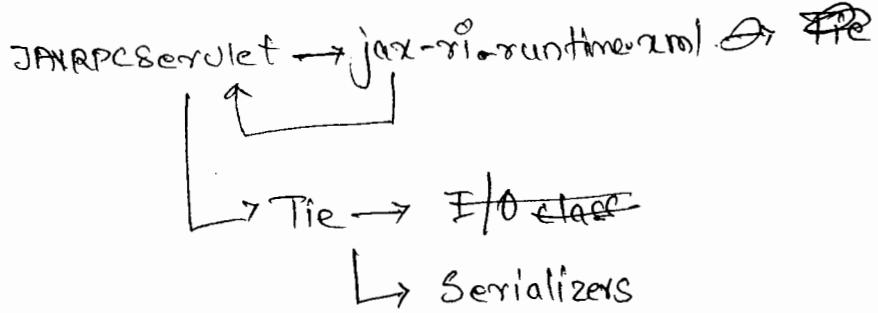
→ So, ~~rest~~ body starts with method name to easily  
identify rest operation & hence, encrypted.

Q18 - adv:

① <ssd> 28i:type = "xsd:string" </ssd> & N | </ssd>  
not reqd. overhead in processing

→ Serializers are built gen. based on complex type.

So, SOAP and sent by consumer having "type encoding"  
are overhead.



- ② The RPC / encoded SOAP message some parts  
are of ~~xsd~~ & some of ~~@wsdl~~.
- ③ It's legal WSDL, but not ws-I compliant.

ML - LiteralAdv

- ① WSDL is self explanatory.
- ② OP name in msg.
- ③ No type encoding
- ④ WS-I compliant

Disadv.

- ① Still SOAP msg contains xsd & wsdl.

Document-literal (<sup>contains data</sup> Data will be sent literally)

Adv

- ① In wsdl <types> has element another than complex type
- Expected by  
probably &  
xsd: element name = "MemberInfo" hence the  
<xsd:complexType> <xsd:complexType>
- </xsd:element>
- </types>

- ② Directly elements will be passed.

→ message part section will contain name & element not types.

@WebService  
Interface Bank

Receipt withdraw (@WebParam (targetNamespace = "",  
(RPC-Enc | RPC-lit) (goes to msg part)  $\rightarrow$  partName = "",  
(Document literal will have name of element)  $\rightarrow$  name = "")

$\rightarrow$  If partName isn't specified then.

<message>

<part name = "arg0">

</message>

~~Adv~~ Adv

① No type enc.

② Validate XML over {schemas} (All are from schema.)

③ WSDL-compliant but with restrictions.

D's Adv

① WSDL is bit complicated.

② Operation name in SOAP msg is lost. Without the name-dispatching can be difficult.

③ WSDL allows only one child of <Soap:body> in the SOAP message.

④ Resolving method call is slow.

Reason: ... in only 1 name

Friday

→ style of wsdl → proposed by Microsoft

## Document wrapped

→ It's not a Message Exchange Format. Only 4 formats are there.

→ It's Document Literal but the way we're writing is document wrapped. ↗

→ In binding section it's document literal.

→ WSDL non-compliant but legal.

•

→ JAX-WS RI uses document unwrapped to generate wsdl.

→ Some, Iimpl may support document wrapped some mayn't

@WebService

document literal  
↓                  ↙ document wrapped

@SoapBinding(parameterStyle = BARE / WRAPPED)

interface Bank {

    Receipt withdraw ( - , - );

}

→ In JAX-WS-RI you can't use "BARE" with more than 1 param.

→ It works in Apache Axis 2

17-n  
Sat

fr

→ ↑

→ ↓

→ ↘

↑

soft.

→ In literal <part name=" " - . >  
plays no role.

17-05-14  
Saturday

Contract First

— RI

Interface Hospital

BillSummary generateBill (AdmissionInfo admissionInfo,

TreatmentSummary treatmentSummary)

}

→ Hospital Web project

→ Paste JAXWS - RI jars

② targetNamespace: http://apollo.com/health/websvc

<types>

<element name="admissionInfo">

<complexType> → Anonymous.

<sequence>

<element name="admissionId" type=">">

= "roomNo"

= "patientName"

< />  
< />  
< />  
< /> - "treatmentSummary"  
"treatmentSummary"

<element name="diseases">  
"cost"

④  $\vdash "bill"$

$\vdash "admissionId"$

$\vdash "amount"$

$\vdash "status"$

{ message name = "generateBill" }

<part name = "in-1" element = "admissionInfo"/>

class HospitalFault

string

hospitalFaultCode

message

}

class BillGenerationException ~~extends Exception~~

HospitalFault hospitalFault;

<element name = "hospitalFault">

<complexType>

<sequence>

<element name = "hospitalFaultCode" type = " " />

<

= "message"

</>

</>

</>

<message name = "BillGenerationFault">

<part name = "fault-1" element = "hospitalFault"/>

</>

→ Every Exception has to be cfg as one fault.

< portType>

{ name = "billGenerationFault" message = "Bill Generation Fault" }

<!-->

< binding -->

< fault name = "billGenerationFault" >

< soap:fault name = "Bill Generation Fault" />

< fault>

→ Gen binding class.

wsdl

|  
wsimport → I/O

→ BEI

→ Impl class

→ msg (mess struct)

It  
not changes so not  
generated.

→ (jaxb binding classes)

→ soap serializers / builders

→ tie

→ model

not generated.

→ service

→ stub

wsimport = edmx

- d src

- keep

- verbose

wsdl

Hospital.jaxws

W81m

WS-INF\Location> \WEB-INF\Hospital.wsdll

18-0  
Ex.

→ Write Impl class  
↳ reqd. of serviceName in Impl class

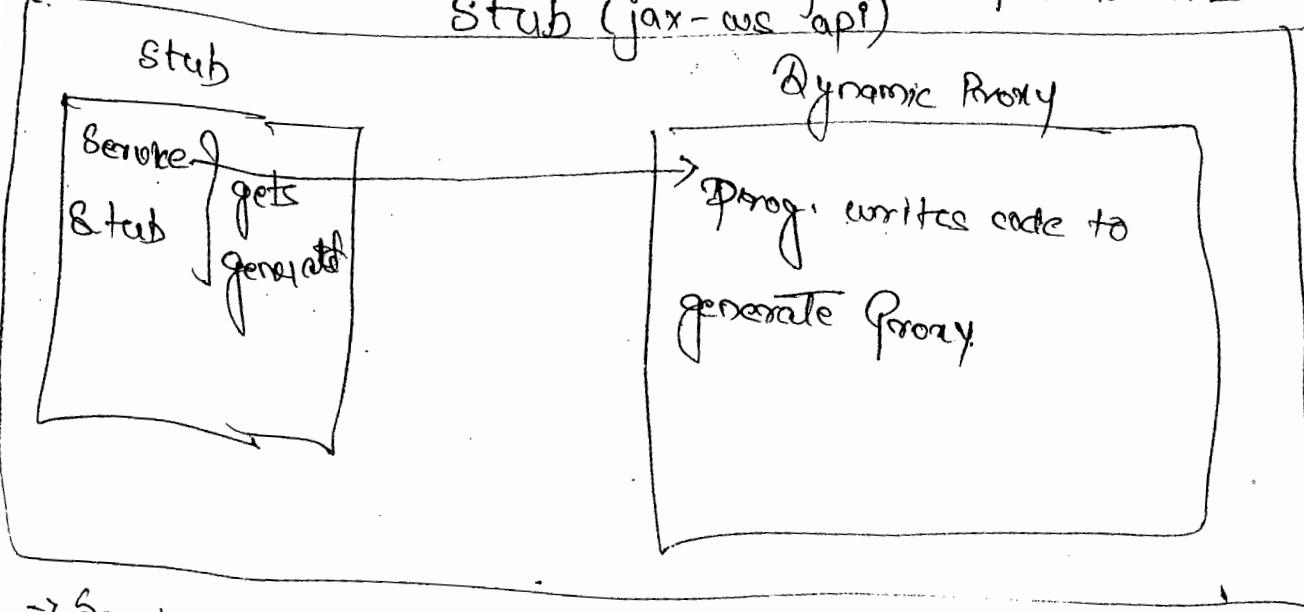
→ In this development we only create ~~an~~ abstract wsdll.

→ 11

### JAX WS - API Consumer

→ Stub based consumer (dynamic proxy)  
(Dynamic Invocation Interface) Dispatch API.  
Stub (jax-ws api)

④



→ Service contains the code to generate Proxy at runtime.

### Dispatch API

→ Sync & Async

wsimport - - - url-of-running-soapdl.

18-05-19  
Sunday

## Dispatch API Based Consumer

→ HospitalImpl.java

@WebService (endpointInterface = " ", serviceName = " ",  
portName = " ")

```
class HospitalImpl {  
}  
}
```

wl won't be taken from  
wsdl. B'coz, if you  
want to use two diff  
soap port bindings (1.1/1.2)  
it can't get from  
abstract wsdl.

## Dispatch Api

- No vendor specific tools reqd.
- we can call any ~~method~~ service method on any ~~port~~ of on  
any service. port
- Consumers will be portable across various implementations.

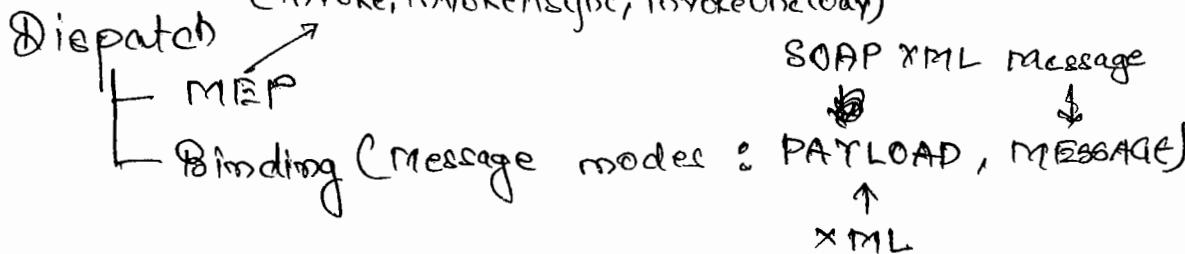
to Dispatch API.

→ Since call was called on method which is ~~not~~ sync.

SOAP Bind.

Dispatch obj uses message

(`invoke, invokeAsync, invokeOneWay`)



hospi

cla

Create Service

-- add port

Create Dispatch -- service knows how to create

`javax.xml.webservices.Service (AS)`

↳ contains static factory

Service hospitalService = service.createService()

`new QName("tns", "hospitalImplService");`

hospitalService.addPort (~~new~~ new `QName ("tns", "HospitalImplPort")`,

Dispatch hospitalDispatch  
SOAPBinding --> );  
SAAJ API

Dispatch<SOAPMessage> hospitalDispatch = hospitalService.createDispatch (

`"PortName", MessageMode.MESSAGE,`  
`SoapMessage.class);`

MessageFactory factory = MessageFactory.newInstance();

SOAPMessage soapMessage = factory.createSoapMessage();

SOAPBody body = soapMessage.getBody();  
SOAPElement ~~rootElement~~<sup>opNmlement</sup> = body.<sup>child</sup>addElement();  
SOAPElement

hospitalDispatch.invoke(soapMessage);

~~main()~~  
class HospitalDispatchTest{  
SERVICE-NM = ""  
PORT-NM = ""  
TARGET-NM = ""  
TYPE-NM = ""  
TARGET-ADDRESS = "http://127.0.0.1:8080/apollo";  
p8vm(-)}

Thursday

# Apache Axis 2

Inte...  
systems

→ ~~Web~~ 2nd gen web service engine by Apache.

→ ~~Apache~~ New impl for JAX-WS.

→ No interface

## Features

→ Completely coded in Java.

→ Java & C based web services can be developed.

→ Supports SAAJ API. (Has its own impl for API)

→ Can generate WSDL at runtime.

→ Can build provider & consumer.

→ Supports multiple transport protocols.

→ JAX-B and third party API (XmlBeans, GAXB...)

~~also~~ data binding protocols supported.

→ Security of web services are there (Advanced features)

→ Dynamic Deployment Model is impl in diff. way.

## How to work?

→ Comes in 2 distributions

→ Binary distribution 1.6.1 } Both should have  
→ war distribution 1.6.1 } same version

→ Current ver - (1.6.2)

→ We're studying (1.6.1)

→ 2

→ waso

→ jdk

6

Set..

box

→ Col

→ E

⑥ 101

{ x

(A)

i..1

→ To support new deployment technology war distribution is given.

~~java~~

→ 2 flavours of java (distribution)  
└ jdk (tools)  
└ jre (interpreter)

- war distribution is runtime engine of Apache Axis 2.  
→ jdk 1.6 is only compatible with Apache Axis 2.

~~axis2~~

Setting Env

axis2-1.6.1-bin.zip (Binary dist)

→ Copy to C:\

→ Extract to here

C:\axis2-1.6.1

└ webapp  
  └ etc  
  └ samples  
  └ lib (all need & optional jars)  
  └ bin (tools - wsdl2java, java2wsdl, ~~axis2anttask~~ standalone server)  
  └ conf (axis2.xml)  
└ repository  
  └ services (~~version.xml~~ → apache archieve)  
  └ modules (additional features • marn (module archive))

⑥ Why JDBC API ~~isn't~~ isn't shipped as part of ~~JDK~~ given as additional API?

⑦ Every core Java program may need to talk to DB. If included bootstrap loader will take more time.

→ Runtime env. doesn't have additional features. So, copy over to runtime.

Friday

→ W:

as

→ Res

→ T,

→ I..

① C

Ax

②

[f]

→ Dy

Ej

→ Cup

devel

ui

→ A

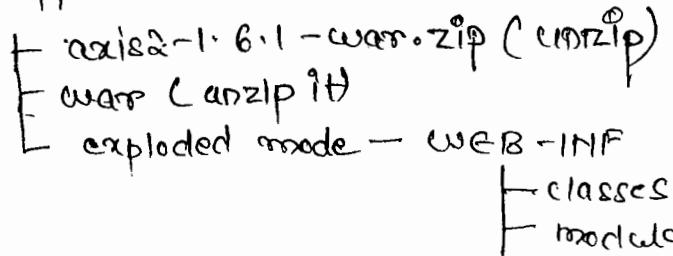
I

## Runtime Env Setup (war dist)

axis2-1.6.1-war.zip

d:\sandbox\apache-tomcat-6.0.20

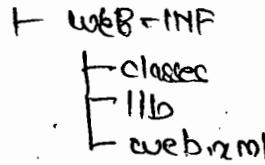
+ webapps



→ Any project can be deployed in server in two modes

+ Packaged mode (war)

- Exploded mode (extra folders)



→ we need to start server from console b'coz if you

do from IDE it will look into META-INF not

webapps

http://localhost:8080/axis2

username:	admin
password:	axis2

any  
23-08-14

Friday

→ Without any annotation you can expose a method as Web Service.

→ Restful Service is provided by Apache Axis 2.

→ Transaction isn't supported.

→ 1.6.2 has bug it doesn't start server.

⑥ Where can I see the services running in Apache Axis 2?



### How to deploy Web Service in Apache Axis 2

→ Dynamic Deployment Model of Apache Axis is one of biggest draw back due to complexity.

→ Copying the physical bits of app. to server runtime file deployment for traditional java developer. But we need to use AdminClient.

→ Apache Axis 2 supports dynamic deployment tech.  
It has diff. deployment model.

- webapps

+ axis 2

+ WEB-INF

+ lib

+ classes

+ web.xml

+ modules

+ services

+ conf

+ axis2.xml

runtime env specif  
cfg

→ Packaging model we need to package our web app

i.e. (.aar) - Apache application archive.

+ aar

+ META-INF → manifest.xml, wsdl, wsdd

+ classes (or) → package (folder)

+ SEF

+ SEI-Impl

+ I/O classes

services.xml

+ .class (Java class)

+ lib

In case if you're  
using additional jars

else optional

+ META-INF b'coz lib

+ manifest.xml is to

+ package axis2

+ .class

WEB-INF

copy & paste to  
services folder to  
deploy & remove it to  
undeploy.

→ Auto deployment feature

→ Top

to

→ Web

be

car

→ Ap-

dev

IP

→

-

→ C

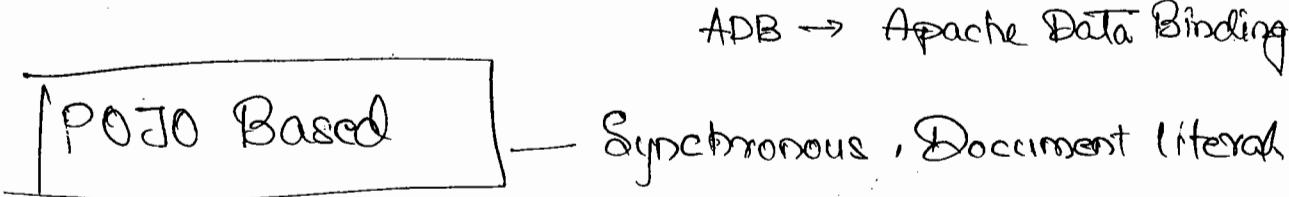
→ Tomcat server deploys our web app but

to deploy war we need another deployer.

→ We can now create core java project. Since, it can be exported as jar. Then add things to make it war.

→ Apache Axis2 supports 2 ways of web service development:

- POJO Based (Real time) (Used 60% 99%)
- Jax-ws api compliant ws development



→ Java Project → Passport

interface Applying PassportManagement {

    Receipt apply(PersonInfo personInfo);

↑  
Single param (doc literal)

→ Build path → Axis2 → Lib → add. (only jars (~~axis~~))

→ Create folder wsdl.

→ Create wsdl in wsdl folder

### PassportManagement.wsdl

in localhost:9090/wsdl

11 " /types

Document Literal.

Profile

→ C

SSDN  
name

→ C

Applet  
status

→ C

→ Try removing service & binding.

→ C

→ Generate binding checks.

Kic

Set JAVA\_HOME.

IP

Set path to bin → "%path%; %axis2%\bin"

C

wsdl2java -help

Set AXIS2\_HOME = C:\Axis2\axis2-1.8.1

F

Break

F

24-05-14  
Saturday

→ Apache Axis doesn't support document wrapped.

Default data binding : adb

Project Dir > wsdl&java - wsdl2

→ Write logic to generate implementation class

→ Every .wsdl has to be loaded by different class loader.

→ Start Server

→ apache-tomcat > startup

→ Deploy service.

Right click on proj → Export It as jar → next → expand.

only select ser. Browse.. (for location) passport.jar

passport.jar → next → finish.

→ Copy the file paste to webapp → Axis2 → ... → services.

→ Open SOAP UI & test.

~~SOAP UI~~

→ If you remove the .wsdl an error page will be displayed,

# JAX-WS style of web service development

Sund

```
@WebService(serviceName="EchoService", portName="theSOAPport", targetNamespace="")
public class Echo {
    @WebMethod(operationName="echoMessage")
    @RequestWrapper(localName="echoMessage")
    @ResponseWrapper(localName="result")
    @RequestWrapper(localName="echoMessage")
    @ResponseWrapper(localName="result")
    public String echoMessage(String message) {
        return message;
    }
}
```

→ Export as jar (.jar)

→ Create new folder in axis2 -- / servicejars

→ Paste jar file.

jax-rpc api  
  |  
  +-- jax-rpc si  
  
jwsdp 1.0.x {jdk1.4}  
jwsdp 2.0.x {jdk1.5}

+ apache-axis  
  +-- dynamic deployment model.  
      +-- gui development.

→ SUN GUI tools  
  +-- Netbeans  
  +-- Sun glass-fish server

Oracle

  +-- RAID  
  +-- Grid computing. (for huge table problem)

25-08-14  
Sunday

→ Oracle vs IBM vs SAP

→ Oracle Fusion products

↳ Oracle self funding

Larry Ellison

Group (R&D)

↳ Oracle Fusion tech (Server technologies)

↳ Oracle Fusion Apps (developed using fusion tech)  
(application engineers)

↳ oracle apps engineers

— oracle customer oracle (ocs)  
work at client loc.

## Fusion tech

↳ Java

— ADF (Application Development Framework)

↳ Not flexible. You have to tune your need to it

JET is substitute

— OSB (Oracle Service Bus)

— SOA (SCA specification) — acquisition

— Weblogic Server

Mediator

— Oracle Webcenters (BEA Portlets)

## Fusion Apps

↳ Oracle Apps

↳ Oracle E-Business suite

↳ Oracle PeopleSoft

↳ Oracle JD Edwards

Aquisitions

## Initial Release

— SOA 10g — At that time oracle has (OC4J Server)

↓  
not Weblogic

— SOA 11g — @ Weblogic Server

## Oracle Container & JBoss

Integration Tier  
Technology

-> In

J2EE servers (not full fledged)



OC4J (acquired)



-> Ac

Weblogic (Acquired from BEA Weblogic)

-> Ecl

BEA Aqualogic Weblogic Server

plm



→ IBM bad tech. & h/w business. But, Oracle was in tech

So, Oracle bought Sun. (Sun Solaris machines.)

TK - ~~Thomas~~ Thomas Kurien - Product Development Director  
(Indian)

→ Sun Solaris → Oracle Hexalogic Box. (Facebook, YesBank,  
Vodafone, Airlines)

-> C

IRETC → Weblogic Servers.

Oracle

| IDE - JDeveloper  
| Platform - WebLogic

Sun

| IDE - NetBeans  
| Platform - GlassFish

Sun One App Server

↳  
GlassFish

→ Sun Tech. Seminars

[www.javaonet.org](http://www.javaonet.org)

→ Glassfish (fastest startup & shutdown)

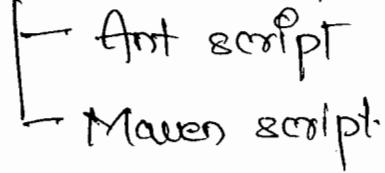
→ Glassfish has web services jar (Metro Engine)

-> In real time IDE aren't used for Web Services.

-> Automated build scripts are used.

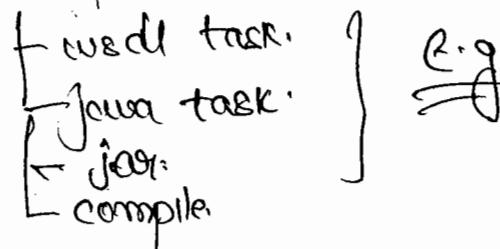
-> Eclipse supports for ws development we have to add plugin (Apache Axis 2 only)

### Automated Build Script



-> Tools will be automated in build scripts

-> Build script has tasks like







## Test class

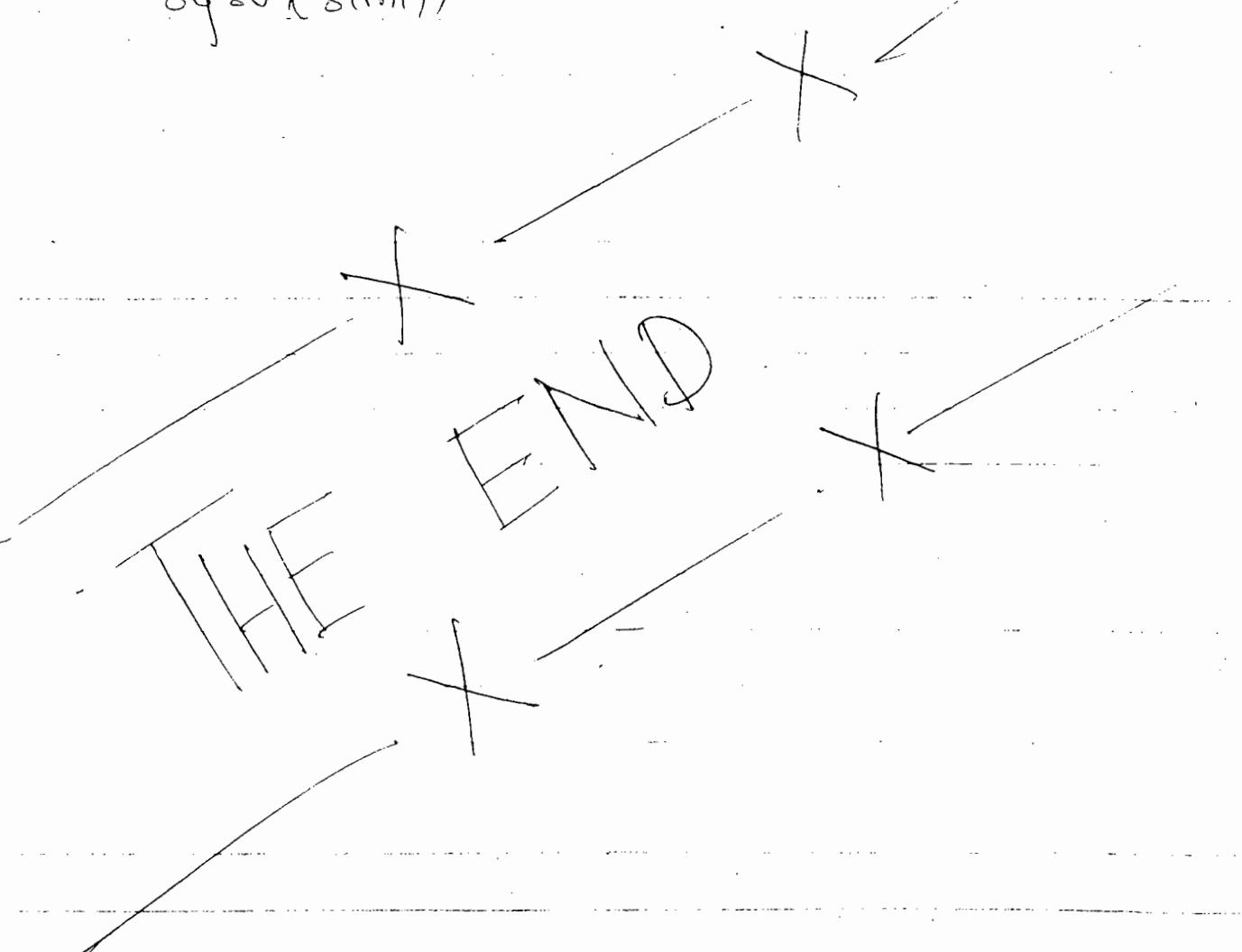
// create context

TradeService service = context.getBean( " " );

StockCalculateDTO sqd = new ( " " );

StockStatementDTO stmt = service.buyStock( sqd );

System.out.println( stmt );



public class StockQuotedTo {

StockNm

exchange

amt

panCard

personName?

//getters & setters

public class StockStatementDTO {

StockNm

units

amount

//getters & setters

c:\> proj-dir> wsdl2java -fe javabean -d6 jaxb -cl src

-client -clientjar StockExchange.jar

wsdl path.

copy jar to lib. → Build path

[Application-context.xml]

Apache CXF class

<bean id="stockExchange" class="TaxiWCFProxyFactoryBean" />

<bean id=" " />

<bean id="tradeService" class="TradeService" />

<property name="port" ref="stockExchangePort"/>

</bean>

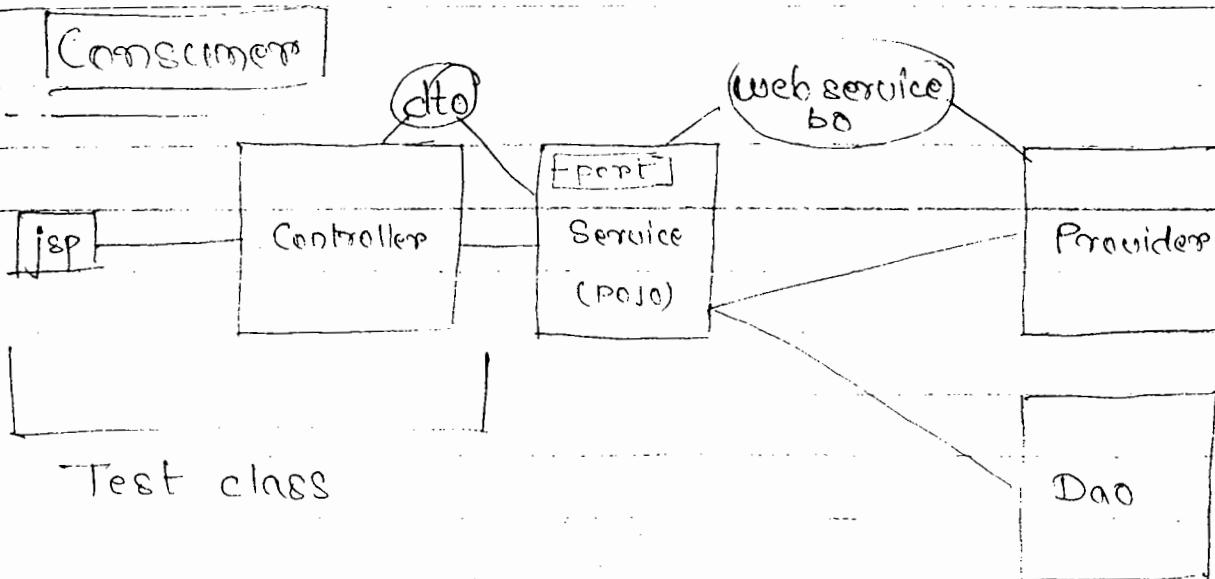
: jaxws:client id = "stockExchangePort"

service-class = "SEI-Interface"

address = "localhost: 8081 / ... "

/>

'>



→ Providers has to be injected to service. As it's external  
we'll use instance-factory method instantiation.

new project → StockClient → Paste jars.

com.sc.dto

com.sc.service

com.sc.test

com.sc.binding      ? save it      X delete

public class GreekTradeService {

private StockExchange port; // better injection

public StockStatementDTO buyStock (StockQuoteDTO stock)

// better injection of port

StockInfo stockInfo = new StockInfo();

call  
" . set ( - - ) " // write all setters

MemberInfo memberInfo = new MemberInfo();

call  
" . set ( - - ) " // write all setters

Statement stmt = port.buy (stockInfo, memberInfo);

StockStatementDTO s = new StockSt ... (stmt.get

return s;

## web.xml

<listener>

<listener-class> ContextLoaderListener

</listener>

<context-param> /WEB-INF/context-param.xml  
    ↳ beans  
    ↳ servlet.xml

<s>

<s-n> exf servlet

<s-c> CXFServlet

<los> 2

<list>

<ls-m>

<s-n> exf servlet

<u-p> /services/\*

</ls-m>

deploy on server

In impl class

↳ usual location annotation

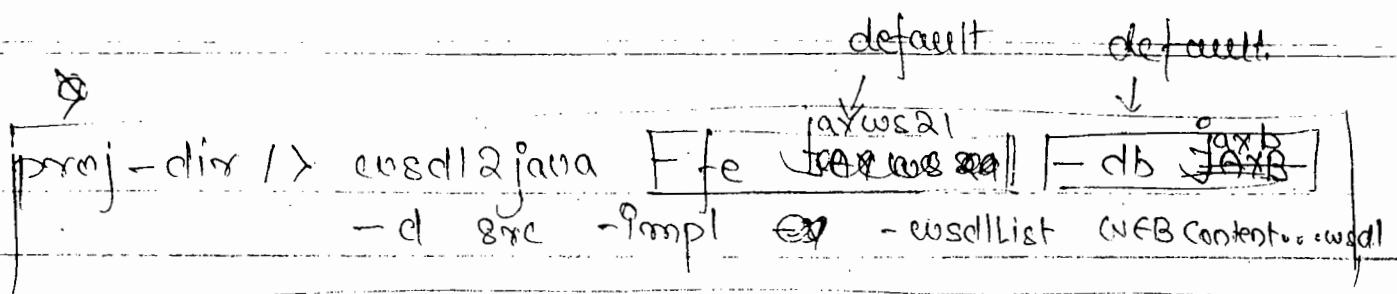
↳ remove it (OK)

↳ "/WEB-INF/- - .wsdl"

http://localhost:8081/stackweb/services → copy wsdl url

Soap ui → Test.

Apache - CXF supports JAX-WS 2.1 and JAX-RS 2.2  
JDK6 API: JDK7



proj-dir /> wsdl2java    -d src -impl  
-wsdlList    WebContent/WEB-INF/my.wsdl

Service class

```
- return = new Statement();  
-> - return.setStockNm( stockInfo.getStockNm());
```

cfg as Spring Beans

(since they're Providers so special tags are there)

CXF-beans.xml

→ import Jaxws namespace.

```
<jaxws:endpoint id="stockExchange"  
implementation = "com.. StockExchangeImpl"
```

Specifies URL for → address = "/stock" >  
exposing

</jaxws:endpoint>

with id its normal  
bean. with address  
acts as web  
service

# Contract First Approach

Dynamic Web Project → StackWeb → Tomcat 6.0.20  
→ next → wsdl

→ Copy and Paste wsdl.

→ Paste jars from apache-cxf lib.

cmd > set path = %path%; c:\apache-cxf-2.6.9\bin

cmd > proj-directory

cmd > wsdl2java -help

Apache CXF works with front end prog. [API]

↳ Jax-ws api      ↳ Jax-rs api

-fe l - frontend

-db

-P

-SD

-d

-compile

-classdir

-impl

-server

-client

-clientjars

-cull

-Keep

-wsdllocation

-asyncMethods

-baseMethods

data binding

wsdl namespace → package  
Service name.

Output directory

Generate java code.

Class files directory.

Dummy service impl

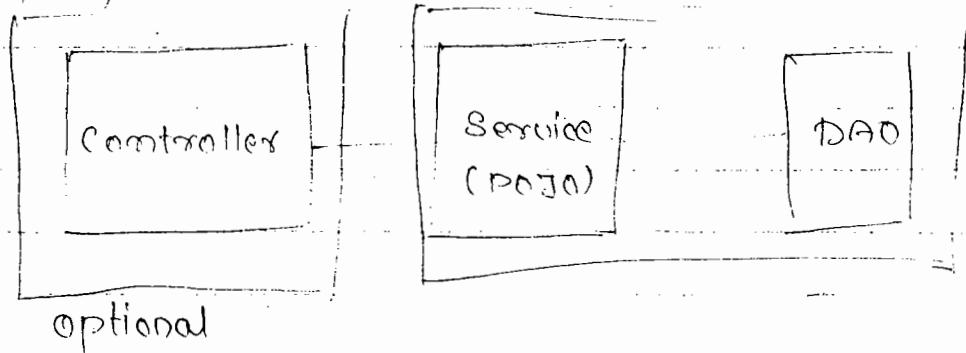
Package client + wsdl for

Server + Client + Ant script.

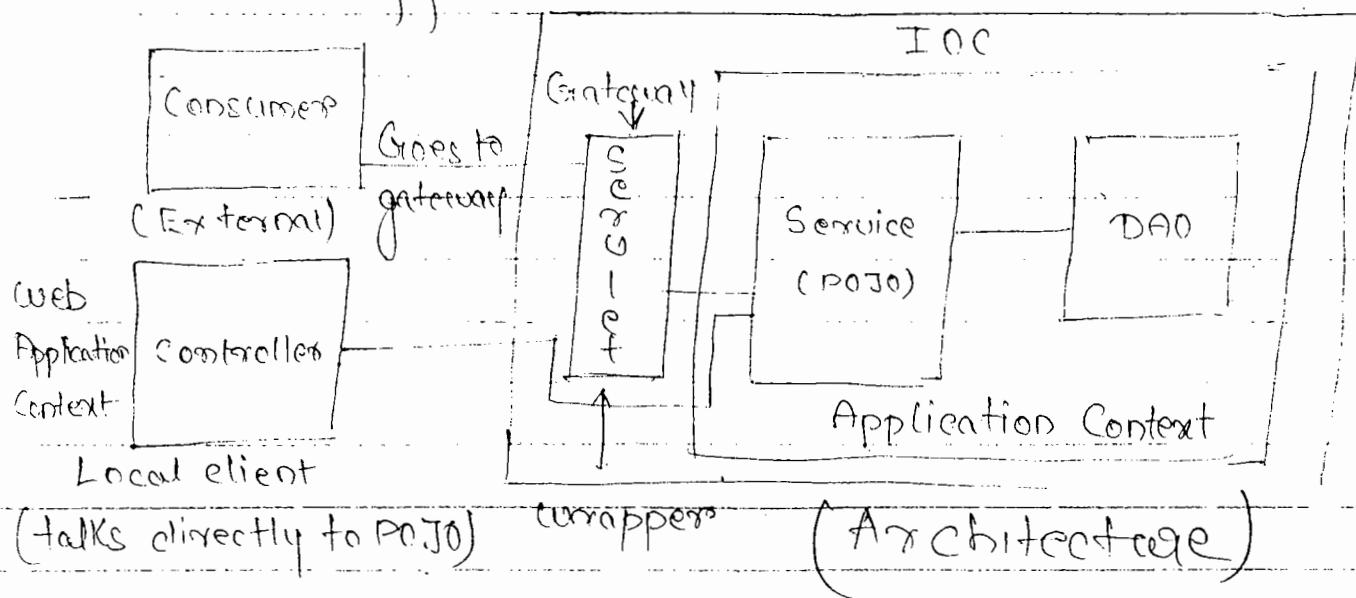
Old files overridden or not.

# How development looks

## Developing Provider :



- Req. can be sent by controller or external clients (other apps)
- Provider needn't be always web app
- DAO should be injected to Service so both should be cfg as beans.



- Servlet will try to fwd req. to one of the spring beans in IOC container.

→ Apache CXF binary dist. contains all jars for  
development + runtime tools, current ver (2.7.x)

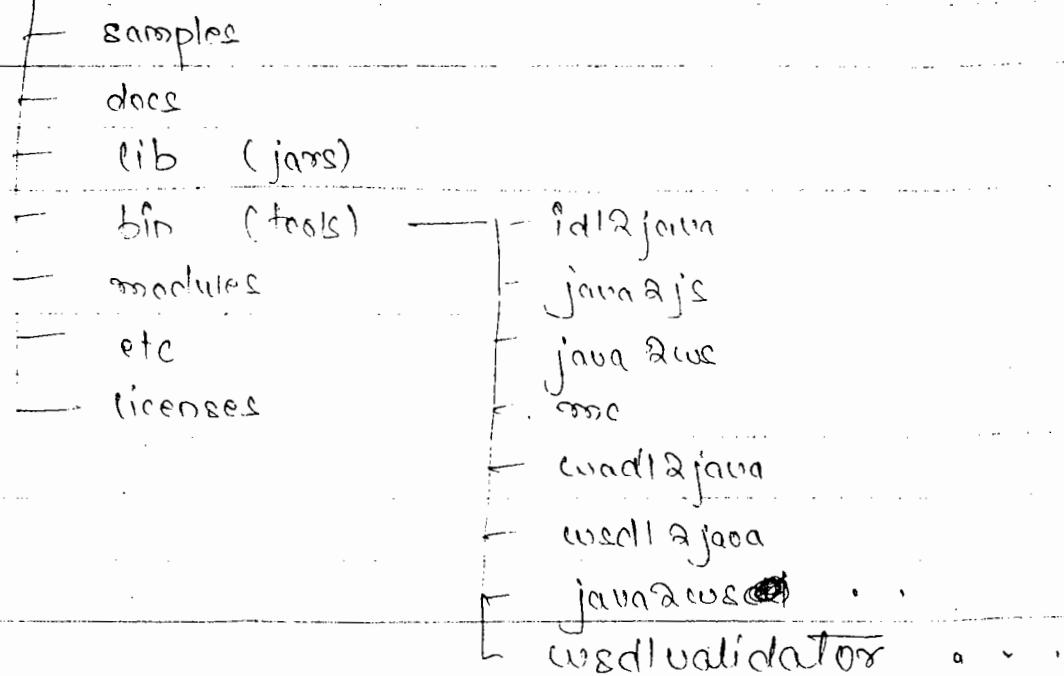
→ 2.7.9 version we're using.

apache-cxf-2.7.9.jar

### Development Env. Setup

→ Extract to c:\

apache-cxf-2.7.9



→ Contract first is preferred more.

# Apache CXF

Q) How to develop web services in Spring project?

A) Apache CXF.

→ Spring has a web service module using contract first approach and is very weak. Consumer development is very feeble.

→ You can develop web services in using Spring using Spring Web Service module. No security, transactions.

→ Apache CXF considers Spring as first class citizen for exposing web services. It exposes Spring Beans as web services. No other alternate.

→ Apache CXF is 3rd gen web service engine by Apache. It's replacement for Apache Axis2.

## Features

- ↳ Relies on Spring to expose web service.
- ↳ HTTP, JMS, JBI, RMI/IIOP, SMTP supports
- Soap + RESTful services. Not fully JAX-RS API compliant.
- Security, Atomic Tx, Transaction, Reliable Messaging . . .
- (\*) Supports various specs also, only one impl supporting.
- Very strong tooling support (Contract first & last)
- ↳ Generates minimal amt. of code to expose web services.

## SOAP vs REST

### REST :

- ① It's not protocol, it's architectural style.
- ② Completely stateless
- ③ Good caching mechanism over HTTP protocol.
- ④ Web style. (SOAP is distributed tech)
- ⑤ No standard definition language to expose interface of resource to client. (No WSDL) [response content as hypertext]
- ⑥ Suitable for few apps. (Mobile, PDA's) (Not for enterprise app)
- ⑦ Easy to integrate with existing web services, no change in existing architecture. (Just add rest)
- ⑧ Don't force message format, it can be XML or JSON
- ⑨ [Contract - last only]

### SOAP :

- ① XML based messaging protocol.
- ② Has specification.
- ③ Has spec for stateful impl.
- ④ No caching support.
- ⑤ Standard description language, WSDL, over which consumers & providers can exchange data over standard interface.
- ⑥ Less plumbing code. Used for Enterprise Application
  - + Transaction
  - + Security
  - + Addressing.
- ⑦ XML based protocol and cannot support any other format like JSON.

## Formatting Decisions

XML → Requires a greater effort to decide how to map appropriate types to XML elements & attributes.

JSON → More Direct Mapping.

## Size

XML → More overhead. Doc size length in size.

JSON → Syntax is very ~~less~~ terse & yields formatted text where ~~no~~ space is consumed.

## Parsing to JS

XML → DOM. You need to code to map text back to JS

JSON → Additional code is reqd. apart from using eval() function.

## Learning Curve

XML → More.

JSON → Less

## Tools

XML → Rich tool support.

JSON → No rich tool support

## Data Types

XML - No data types relies on schema.

JSON - Scalar data types [Group of values - Arrays & Objects]

### Support for Arrays

XML → Arrays have to be expressed by conventions.

JSON → Native Array support. ([val1, val2, ...])

### Support for Objects

XML → Obj have to be represented through conventions,  
often through a mixed use of attributes & elements.

JSON → Native Objects support

### Null Support

XML → As I said (tie it on elements) in XML  
instance document plus an import of the appropriate  
namespace)

JSON → Native support. (recognizable value)

### Comments

XML → supports & usually available through API

JSON → No

### Namespaces

XML → ✓

JSON → No. Naming collisions are avoided by nested  
objects or using prefix in an obj member.

app/\*+xml, text/\*+xml  
p/\*

org.w3c.dom.Document  
java.lang.String

### XML

- ① Extensible Markup Language
- ② Defines set of rules for encoding doc in human & machine readable format.
- ③ Has specification

→ Markup Language.

### JSON

- ① Javascript Object notation
- ② Text-based open standard designed for human-readable data interchange.
- ③ No specification.
- ④ Derived from JS objects for representing simple DS & associated arrays called Objects
- ⑤ One JSON format is often used for serializing Data Interchange

### Type of formats

#### Extended format:

xml - SGML

JSON - JavaScript

Developed by

xml - W3C

JSON - Douglas Crockford for using it at State 8/10

## Content Negotiations

which mediatypes are mapped to which data type  
of json is binding.

```
@Path("/customers")
public CustomerResource {
```

```
@GET
```

```
@Produces({AppXml, AppJson})
```

```
public Customer getCustomer(String id)
```

If Java runtime checks "accept" headers and  
mediatype are matching or not. Java runtime  
is returned Customer obj after processing then  
invokes choice in which mediatype client is  
expecting data

Content Negotiation tells what type of parameters  
can be taken for diff. mediatype.

Media Type	Java Types
app/xml, text/xml	JAXB
application/*	JAXB anno classes
json, application/* + fastinfoset	
application/atom+xml	

ie 5.5+ → JS supported

<head>

```
<script type="text/javascript">
    function findCustomer() {
        var url = null;           var cust = null;
        var id = null;
        var httpRequest = null;
        id = document.getElementById("id").value;
        alert("id:" + id);
        url = "http://localhost:8081/CacheWeb/rest/customer
               /detail/" + id;
        alert(url);             ← Modern Browser
        if (window.XMLHttpRequest) {
            httpRequest = new XMLHttpRequest();
        }
        else {                  ← IE 5.5, 6.0
            httpRequest = new ActiveXObject("Microsoft.XML
            Scripting.XMLHTTP");
            httpRequest.open("GET", url, true);           ← callback
            ↑                                     true = async, false = sync
            method                                I need to write a method, &
            httpRequest.send(null);                   tell it to check
                                                       server for response
            httpRequest.onreadystatechange = function() {
                if (httpRequest.readyState == 4) {
                    if (httpRequest.status == 200) {
                        cust = JSON.parse(httpRequest.responseText);
                        document.getElementById("resp-text").innerHTML =
                            "<b>id">" + cust.id + "<br>name :<b>" +
                            cust.name + "</b>";
```

→ JSON can be used to send data from Web Resource and Web Application.

XML

+ parsing  
+ streaming  
└ binding

JSON

+ parsing  
+ streaming

no structure = no binding.

→ Most of JAX-RS impl vendors provided ~~Reader~~ Readers & Writers to convert data to JSON.

Jettison

└ JAXB - JSON

② Consumes (MediaType.APPLICATION\_JSON)

→ For enterprise app never use JSON.

As part of web 2.0 standards W3C organization gave AJAX [Asynchronous JavaScript And XML Processing] for submitting a portion of page instead of whole page.

AJAX

<body>

  <b> Find Customer

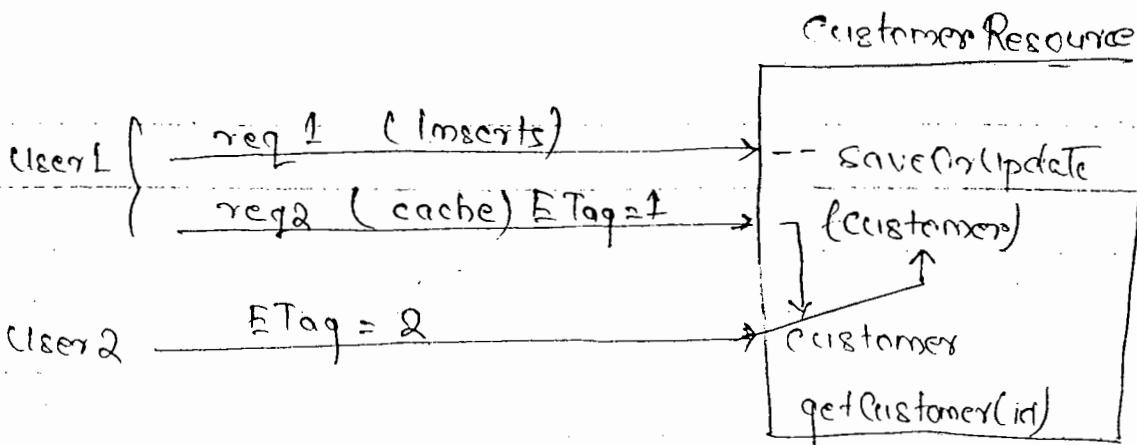
  Id: <input type="text" />

  <div id="resp-text"></div>

</body>

Id:	<input type="text"/>
<input type="button" value="Find"/>	

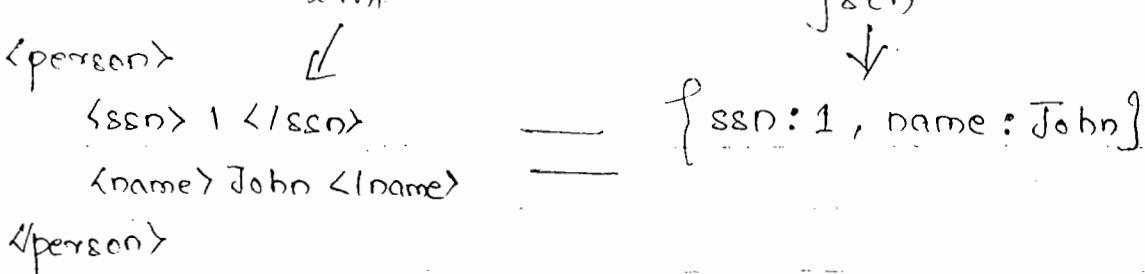
16-06-14  
Monday



## JSON [ JavaScript Object Notation ]

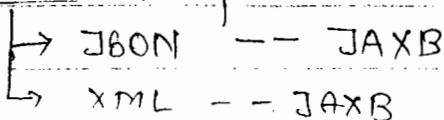
- Alternate for XML.
- To represent little info. we have to write tags. XML is verbose.
- XML steals more amt. of bandwidth

→ JSON exactly represents meta syntax of Jscript language.



→ Any XML can be represented in JSON

→ Jettison organization [open source]



→ No schema language for JSON.

→ Can't validate

response saveOrUpdate (Customer, @Context request) {

dbCustomer =

if (dbMap.containsKey (customer.getId ())) {

Customer dbCustomer = dbMap.get (customer.getId ());  
builder = request.evaluatePreconditions (dbCustomer);

else {

dbMap.put (customer.getId (), customer);

return response.ok (.....);

If (builder == null) {

dbMap.put (customer.getId (), customer);  
creationBuilder.build ();

New Header

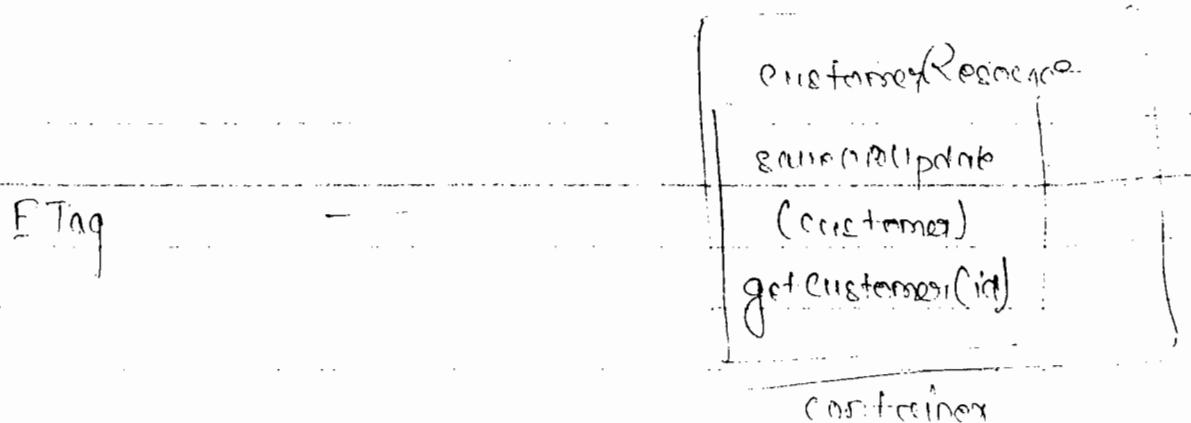
Etag — value — will  
last-modified-since — value — doesn't match

will match

→ For the first request Etag will not be checked for.

```
// data found in cache  
if (builder != null)  
    return builder.cacheControl(cc).build();  
  
return response.  
    cacheControl(cc).etag(etag).  
    build();
```

! // Try IE , Mozilla fire for giving pbms



→ I modified resource then get the data.  
Again I send update someone modified it before. Copy of ETag with me isn't correct.  
B'coz ETag value with me is old one.  
So, how to get current ETag value.

@XmlElement

```
private Date date; lastModifiedDate;  
public Customer(date){}
```

! Getters & getters

Abstraction of Model  
request send by client

```
public CustomerResponse getCustomer(int id, @Context Request request){  
    cc = new CacheControl(); cc.setMaxAge(60000);  
    customer = dbMap.get(id)  
    return response.ok().entity(customer).cacheControl(cc).build()  
        .cacheControl(cc).  
        entityTag(etag).build();  
}
```

request.evaluate

request.evaluatePreConditions(ETag) → current db  
fn

EntityTag etag = new EntityTag(cc.hashCode());

builder = request.evaluatePreConditions(etag);

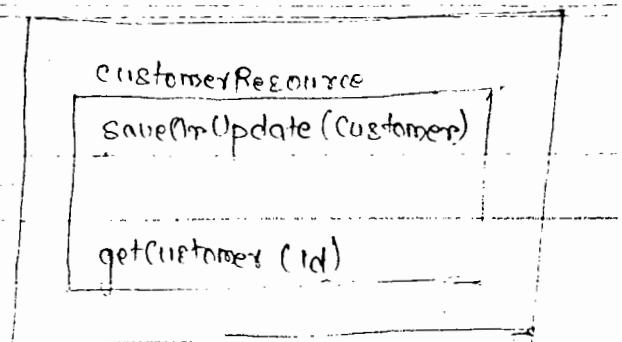
↳ Client side

responseBuilder

```
if (!builder.equals(etag))  
    return builder.cacheControl(cc).build();  
} → redirects to cache
```

// overriding hashCode() & equals() in Student.

```
new EntityTag(String.valueOf(customer).  
cc.setMustRevalidate(true));  
customer = dbMap.get(id);  
etag = new EntityTag(String.valueOf(customer.hashCode));  
builder = request.evaluatePreConditions(etag);
```



Pblm:

→ Data is cached. If modified at server we'll get stale data only

```

xml
public Response getCustomer(Integer id) {
    CacheControl cc = ee.getCacheControl();
    cc.setMaxAge(60000); // 60 sec
    cc.setPrivate(true);
    // cc.setNoStore(true);
    return response.ok().entity(dbmap.get(id)).cacheControl(cc)
        .build();
}

```

}

→ check whether data is modified or not at server. we'll save bandwidth if not modified as we'll not fetch from server instead redirect to cache.

① How server knows last access and stale data are not in cache?

Ⓐ Last-Modified-date:

(or)

Etag: [unique no generated based on current state of obj (Hash value)] (unique tag identifying entity.)

Entity Tag

com.eco.util

@ApplicationPath("/rest")

public class RestApplication extends Application

singletons.add(new CustomerResource()));

// getSingletons()

public class CustomerResource {

// Modify

private Map<Integer, Customer> dbMap;

@POST @Path("/{saveOrUpdate}") @Consumes(MediaType.APPLICATION\_XML)

public String addCustomer(Customer customer)

dbMap.put(customer.getId(), customer);

System.out.println("Customer added");

@GET @Produces(MediaType.APPLICATION\_XML) @Path("/{id}")

public Customer getCustomer(Integer id) {

return dbMap.get(id);

/rest/customers/{saveOrUpdate}

/rest/customer/{detail}/

<customer>

<id>1</id>

<name>john</name>

</customer>

@Path("/customer")

public class CustomerResource

@GET @Path("/{name}/{id}") @PathParam("id")  
public String getCustomerName(@PathParam("id") id) {  
 return "Customer [" + id + "]";

cal = Calendar.getInstance();  
cal.set(2014, 15, 6, 6, 15, 0);  
Y D M H Min Sec

Response.ok(entity("N.T Rama Rao")).expires(  
cal.getTime().build());

/rest/customer/name/10

cal.set(Calendar.YEAR, 2014);  
" " MONTH(5); → 0-11  
" " DAY\_OF\_MONTH(15);  
" " HOUR(6);  
" " MINUTE(15);  
" " SECOND(0);

@Path("/{firstname}/{id}") @GET @Produces("text/plain")  
public String getFirstName(@PathParam("firstname") Response

CacheControl cc = new CacheControl();

cc.setMaxAge(10); // 10 sec

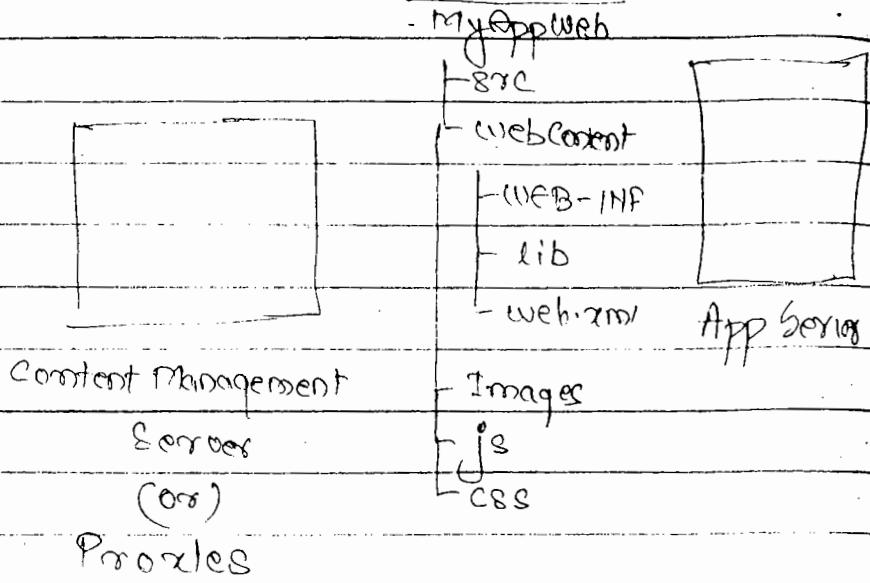
cc.setNoStore(true);

cc.setPrivate(true); // Only client store not intermediate

// cc.setNoCache(true); // Checks server first then cache

return Response.ok().entity("Rama Rao").cacheControl(cc);

build(); ↑  
Set cache control header also



→ Images, js... are static files if you load them in memory space will be wasted. So, put them in Content Management Server.

noCache : Always don't use directly if in cache then evaluate it's fresh data or not from server

~~don't use~~

CacheWeb → copy (checkbox).

com.caucho.resource

com.caucho.domain

① XMLRootElement (name = "customer")

② XMLProcessorType (XMLAccessType field)

③ XMLType

public class customer {

④ XMLElement

int id;

String name)

```
public class StudentResourceTest {
```

```
    public void test() {
```

```
        StudentResource resource = new StudentResource();
```

```
        resource.register(10, "john");
```

## Caching

- Scaling up of system as mtg is stored on servers.
- Resilient principle: Communication Stateless.

HTTP 1.0 — minimal support for caching

↳ expires : 15 Jun 2014 ; 06:00:00 IST

headers

data expires at 6'o clock  
today upto 10:00 PM  
In my cache.

HTTP 1.1 — Better support

↳ cache-control (new header introduced)

↳ parameters are

↳ max-age

↳ no-store

↳ private

↳ no-cache

cache-control : max-age:10m; no-store; private; no-cache

## Using Programmatic Security

@Path("/bank")

```
public class BankResource {
```

@Context

```
private SecurityContext context;
```

```
getBalance() {
```

```
if (!context.isUserInRole("admin")) {
```

```
throw new ForbiddenException("Unauthorized user");
```

```
}
```

## Sending Form data

```
public Rating register(int id, String name) {
```

```
Form form = new Form();
```

```
form.param("id", String.valueOf(id));
```

```
"name", name);
```

```
target = client.target(BASE + "/{id}");
```

```
accept(MediaType.TEXT_PLAIN);
```

```
POST(Entity.form(form), ApiResponse);
```

```
if (response.getStatus() == 200) {
```

```
details = response.readEntity(String.class);
```

```
return details;
```

```
} return null;
```

```
}
```

Enter the details of new user to add

Realm (Application Realm) : ↴

Username: user1

password: user1

re-enter: ↴

• Role is Admin ↴ or Moderator or Normal

Access to specified resource has been forbidden

HTTP Status 403 ↪ if username & password don't match.

## Using Annotations

@Path("/bank")

for all methods

@RolesAllowed({"admin"})

public class BankResource {

  @RolesAllowed({"normal"})

  public void float getBalance()

}

In web.xml

<login-config>

<auth-method>

<realm-name>ApplicationRealm

overriding for specific method

@ApplicationPath("/rest")

public class SecurityApplication extends Application

// override getSingletons

}

→ Every container doesn't support annotations (inherent security).

→ If web.xml and annotations both are defined then web.xml takes priority.

## web.xml

<security-constraint>

<web-resource-collection>

<web-resource-name> /rest/bank/\* Bank Resource Authorization

<url-pattern> /rest/bank/\*

<http-method> GET

<web-resource-collection>

<auth-constraint>

<role-name> admin

<role-name> moderator

<auth-constraint> </security-constraint>

<login-config>

<auth-method> BASIC

Authorization → role

Only admin &

moderator can

access that

- normal users.

Authentication.

<login-config>

<security-role>

<role-name> admin

<security-role>

<security-role>

<role-name> moderator

<security-role> // Add normal also

realm → Grouping users  
into groups.

Marketing realm

Sales realm

Developers realm

<web-app>

For better mgmt.

c:\> cd jboss-as-7.1.1.Final\bin

bm > add-user.bat

What type of user do you wish to add?

a) Management user

b) Application user

(a) : b

## Caching

- Using ResponseBuilder to cache
- Working with CacheControl

## Working With Security

- In restful services security isn't great. There is no way to impl enterprise security. It has to rely on Basic Authentication recommended by http in which username & password are sent with url.
- You rely on web.xml config as we use some container. So, we'll use JBoss security mechanism.

### Ways to work with Restful Services Security:

- ① Configuration
  - ② Annotation
  - ③ Programmatic
- 3 ways

## Caching Configuration

```

@Cacheable
@Path("/bank")
public class BankResource {
    @Path("/{account}")
    @Produces(MediaType.TEXT_PLAIN)
    @GET
    public String getBalance(@PathParam("account") String
                           accountNo) {
        return "Rp (" + accountNo);
    }
}

```

```
Response response = future.get();
if (response.getStatus() == 200)
    soap(response.readEntity(String.class));
}
```

## // Callback

```
public class StockResourceCallbackClient {
    public void(-) {
        target = client.target(BASE - (URI));
        target.path().resolveTemplate().request().async()
            .get(new StockResponseCallback());
        } ↳ registers with runtime
    }
}
```

```
private final class StockResponseCallback implements
    InvocationCallback<Response> {
    }
```

```
public void completed (Response response) {
    soap(response.readEntity(String.class));
    }
```

```
public void failed (Throwable throwable) {
    throwable.printStackTrace();
    }
}
```

# Asynchronous Client Consumer

→ If BaaS pushes you to take longer time then use  
async consumer.

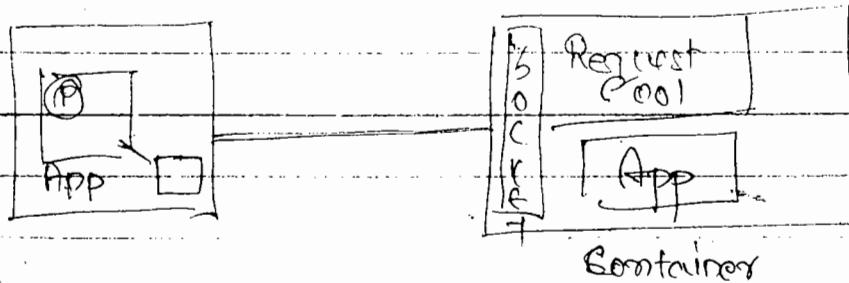
## [Asynchronous Consumer Types]

F Polling

F Callback

Polling → Prog. asks every few interval of time to server  
if response is ready or not. we have to code it.

Callback → Prog. registers one obj to runtime, runtime  
will ask for response to server. When response is ready  
runtime calls the class.



```
public class StockPriceConsumerClient {  
    private String BASE_URI = "http://localhost:8081/ASyncResponse/  
    rest/stock";  
}
```

```
psvm(-){  
    builder = ClientBuilder.newBuilder();  
    client = builder.build();  
    target = client.target(BASE_URI);  
    Future<Response> future =  
        target.path("/path/{stockNm}").resolveTemplate("stockNm", "ICICI");  
    request().async().get();  
    // We're polling using Future obj.  
    while(!future.isDone()) {  
        Thread.sleep(10);  
    }  
}
```

private void ~~onMessage~~(

private final class StockExecutor implements Runnable {

private AsyncResponse asyncResponse;

// cons injection.

public void run() {

S.O.P ("Big work");

try {

Thread.sleep(1000);

} catch (InterruptedException) {

}

} a.

@Path "/prices/{stockNm}")

public void getStockPrice(@PathParam -- ,

@Suspended AsyncResponse asyncResponse))

StockInfoExecutor executor = new

StockInfoExecutor (asyncResponse);

new Thread (executor).start(); // & returns control  
// back destroying the  
asyncResponse.resume (Response.OK ("224")); caller  
thread

→ req. comes to container. First it goes to socket container then the threads are placed in waiting state in Thread pool and after the servlet container process the req., returns response to sockets. The socket upon successful resp will close connection.

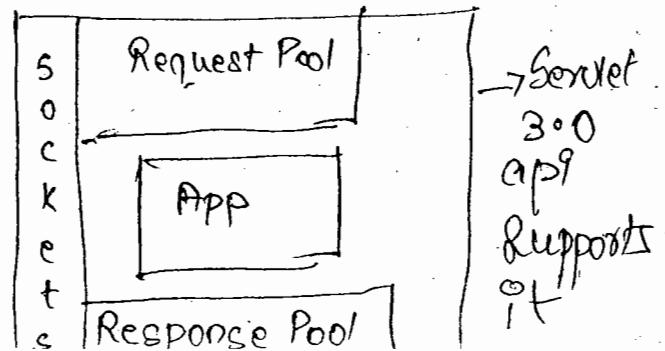
→ If TimeToLive in thread pool is more than new threads coming is more & threads executing speed will be less.

→ Servlet api 3.0 has async. capability.

→ Now there're two diff phases

[req. phase  
response phase]

→ Socket receives req. & gives to Request pool & then goes for processing. After the thread is given to app. the Request pool will be released. After processing response goes to Response pool now.



↓ ↓ ↓  
http request  
use only when app takes  
time to execute.

```
ArrayList<AsyncResponse> responses = null;
```

```
public void resource()
```

```
responses = new ArrayList<AsyncResponse>();
```

```
}
```

```
}
```

web.xml

```
<s>
```

```
<s-n> resteasy
```

Asyc Preq is enabled  
(else wait pool willnt be there)

```
<s-c> HttpServlet30Dispatcher
```

```
<async-support> true </async-support>
```

~~```
<s-p> resteasy-ecas
```~~

```
<f-p> resteasy-servlet-mapping.prettix -- treat.
```

```
<!--> !
```

```
</s>
```

```
<s-m>
```

```
<s-n> resteasy
```

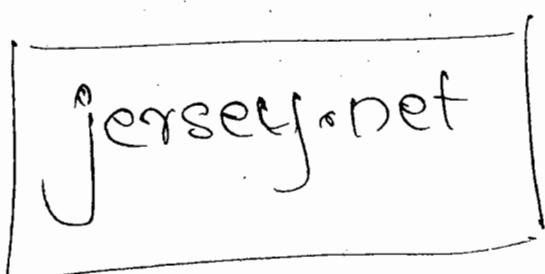
```
<u-p> /rest/*
```

```
</s-m>
```

→ <async-support> is jboss tag.

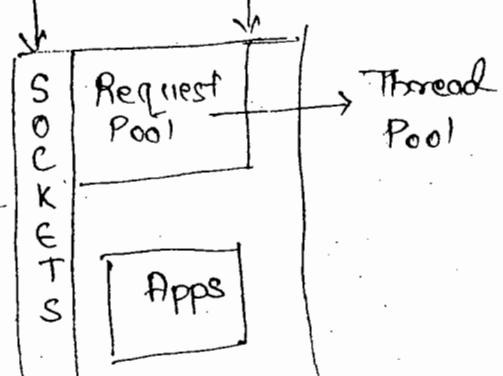
↳ jboss resteasy specific cfg.

/rest&#8226;/stock/cicici&#8226;/topic



http request

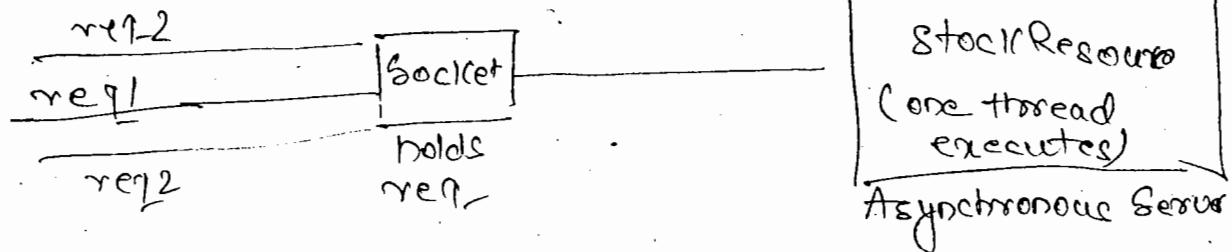
Transport Container  
Servlet/EJB Container



↓  
ELB

Prob] Repeated req are coming for same resource  
in server making it load.

Sol] Build a resource that collects many req, instead  
of creating new thread only one thread will  
be created.



## Asynchronous Processing at Server Side

Dynamic web Project → Async Resource → code(m)

com.eom.resource

① Path("stock")

public class StockResource {

② GET  
③ Produces (text/plain)

④ Path("price/{stockNm}")

public float getStockPrice (@PathParam("stockNm")

void

String stockName;

⑤ Suspended

return 242.24f;

responses.add(response);

} new Thread() {

public void run()

sop("performing..");

??.start();

⑥ AsyncResponse response) {

Suspends current

thread and add to

queue and do the

processing in separate

thread and return

for (AsyncResponse asynresp: responses) { the result to all

asynresp.responseText(Response.out("242.24").build()); the suspended thread

9963/11226

D K W W

V R O X

9963/11226



```

public Customer getCustomer (String ssn) {
    try {
        builder = ClientBuilder.newBuilder();
        // Register
        target = client.target (Base_URI);
        Response response = target.path (" /final/{ssn}").g. resolveTemplate ("ssn", ssn).g
            request ().accept (Text_plain).get ();
        if (response.getStatus == 200) {
            customer = response.readEntity (Customer.class);
        }
    } catch (Exception e) {
        finally {
            return customer;
        }
    }
}

```

## Asynchronous JAX-RS API

Prob1: Processing request if timeout occurs

Prob2: More than 60ms it shouldn't increase timeout.

Sol1: Build a consumer who collects response in another thread releasing the first.

```
public void addCustomer (Customer customer) {
    try {
        builder = ClientBuilder.newBuilder();
        builder.register (CSVMessageBodyReader.class);
        builder.register (CSVMessageBodyWriter.class).build();
        target = client.target (BASE_URI)
            .accept (MediaType.TEXT_PLAIN);
        Response = target.path ("/new").request ().post (Entity
            .entity (customer, MediaType.TEXT_PLAIN));
        if (response.getStatus () == 200) {
            System.out.println (response.readEntity
                ());
        }
    } catch (WebApplicationException e) {
        finally {
            client.close ();
        }
    }
}
```

```
class CustomerResourceTest {
    private CSVResource csvResource;
    private Customer customer;
    private String name;
    private String id;

    @Before
    public void setup () {
        customer = new Customer ();
        customer.setName ("John");
        customer.setId ("12345");
    }

    @Test
    public void testAddCustomer () {
        csvResource.add (customer);
        Customer foundCustomer = csvResource.getCustomer (1);
        assertEquals (customer, foundCustomer);
    }
}
```

## Accessing Headers

```

private Client client;
public String getStudentDetail (String sid) {
    return details;
}
public StudentResource() {
    client = ClientBuilder.newClient();
}

public String getHeaders() {
    target = client.target(BASE-URI);
    Response res = target.path("fullHeaders").request().headers("myHeader", "header1").get();
    return res.readEntity(String.class);
}

protected void finalize() {
    client.close();
}

```

## class StudentResourceTest

```

StudentResource res = new StudentResource();
System.out.println(res.getHeaders());

```

Sending Object as Input and Object as output.

## com.csvecustomcontenthandler.client

- domain `com.messageBodyReader`
- handlers `com.messageBodyWriter`
- common - `CSVType`

## class CustomCSVResource

```

BASE-URI = "http://localhost:8081/CSVContentHandler/test/customer";

```

```
response.bufferEntity();
```

```
String plInfo = response.readEntity(String.class);
```

```
s.o.p(plInfo);
```

Can be

called any  
no. of times

### com.ws.injection.client

```
class StudentResource {
```

```
private String BASE-URI = "http://localhost:8081/webservices/injectionrest/  
student";
```

```
/x    private Client client;
```

//client has no state. So little connection, so create one.

```
public StudentResource()
```

```
    client = ClientBuilder.newClient();
```

```
} -> try {
```

```
    Client client = ClientBuilder.newClient();
```

```
    Target target = client.target(BASE-URI);
```

```
Response response = target.path("detail/{sid}").  
                    resolveTemplate("sid", "81");
```

```
request = request.cookie(newCookie("cid", "c1")).get();
```

```
if (response.getStatus() == 200) {
```

```
    String details = response.readEntity(String.class);
```

```
s.o.p(details);
```

```
} finally { client.close();
```

```
}
```

```
{
```

#2<sup>nd</sup> way

Invocation invocation = target.request().accept(MediaType.TEXT\_PLAIN).buildGet();

Response response = invocation.invoke();

if (response.getStatus() == 200) {

String quote = response.readEntity(String.class);

System.out.println(quote);

}

class CustomerResourceClient

private String BASE\_URI = "http://localhost:8081/BindingURIAndsubRequest/customer";

public void getCustomer() {

client = ClientBuilder.newClient();

target = client.property("connectiontimeout", "500");

target.target(BASE\_URI);

Response response = target.path("{name}").resolveTemplate("name", "John").

matrixParam("height", "6.5").request().get();

→ Status.OK (Enum) API problem

if (response.getStatus() == 200) {

String personalInfo = response.readEntity(String.class);

can be called only once  
then closes.

Response

re

Builder level environment method

get() } Each time new ~~Invocation~~ Invocation obj is  
post } created. No reusability.

buildGet() } Optimized way. Reusability.  
buildPost()

Invocation invk = builder.buildGet();

invk.invoke();  
invk.invoke();  
invk.invoke();

Icomonsuri-subresources-client ]  $\xrightarrow{\text{2nd Resource}}$

public class carResource

param( )

private static final String

BASE\_URI = "http://localhost:8081/Binding(IRI)andSubRes/  
res/car/{model}";

param( )

builder = ClientBuilder.newBuilder();

client = builder.build();

target = client.target(BASE\_URI);

// appending sub resource path

target = target.path("/{enquiry}");

target = target.resolveTemplate("model", "swift");

" " " " " ("enquiry", "m-")";

// First way

// Direct query param.

```
target.queryParam("name", "Raja");
```

// access Resource class method.

```
target.getResource().get();
```

But  
at the class level. Using it we can  
use any subresources of  
Resource class

~~interface~~ interface Invocation {  
 interface Builder {  
 buildGet(), get()  
 buildPost(), post()  
 buildPut(), put()  
 buildDelete(), delete()  
 }  
}

creates Invocation obj by populating  
respective data for it.

initTarget / creates Request Builder obj

```
response = target.request().accept(MediaType.TEXT_PLAIN).  
           .get();
```

// For 10 req, we have to call 10 times the ~~get()~~ get()  
with new data

// For Invoking only one time

Invocation invocation = target.request().accept().language().  
 buildGet();

```
invocation.invoke();
```

```
if (r.getStatus() == 200) {
```

```
    String message = response.readEntity(String.class);
```

```
    System.out.println(message);
```

```
}
```

FILE

WebTarget Targets a specific ~~area~~ or ~~service~~ URL From jax-rs api ~~API~~

client creates connection pointing to WebTarget  
(creates WebTarget obj)

ClientBuilder (A.C) — newBuilder  
↳ knows how to create Client obj

↳ knows how to create WebTarget obj

ClientBuilder — timeout [ ] For all clients  
client — [ ] Specific clients

configuration(I)  
— property()  
— register()  
...  
impl  
impl  
impl

ClientBuilder  
client  
WebTarget

register(class) — Per req new obj;  
register(object) — for singleton

→ ~~Start~~ Refer E.g. 1. (Start server) → Create RSCClient  
resteasy jars final → lib → copy all jars → paste

in new folder lib → com.firetree.client

public class ClientClient {

psvm(-){

ClientBuilder builder = ClientBuilder.newClient();

or

client = builder.build(); // creates client with something

static method to create  
client with empty obj

clientBuilder builder = ClientBuilder.newBuilder()

builder.property("connection.timeout", "500");

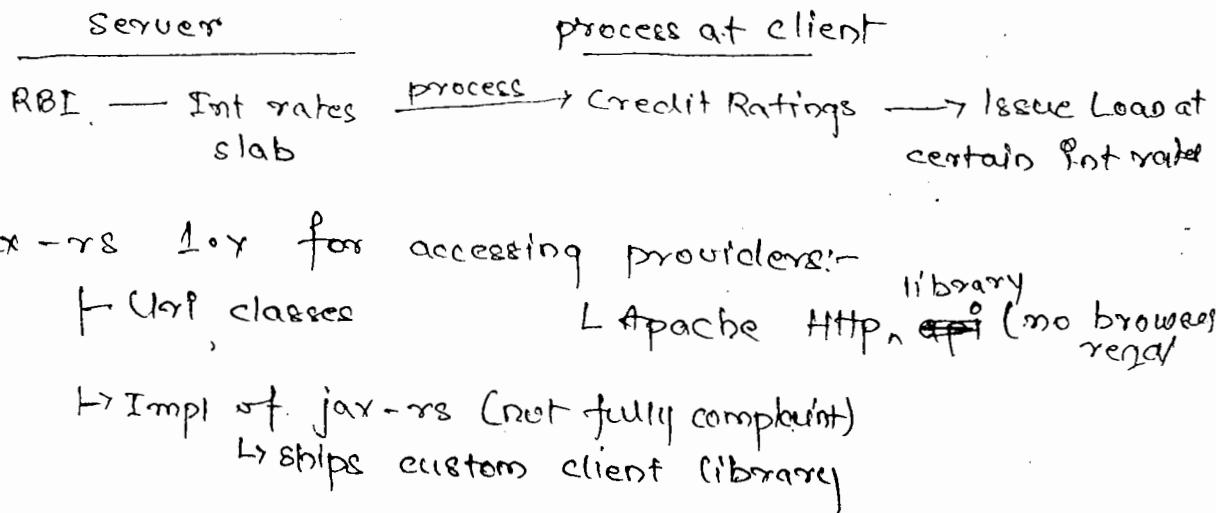
client = builder.build();

15-06-19  
Sunday

jax-rs 2.0

## Working with Jax-rs client

- jax-rs 1.0 gave classes for creating providers only.
- jax-rs 2.0 gave api for building clients and provider;



⑥ What're the things client can put as part of request?

A) client

- ↳ transport semantics (can timeout...)
- ↳ http headers
- ↳ cookies
- ↳ template parameters (substitute it before sending)
- ↳ query parameters
- ↳ Matrix parameters
- ↳ <sup>orignal</sup> Form data
- ↳ entities (data) (content-handlers) (File, byte[], InputStream, Reader, MultivaluedMap, Object)
- ↳ accept headers
- ↳ register (providers)

- get  
- post  
- put

```
class AppointmentDelegate {
```

```
    public void bookAppointment (String work) {
```

```
        throw GenericException ("unable to ..");
```

```
}
```

```
class GenericApplication extends RuntimeException {
```

```
    String reason;
```

```
    // cons (reason) {           // 0-param also.
```

```
}
```

```
    // better getters
```

```
{
```

→ If Resource class has 10 methods then we

23/2

need to wrap exception at 10 places. It's risky.

@Provider ← registers with jaxrs runtime

```
class GenericExceptionMapper < implements ExceptionMapper {
```

```
    public Response toResponse (GenericException ge) {
```

```
        return Response.status (Status.SERVICE_UNAVAILABLE).
```

```
            entity (ge.getReason()). type (MediaType.TEXT_PLAIN)
```

```
            build();
```

```
class CourierNotFoundException extends Throwable {  
}  
}
```

```
class CourierNotFoundExceptionMapper implements  
ExceptionMapper<CourierNotFoundException> {  
}  
}
```

ExceptionWeb  
└ src  
 └ resource  
 └ dao  
 └ util

```
@Path("/appointment")  
class AppointmentResource {  
    @POST  
    @Produces("text/plain")  
    @Path("/add/{work}")  
    String addAppointment(@PathParam("work") String work)  
}
```

try

```
AppointmentDelegate = new AppointmentDelegate()
```

```
delegate.bookAppointment(work);  
} catch(GenericException) { new ServiceUnavailableException(); }  
return "@ \"booked\"";
```

503 service  
unavailable

→ Always `Book()` isn't executed successfully but it ~~throws~~ throws exception back to client. But, always it's not good to return 500 server error. Based on error we should give diff. status code for every status code, ~~the~~ jax-rs provided Exception classes. But, try catch will duplicate code for catching.

-T

me

@  
cl

→ Always throwing webApplicationException will not give appropriate status code

② Path ("courier")

```
class CourierResource
```

```
    public String track (Courier courier) {
```

```
        Status status = null;
```

```
        try {
```

```
            //db query → HB specific
```

```
        } catch (NotFoundException e) {
```

```
            throw new BadRequestException (e);
```

```
}
```

```
}
```

→ we need to catch & throw every place where NotFoundException occurs so code

duplication in app.

→ So, better way is Exception Mappers

Response.status(—).entity().type("application").build();

→

or

## Exception Handling

①

etc

WebApplicationException by default  
JAX-RS binds it to 500 status code

@Path("/couriers")

public class CourierResource {

public String bookParcel(ParcelInfo —) {

if (parcelInfo.getZip() == allowableZip) {

throw new WebApplicationException();

}

9

wt

9

du

WebApplicationException (Root)

→ Sc

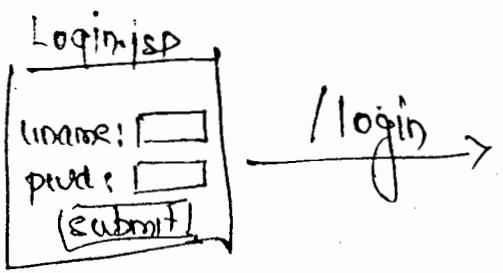
EntityNotFoundException

BadRequestException

ForbiddenException

InternalServerErrorException . . .

ServiceUnavailableException



e'  
 class LoginServlet {

//authenticates

// find cookie

if (lastVisitedPage.equals("viewPlans")) {

lastPlans new ModelAndView

// redirect: viewPlans.htm");

}

}

public Response recharge(MobileInfo) {

Cookie = new Cookie("mobileNo", mobile.getMobileNo());

builder = builder.cookie(cookie);

Discards body ↙

```
response = Response::status(StatusCode::OK).entity(entity).build();
```

• SERVICE\_UNAVAILABLE ✓

↑  
503 Service unavailable

eBay

Add to cart

→ cart id is generated and stored in db  
with ~~cart~~ same cart id no. for all the  
items and stores as cookie in your  
browser. Now, session isn't used.

→ To make cookie cross platform for every  
browser ~~use~~ ~~not~~ associate cart with user Id  
& store it in another db table.

```
public Response recharge (MobileInfo mobile) {
```

response body  
↓

```
    ResponseBuilder builder = Response.noContent();
```

```
    builder.header ("network-type", "gsm");
```

```
    Response response = builder.build();
```

```
    return response;
```

```
}
```

@POST  
@Produces ("application/xml")  
@Path ("mobile/{mobileNo}/serviceMsg") @PathParam ("mobileNo")  
public StatusInfo activateService (String mobileNo,  
 @QueryParam ("serviceMsg") String serviceMsg) {

```
    StatusInfo si = new StatusInfo();
```

```
    si.setMobileNo (mobileNo);
```

status code  
200 OK

```
    si.setStatus ("Failed");
```

```
    return si;
```

```
}
```

mobile/activate/911111/enableGps

```
public Response activateService (- - -) {
```

```
    response = Response.notModified (), entity (si).build ();
```

creates Builder with  
static code not modified

adds  
data to  
return

```
}
```

return response;  
304 Not Modified

# Factory vs Builder

factory.newDatetimeBAXParser();

No inputs → empty obj is created  
↳ After it set values to it!

## Builder

↳ entity, status code, headers, cookies  
↳ (Object).Response

new Dynamic web project

ResponseWeb

copy web.xml

com.rw.resource  
com.rw.domain

class MobileInfo

mobileNo;

amt;

}

@XmlRootElement(name="statusInfo")

@XmlType

@XmlAttribute (FIELD)

class StatusInfo

@XmlElement

mobileInfo;

status; //setters &

getters

}

@Path("/mobile")

public class MobileResource {

@Path("recharge")

@Consumes(MediaType.APPLICATION\_XML)

@POST

public void recharge(MobileInfo mobile)

System.out.println("recharged. - "));

return  
JAX-RS appends

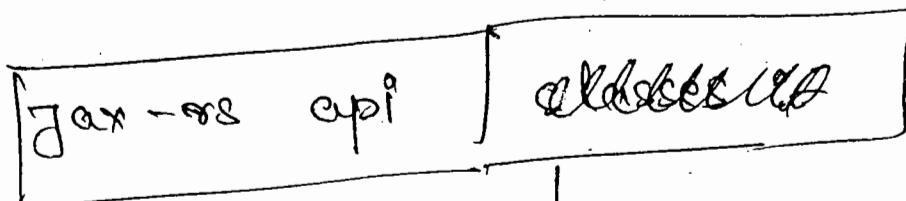
204 status code

status code is mandatory rest optional.

## Response

- ↗ entity
  - ↗ http status code
  - ↗ http headers
  - ↗ cookies
  - ↗ Language
  - ↗ encoding
- ↗ content type  
↗ request body

→ As part of response sometimes we need to return the above part of response



↳ Response (A.C) ↳ ResponseBuilder (AC)

~~factory~~

↳ ResponseBuilder (AC)

- build() → creates Response obj

Static method

- - OK()
- - ServerStatus error
- - ServerStatus( )
- - noContent()
- - notModified()
- - entity()
- - status()

anti-pattern

Bug of API  
(creates obj)

Ex: `Response.OK()` → ResponseBuilder obj  
will code status code

200

build() → creates Response obj with

logically appropriate status code.  
I think

[`ResponseBuilder builder = ResponseBuilder(200);`] ]

⑥ Context

Private Providers provides,

Content Resolver (JAXBContext) resolver = providers.

getContentResolver (JAXBContentResolver.class  
mediatype);

jcontext = resolver.getContext (classType);

## Server Responses and Handling Exception

### Server Responses

should execute successfully

→ By default jax-rs resource method by default

returns ~~status~~ status code to client.

interface CustomerResource

public String addCustomer()

public void updateCustomer()

HTTP is sync,  
reply

Exception  
occur

GET, PUT, POST, DELETE

200 - OK

204 - no content

500 Internal Server Error

## Jar - vs Interface

## Interface Providers

ContextResolver getContexResolvers();

MessageBodyReader getMessageBodyReader(L)

MessageBodyWriter getMessageBodyWriter();

1

for accessing registered Provider.

⑥ Context- Providers providers; } In ~~XMLBoo~~  
} XMLMessage Body writer

```
writeTo () {
```

```
jContext = providers.getContentResolver(classType,  
mediaType);
```

ContextResolvers < JAXB context > resolver = jContext.

```
getContentResolver (Intent,  
Uri,  
MediaType);  
(classType);
```

```
finally {
    out.close();
}

readFrom() {
try {
    String rawDate = extractRawDate(inputStream);
    Map<S,S> genericDataMap = convert(rawDate);
    Object object = build(genericDataMap, classType);
} catch (...) {
    throw
}
return obj;
}
```

POST /rest/customer/new.

~~SSN-XML~~  
SSN=10, name=John

Address → Try Text-PLAIN  
\*/\*

private String buildResponse (Class<Object> classType,  
Object object) { obj contains  
StringBuffer buffer = new StringBuffer();  
boolean isFirst = true;

field[] fields = classType.getDeclaredFields();

for (Field field : classType.getDeclaredFields()) {

for (Field field : classType.getDeclaredFields()) {

private  
Field can  
be accessed

field.setAccessible(true);

if (isFirst) { buffer.append(field.getName()); append(":"),

append(field.get(object));

}

else {

buffer.append(","); append --- same

}

isFirst = false;

} return buffer.toString();

}

writeTo () {

try { out = new PrintWriter(--);

content = buildResponse (classType, object);

out.print(content);

} catch (...) {

throw new WebApplicationException (e);

```
for (String attr : dataMap.keySet()) {
```

```
    method = getSetterMethod (classType, attr);
```

```
    method.invoke (object, dataMap.get (attribute));
```

```
getSetterMethod (Class<Object> classType,
```

```
public Method getMethod (String attribute) {
```

```
    String methodName = "set" + attribute;
```

```
    Method[] methods = classType.getDeclaredMethods();
```

```
    for (Method m : methods) {
```

```
        if (m.getName().equals (methodName)) {
```

```
            return m;
```

```
    } return null;
```

```
@Provides
```

```
@Produces (wildcard)
```

```
class BsuMessageBodyWriter implements MessageBodyWriter<Object> {
```

```
    IsWriteable (-) {
```

```
        if (classType.isAnnotation (BsuType.class)) {
```

```
            return true;
```

```
            return false;
```

field

private  
attr c  
be ac

Map<String, String> convert  
private Object build (Class<Object> classType,  
String data )

Map<String, String> genericDataMap = new HashMap<, >();

StringTokenizer tokenizer = new StringTokenizer (data, " , ");

String parts [] = null;

String token = null;

while (tokenizer.next.hasMoreTokens ()) {

token = tokenizer.nextToken ();

parts = token.split (" = ");

genericDataMap.put (parts [0], parts [1]);

}

return genericDataMap;

}

public build (Map<String, String> dataMap,  
Class<?> classType) {

Object = classType.newInstance (); // customer obj

Method method = classType.getDeclaredMethod

("methodName", orgType);

or

⑥ Consumes (wildcard)

⑦ Provider

class CSVMessageBodyHandler implements MessageBodyReady<Object>

isReadable( . . . ) {

if (ClassType.isAnnotationPresent(CSVType.class))

return true

}

return false

}

readFrom( ) { extractRawData

String extractData

private Object build(InputStream inputStream) {

⑧ buffer = new StringBuffer();

bis = new BufferedInputStream(inputStream);

while ((c = bis.read()) != -1) {

buffer.append((char)c);

return buffer.toString();

}

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.TYPE) → Only used as class level.

```
public @interface CSVType {  
}  
}
```

@Path("/customer")

```
class CustomerResource {
```

@Post @Path("/new")

@Consumes(MediaType.WILDCARD) @Produces(TextPlain)

```
String addCustomer(Customer customer) {
```

```
    return customer.getSSN();
```

|  
@Get @Produces(Wildcard) @Path("/find/{ssn}")

```
public Customer findCustomer(@PathParam("ssn") String ssn) {
```

@PathParam("ssn") String ssn) {

```
Customer c = new Customer();
```

```
c.setSSN(ssn);
```

```
c.setName("John");
```

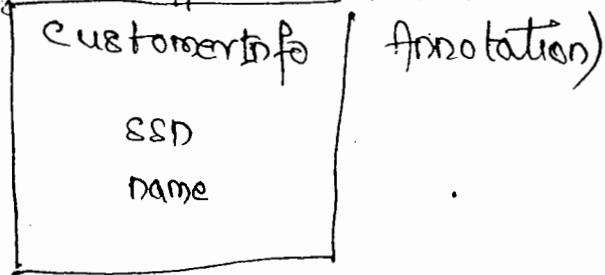
```
}
```

## CustomerResource

```
String addCustomer(CustomerInfo customer)
```

```
CustomerInfo findCustomer(int id)
```

@CSVType — (Custom



CSV file

```
[id=1, name=xyz]
```

src

```
com.ceph.resource
  -- domain
  -- handlers
  -- common
```

```
public class Customer {
```

```
    String id;
```

```
    String name;
```

```
// getters & setters
```

```
}
```

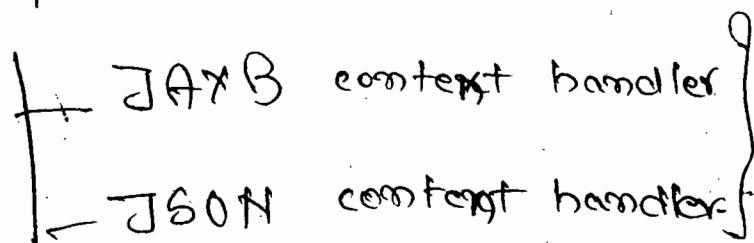
→ Inject ContextResolver in MessageBodyReader &

MessageBodyWriter classes

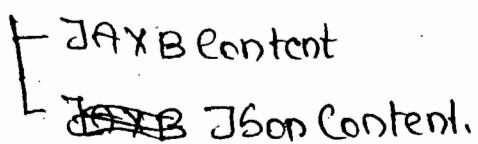
→ MessageBodyReader &  
a writer.

→ If you remove @Provider then if parameter type  
or resource type is JAXB / JSON then also  
JAX-RS runtime will register it b'coz of  
default context handlers.

### Default Context Handlers



Content Handler ~~Butt~~ in-built



CSV → will n't recognize.

Write own custom Content Handler.

# Context Resolvers

@Provider

```
public class JAXBContextResolverImpl implements ContextResolver<?> {
    private JAXBContext jContext;
    JAXBContext getContect(Class<?> classType) {
        if (classType.isAssignableFrom(PersonalInfo.class)
            || classType.isAssignableFrom(AccountInfo.class))
            return jContext;
        return null;
    }
}
```

```
JAXBContextResolver()
```

```
jContext = JAXBContext.newInstance(AccountInfo.class);
```

?  
?

// write to

→ @Context → Injecting provider class

↳ [ @Context  
private JAXBContext jContext = ctxResolvers.getContext(classType); ]

ContextResolver<JAXBContext>  
ctxResolvers;

```
catch (JMSException e)
```

```
    throw new WebApplicationException(e)
```

```
}
```

```
}
```

Code example

Insert/bank/newAccount

Raw

<personalInfo>

<ssn> 5551</ssn>

<name>john </name>

</personalInfo>

Problem

→ In Reader MessageBodyReader and MessageBodyWriter

We're creating heavy JAXBContext twice.

Bigest Problem

For every any no. of requests JAXBContext  
Should be created only one.

• ContextResolvers → Solution

↳

14-06-19  
Saturday

resteasy.xml = true

@Provider @Produces (Application-xml)  
public XMLMessageBodyWriter<?> impl MessageBodyWriter<Object> {  
ResourceClass obj  
    getSsize (Object objResourceType object ,  
            Class<?> classType, Type rawType,  
            Annotation[] annotation, MediaType mediaType){  
        return 0; // runtime will take care of it.

2

isWriteable ( ) {

If (classType.isAnnotationPresent(XMLRootElement.class))  
    return true;

    return false;

} writeTo (Object object, classType, rawType, annotation  
            mediaType, ~~HttpHeaders~~, OutputStream) {

try {

JAXBContext jContext = JAXBContext.newInstance(classType);

Marshaller marshall = jContext.createMarshaller();

marshall.marshal (object, outputStream);

```
④ @Concurrents (App - zml)
public class XmlMessageBodyReader implements MessageBodyReader {
    isReadable (ClassType, mimeType, annotation,
               mediaType) {
        if (classType.isAnnotationPresent(XmlRootElement.class)) {
            return true;
        }
        return false;
    }
    Object readFrom (ClassType, mimeType, anno, mediaType,
                     HttpHeaders, InputStream) {
        try {
            Object obj = null;
            JAXBContext jcontext = JAXBContext.newInstance
                (classType);
            Unmarshaller unmarshaller = jcontext.createUnmarshaller();
            obj = unmarshaller.unmarshal (InputStream);
        } catch (Exception e) {
            throw new WebApplicationException (e);
        }
        return obj;
    }
}
```

```
com.cch.resource
  " 4 . domain
    " 5 . handlers
      " 6 . util
```

@XmlType(@AccountInfo)

```
class PersonalInfo {
  @XmlElement
  private String accNo;
  @XmlElement
  @AccountInfo name;
  type;
}
```

//setter & getters

class AccountInfo

```
@XmlElement
 8&nbs;
  name;
```

```
xmlRootElement(name = "accountInfo")
@XmlElement(type = XmlAccessType.FIELD)
```

⑥

@XmlType(name = "accountInfo")

```
propOrder = {"accNo",
  "name",
  "type"})
```

@Path("/bank")

class BankResources

@Post

```
public AccountInfo openAccount(@POST ->
  @Produces("application/xml"))
  @Consumes("application/xml")
```

```
public AccountInfo openAccount(PersonalInfo pInfo) {
```

```
  accountInfo = new AccountInfo();
```

```
  accountInfo.setAccNo("Acc11");
  accountInfo.setName(" - - - ");
```

```
  return accountInfo;
```

⑥

→ Jax-rs runtime provides 2 ~~classes~~ <sup>Interface</sup> :-

- MessageBodyReader	}	Register it with
- MessageBodyWriter		HttpServletDispatcher and specify mediatype

→ Jax-rs runtime doesn't know how to convert into obj. So, write the above classes & register it with HttpServletDispatcher

→ When input comes jax-rs runtime checks

① Consumes the choice for respective reader.

MessageBodyReader

- readFrom
- isReadable

MessageBodyWriter

- isWrittable
- getSize
- writeTo

→ B-logic will be data format independent.

## Custom Content Handlers

→ In predefined content handlers input comes as raw type not object.

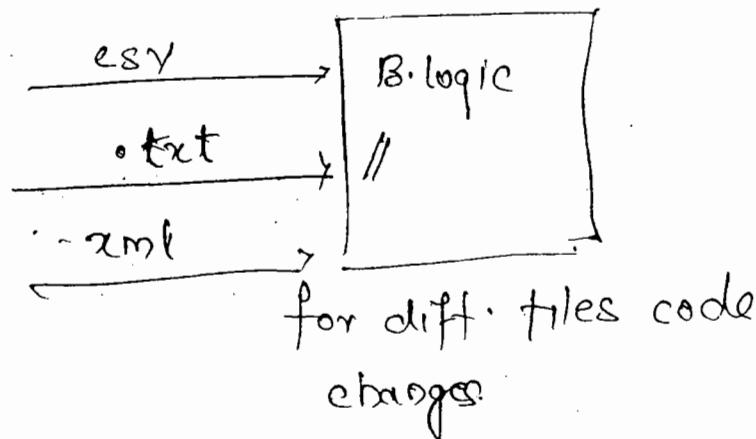
- → XML



resource class

B-logic (Objects should be passed)

Representation  
Orientation



→ Centralize conversion code.

```
<personalInfo>
  <ssn></ssn>
  <name></name>
</personalInfo>
```

/rest/bank

JAX-RS runtime

HttpServlet  
Dispatcher

/rest/\*

Application [xml]

MessageBodyReader

MessageBodyWriter

```
@Path("/bank")
public class BankResource {
  @Post
  @Consumes("application/xml")
  @Produces(" ")
  public AccountDetails
    openAccount(PersonalInfo)
}
```

```
@Path("/customers")
public class CustomerResource {
  @Post
  @Consumes("application/xml")
  @Produces(" ")
  public String addCustomer(
    CustomerInfo cInfo)
}
```

3-06-14  
Friday

## File

→ Uploading files e.g. form16.

④ Produces (text - plain)

④ Consumes (MULTIPART-FORM-DATA)

④ Post @ Path ("applyLoan")

```
public String applyLoan(File form16){
```

```
int c = -1;
```

```
try{ return "Your form16 has been";
```

```
fileReader reader = new FileReader(form16);
```

```
while((c = reader.read()) != -1){
```

```
System.out.print(c);
```

```
}
```

Jax-rs

```
} catch (...) { → file will be temporarily treated
```

```
?
```

file in server loading in

```
finally {
```

one shot.

```
reader.close();
```

```
}
```

WEB-INF

└ applyloan.jsp

```
<body>
```

```
<form action="rest/bank/applyloan" method="post">
```

```
<input type="file" name="form16"/>
```

```
<input type="submit" value="upload" />
```

```
</form>
```

```
</body>
```

doc = builder.parse(new String(payeeInfo));

NodeList accNoNodeList = doc.getElementsByTagName("item(0).getFeatureContent());  
return accNoNodeList;

String s = new String(payeeInfo);

String s = new String(payeeInfo);

String accNo = s.substring(s.indexOf("<accNo>") + 7,  
s.indexOf("</accNo>"));

@Path("deposit")  
@Consumes(MediaType.APPLICATION\_FORM\_URLENCODED)

@Post  
@Produces(MediaType.TEXT\_PLAIN)

public String deposit(MultivaluedMap<String, String>  
formData) {

return "Amt is deposited into acc"

return formData.getFirst("accNo");

accNo	act
amount	1248

as input.

E.g MultivaluedMap

```

try {
    while ((line = lineReader.readLine()) != null) {
        buffer.append(line);
    }
} catch (Exception e) {
}
return buffer.toString();
}

```

`@POST`  
`@Consumes ("application/xml")`  
`@Produces ("application/xml")`  
`@Path ("/{addPayee}")`

`public byte[] addPayee (byte[] payeeInfo) {`

`StringBuffer response, accNo;`

`accNo = getAccountNo (payeeInfo);`

`response.append ('<payeeack">')`

`append ("<accno>").append (accNo);`

`append ("<accno>").append ("Your message is being  
processed").append ("<message>").append ("</message>")`

`return response.getBytes();`

`Response xml`  
`<payeeack>`  
`<accno></accno>`  
`<message></message>`  
`</payeeack>`

`private String getAccountNo (byte[] payeeInfo) {`

`DocumentBuilderFactory factory = DocumentBuilderFactory.  
newInstance ();`

`DocumentBuilder builder =`

⑥ Produces (Application - XML)

⑦ Path ("balance/{accno}")

```
public StreamingOutput getAccountBalance(@PathParam("accno")
   final String accNo) {
```

```
    return new StreamingOutput() {
```

```
        public void write(OutputStream out) {
```

```
            PrintWriter pw = new PrintWriter(out);
```

```
            out.print("<bank>");
```

```
            " ", "<accno>" + accNo + "</accno>");
```

```
            " ", "<bal>" + 0.0 + "</bal>");
```

```
            " ", "</bank>");
```

```
        };
```

```
        out.close();
```

~~Fig: StreamingOutput as response~~

⑧ POST

⑨ Consumes (application/x-www-form-urlencoded)

⑩ Produces (text/plain)

⑪ Path ("transfer")

```
public String transferFunds(Reader reader) {
```

```
    LineNumberReader reader = new LineNumberReader(reader);
```

```
    StringBuffer buffer = new StringBuffer();
```

```
    String line = null;
```

```

    while ((c = bis.read()) != -1) {
        buffer.append((char)c);
    }
    catch (Exception e) {
    }
    return buffer.toString();
}

```

Gets input as part  
 of form body

// Write Application class

```

    L getSingletons()

```

### Advanced Rest Easy

Form

From Ac	2001
To Ac	1001

/rest/bank

public ~~void~~ get

→ HTTP req body has content.

→ If data flows as req body how to access & process it in our code.

JAX-RS runtime will automatically gives input as InputStream

public String processOrder (InputStream in)

Predefined received data type

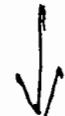
So, no anno reqd,

→ Query, Path, Matrix, Form, Cookie, - gets data from diff. parts of header

→ E.g. : InputStream as param input

@Path("bank")

public class BankResource {



@POST

@Consumes (APPLICATION\_FORM\_URL\_ENCODED)

@Produces (TEXT\_PLAIN)

public String transferFunds (InputStream in) {

    BufferedInputStream bis = new BufferedInputStream (l).

    StringBuffer buffer = new StringBuffer();

    int c = -1;

```
S.o.p ("classType : " + classType);
```

```
for (Annotation anno : annotations){
```

```
    S.o.p (@anno.toString());
```

1

12-06-14  
Thursday

## JAX-RS Content Handlers

allowable  
10 types

- StreamOutput ✓ (O)
- InputStream/Reader ✓ (I)
- File (O)
- byte[] ✓ (I)(O)
- MultivaluedMap<String, String> for form input ↗(I)
- JAX-B
- ~~create~~ Custom ContextResolvers
- Custom Marshalling / Unmarshalling
- (MessageBodyWriters and MessageBodyRenderers)

→ If any class is accessible over web following  
restful architectural principle is Resource class

@Provider

public class CustomParametersConverterProvider implements

ParamConverterProvider {

public < T > ParamConverter < T > getConverter(  
Class < T > classType {  
}) {

If (classType • IsAssignableFrom ( criteria . class ))

ParamConverter < T >  
return new CriteriaParamConverter();

}  
return null;

@Provider — Become deployable resource

used directly by jax-rs runtime.

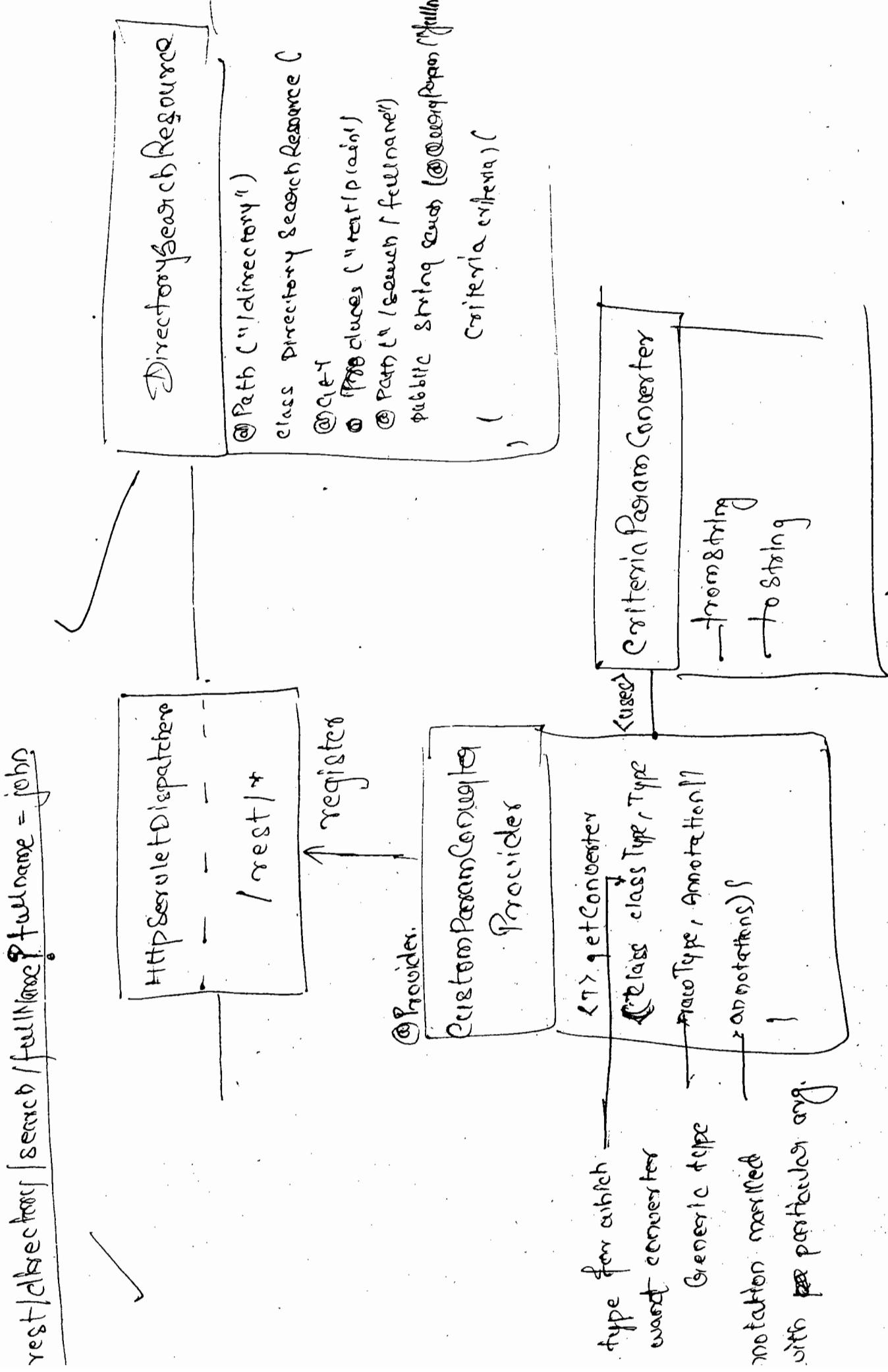
→ Override getClasses() else Provider class

won't be picked up b'coz auto scan will

not work while using Application. It works

for HttpServletDispatcher.

rest | directory | search / fullName ? full name = john



com.cpe.domain;

class Criteria {  
 String fullName;  
 // setter & getter  
}

} This class doesn't classifier for automatic type conversion b'coz no cons or valueOf

java.util...ext. → 2.0 api

public class CriteriaParamConverterImpl implements ParamConverter {

public Criteria fromString (String val) {  
 Criteria criteria = new Criteria();  
 criteria.setFullName (val);  
 return criteria;

public String toString (Criteria criteria) {  
 return criteria.getFullName();

public class ~~CriteriaPath~~ CustomParamConverterProviderImpl  
 implements ParamConverterProvider

public <T> Converter<T> getConverter()

public <T> Converter<T> getConverter (Class<T> classType,  
 Type class)

~~Annotations~~ ~~Annotations~~ ~~Annotations~~

@Path("directory") ("directory") jax-rs 2.0  
public class ~~AddressResource~~ TelephoneDirectoryResource

@Path("/search/{fullName}")

@Produces("text/plain")

@GET

public String search(@QueryParam("fullName") Criteria criteria){

return "tele no : 1234567890 for person:"

criteria.fullName());

[com.cpc.domain] <sup>utd</sup>

@ApplicationPath("/rest")

class DirectorySearchApplication extends Application

// override getSingletons()

// override getClasses() {  
    <sup>etc</sup>

{  
    com.cpc.resource  
    " " . application utd  
    " " . domain  
    " " . converter  
    " " . provider

Jax-rs provides 2 interfaces:

ParamConverter

ParamConverterProvider - getParamConverter()

→ Write our own Parameter Converter

→ Register it with Jax-rs.

toString(Object) - String  
fromString(String) - Object

```
com.pc.util;
class ParamConversionApplication extends Application {
```

```
    private Set<Object> singletons;
```

```
    public ParamConversionApplication() {
```

```
        singletons = new HashSet<Object>();
```

```
        {  
            . . .  
        }
```

```
/rest/meatBunTicket/
```

Jax-RS automatically  
converts Object types  
& collection types  
if you follow rule

```
class MovieInfo {
```

```
    private String id;
```

```
    @QueryParam("eno")
```

```
String cancelTickets (List<Integer> seatNos) {
```

```
    StringBuffer buffer = new StringBuffer();
```

```
    Boolean isFirst = true;
```

```
    for (Integer sno : seatNos) {
```

```
        if (isFirst)
```

```
            buffer.append(sno).append(",");
```

```
    } else
```

```
        buffer.append(seatNo).append(",");
```

```
    return buffer.toString();
```

/rest/meatBunTicket/cancel?eno=1

Add multiple query params & test

③ Collection Types are allowed:-

↳ follows rule ① or ②

user submits multiple param values

```
public String process(@QueryParam("id") List<orders>){  
    * * *  
    }  
    List<Primitives> } both should follow ① or  
    List<Objects> } ②
```

→ Servlets are also restful resource working closely with Http but prog. has to write boiler plate code. So, Sun gave a wrapper on Servlets by giving a more concise api jax-rs, ~~but~~ taking care of boiler plate code.

```
src  
└ com.pc.resource  
    └ com.pc.domain  
        @Path("/nextGenTicket")  
        public class NextGenTicketingResource{
```

```
    @GET  
    @Produces(MediaType.TEXT_PLAIN)  
    @Path("{id}")
```

```
    public String bookTicket(@PathParam("id") MovieInfo movieInfo){  
        return "booked :" + id;
```

```
└ com.pc.domain  
    public class MovieInfo{  
        int id;  
        //getters & setters  
        public MovieInfo(int id){  
            this.id = id;
```

11-06-19  
Wednesday

Jax-rs 2.0

## Working With Custom Parameter Converter

Automatic Type Conversion works

```
@Path("/resource")
class MyResource {
    @GET
    @Produces("Text-Plain")
    @Path("{id}")
    public String process(@PathParam("id") String id) {
        Order order = ...
        ...
    }
}
```

① Primitive data types of java (int, char, boolean, byte, string, short, long, double) are allowed as parameters.

② User defined data types are allowed (class), but certain conditions exists:-

- |- One constructor taking string as parameter (arg)
  - |- static valueOf() method which should take string as argument

@BeanParam — Jay - vs 2.0 only  
feature

6 ways to get values (GET)

{  
    @QueryParam  
    @RequestParam  
    @CookieParam  
    @FormParam  
    @TemplateParam  
    @HeaderParam

} we want to use  
values from these  
ways.

class StudentResource {

If you pass multiple  
parameters it looks ugly.

public String search(@

int id,  
String name,  
String collegeId,  
String lastName)

// better & getters

@Post  
@Path("{collegeId}/search")  
@Produces("text/plain")  
class SearchCriteria

public String search(@

BeanParam

SearchCriteria criteria)

}

we need to create  
this Bean & pass it.

Done by  
HttpDispatcher  
Servlet.

@FormParam("name")  
private String name;

@PathParam("collegeId")  
private String collegeId;

↳ @QueryParam("last") @DefaultValue("p")

@Path("/student")

class StudentResource {

@POST

@Produces("text/plain") @Consumes("application-form-urlencoded")

@Path("/{name}) → matches with form actions

public void register(@FormParam("id") int id,

                  @FormParam("name") String name)

return; -- ;

↳ Instead of form data going to

Servlet comes to Resource class (restful res)

Get → Appends to ~~every param~~ request header

Post → Appends to request body.

Application feature url/FormEncoded - content type  
when sent from Form

rest/new

o Post

Forms	Fakes	Payload
-------	-------	---------

# Add new Head

id	10
----	----

name	John.
------	-------

Application/x-www-form-urlencoded

[sent]

```
for( String paramName : request.getHeaderNames() ) {
```

```
    buffer.append("name").append(paramName).  
    append("value"), append(request.getHeader(paramName))
```

→ If annotation if you know the parameter name then only you can get value. Else you have to use API.

Get all cookies

```
getCookies()
```

@FormParam

→ ~~param~~

insert.jsp

```
<form action="rest/student/neo" method="post">
```

```
    <input type="text" name="id"/>
```

Name

```
    <input type="submit" value="Register"/>
```

```
</form>
```

To Get All headers as part of response

→ HTTP headers will not be in Uri it's in httpHeaders.

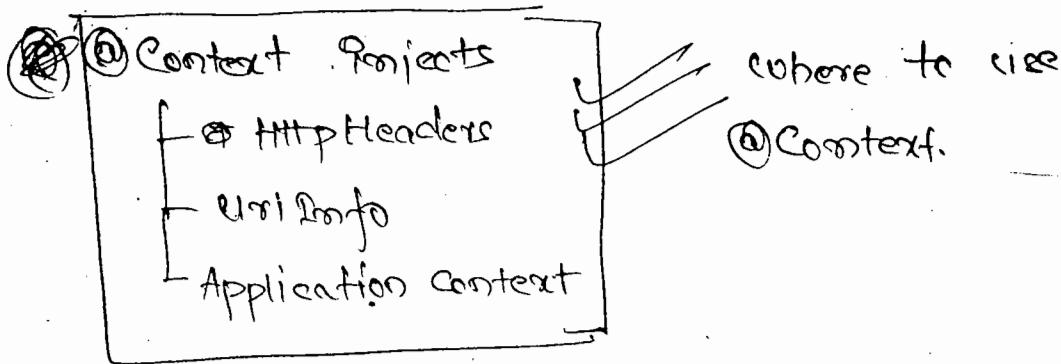
④ GET

⑤ Produces ("text/plain")

API class

public String showHeaders(~~HttpHeaders~~ httpHeaders)

⑥ Context HttpHeaders httpHeaders)



④ GET

⑤ Path (" / allHeaders")

⑥ Produces ("text/plain")

String getAllHeaders ⑦ Content HttpHeaders ~~HttpHeaders~~

MultivaluedMap<String, String> repHeaders =

headers.getReqHeaders();

```
@Class StudentResource {
    @Get @Produces("text/plain")
    @Path("/{detailId}")
    public String getStudentDetail(@PathParam("detailId") int studentId) {
        return " ";
    }
}
```

mandatory-path param

optional-query param

```
student/details
@Get
@Path("/newCookie")
public Response newCookie()
```

cookieMap  
Google Apps

```
Response response = null;
return response = Response::ok().cookie(newCookie().noContent()
    .cookie(NewCookie("id", ...).build()))
```

→ To get Browser Name

⑥ ~~HttpHeaderParam~~

```
@Get
@Path("/userAgent")
@Produces("text/plain")
public String getUserAgent(@HeaderParam("User-Agent") String userAgent)
```

return " " + userAgent;

}

`@Path("id")`

new with every req creates  
new object of class



public String getGrade(@Context UriInfo uri) {

    StringBuilder buffer = new StringBuilder;

    MultivaluedMap<String, String> queryParams =

        uri.getQueryParameters();

    for (String paramName : queryParams.keySet()) {

        StringBuffer

        append(paramName).append("=").append(queryParams.getFirst(paramName));

    buffer.

        .append(queryParams.getFirst(paramName));

    pathParams = uri.getPathParameters();

    for (String paramName : pathParams.keySet()) {

        StringBuffer

        .append(paramName).append("=").append(pathParams.getFirst(paramName));

## Cookie Value

How to access value

10-06-14  
Tuesday

http://app-box.com/restexamples

④ Path (" /student")

public class StudentResource {

⑥ GET

④ Produces ("text/plain")

public String getGrade (@DefaultValue("10")

④ @QueryParam ("id") int id)

/rest/student

→ No query parameters specified so  
Default value will be applied.

→ URIInfo class can be used to access entire

URI.

/rest/student?id=10 & id=11

multiple query param

④ How to access all details of request

or URI? [Multiple query params]

Ⓐ Use URIInfo class.

// Resolving ~~scope scope~~ using Path Segment.

@GET  
④ Produces ("Text-Plain")  
④ Path ("price/{make}/{model}")  
float getPrice (String make, String model)  
String  
String make, @Path -- ("model")  
String model  
④ MatrixParameters ("color") String color)  
return ; - - - - ;  
}

// uri: rest/cars/price/maruti; color = red/  
swift; color = white

~~cars/price/price/mar~~

String make

Path Segment makePS

makePS.getMatrixParameters().getFirst("color") +  
" model color;" + modelPS.getMatrixParameters().  
getFirst("color")

```
MultivaluedMap<String, String> matrixParams =  
    null;
```

```
StringBuffer buffer = new StringBuffer();
```

```
buffer.append("<car>");
```

```
matrixParams = makePS.getMatrixParameters();
```

```
for (String key: matrixParams.keySet()) {
```

```
    buffer.append("<" + key + ">");
```

```
    append(matrixParams.getFirst(key));
```

```
    append("<" + key + ">");
```

```
}
```

```
buffer.append("</car>");
```

```
} return new CarXMLStreamingOutput(buffer.toString());
```

```
private final class CarXMLStreamingOutput implements
```

```
StreamingOutput {
```

```
private String xml; // cons.
```

```
write(OutputStream out) {
```

```
PrintWriter pw = PrintWriter(out);
```

```
pw.println(xml); pw.close();
```

```
}
```

```
/rest/car/month; year=2010;
```

```
color-red·fuelType=diesel;
```

resolving it use PathSegment only, by  
prog. retrieving the values.

### PathSegment Uses:

- Dynamically retrieve all matrix params attached to that path segment
- Scoping problem resolution (in case multiple matrix params with same name exist)

## II Working with PathSegment / URI Fragment

@Path ("/{car}")

Getting matrix params  
from ~~Path~~ URI Fragment

class CarResource {     '/' auto append it

@Get

@Produces

③ Path ("/{make}-{model}")

// Getting Matrix Param Dynamically  
public StreamingOutput getCar (

③ PathParam ("make") PathSegment

/rest/cars/maruthi; year=2010; fuelType=diesel /

swift; color=red.



We need to get all matrix param of particular ~~uri~~. PathSegment.

② PathParam ("make") PathSegment makePS, - ?.

; makePS.getPath();

makePS.getMatrixParameters();

g

g

If you don't know name of parameters. It's multi-valued map.

Key	List of values
-----	----------------

/rest/cars/maruthi; year=2010; color=red /

swift; color=grey



first

You will always get ~~last~~ matrix param value if you have some matrix param name in ur uri. It's limitation for

PathSegment → A position in the path.  
(path fragment)

```
    ↗ PathSegment getPath();
```

→ You can declare any no. of PathSegment.

## Matrix Parameters

④ Path ("car")

```
class CarResource {
```

④ GET

④ Produces (app - XML)

④ Path ("/{make}/{model}")

```
public StreamingOutput getCar (
```

④ PathParams("make") String make, "model")

④ MatrixParam("year") int year, "color".

```
/rest/car/{make}/{model}
```

```
/rest/car/maruthi;year=2010/swift;color=red
```

→ You can have any no. of matrix param  
for PathSegment.

## web.xml

<8>

<8-> resteasy <18->

<8-c> ... <18-c> ...

<18>

<i-p>

< /resteasy > com

<1p> ... mapping ...

</rest>

GET req



http://localhost:8080/RSInjectionWeb/rest/pizza/10/order/  
1-2

## Working with PathSegment

@Path("/pizza")

class PizzaResource {

/rest/pizza/10

@GET

@Produces

@Path("{id}")

public float finalPrice (@PathParam("id")

PathSegment ~~id~~ id PathSegment) {



java -rs provided class for

param extracting value of id.

- If you're using ~~at attribute~~ <sup>level</sup> then create a new obj. else every new req. will override previous req. values.
- If you're creating new obj / req then you can use anno. at attribute level or constructor level. [Concurrency / Thread safe issues]
- If resource i.e singleton then recommended to use annotations at method level.

```
@Path("/pizza/{id}")  
class PizzaResource {  
    @Path("/{order/{id}-{quantity}}")  
    @GET  
    @Produces("Text-Plain")  
    public String placeOrder(int storeId,  
                            int pizzaId, int qty)  
        return --storeId --;
```

{

{

## Scope - Path Parameters

@Path("/pizza/{id}")

class PizzaStore {

@Path("/order/{id}/{quantity}")

@Produces("Text-Plain")

@GET

public String placeOrder(@PathParam("id") int storeId,  
@PathParam("id") int id, @PathParam("quantity")  
int quantity) {

}

http://localhost:8081/crudweb/rest/pizza/101/order/111

→ Method level param always takes precedence

over class level param value.

→ If we use ~~same~~ template parameter

with same name at class level & method

level then method level of path param

overrides it. In above both storeId &

Id will be 1.

09-06-14  
Monday

# Injections

→ In Get req we can <sup>extract</sup> get data to  
two ways:-

— Annotations (mostly used)

— Working with API (Low level)

a) @PathParam

~~class OrderResource~~

@Path("order")

```
class OrderResource {  
    @Path("ffname") → {lastname}  
    @Produces("application/xml")
```

@GET

```
public StreamingOutput finalOrder (@PathParam("fname")  
                                    @PathParam("lastname")  
                                    String fname, String lastName)
```

http://localhost:8080/crudweb/order/a-b

→ A sub resource locator returns obj of resource  
so, it anno with @Path to Server at runtime  
dynamically. It's fully dynamic

Dynamic

@Path("/{car}")



@Path("/{service}")

~~@Path~~

public SubResourceClass - .

Fully Dynamic

@Path("/Vehicle")



@Path("/{path}")

~~not~~ public Object - .

public class SalesResource {

@GET

@Produces (TextPlain)

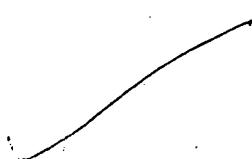
public String getPrice (@PathParam("model") String model) {

return ~~to model + "price"~~;

"22222.22";

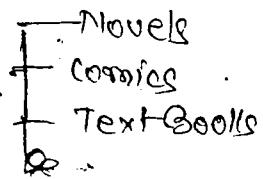
}

}



# Sub Resource Locators

(Resolving req to resource dynamically) Books



@Path("vehicle/{type}")

class VehicleResource {

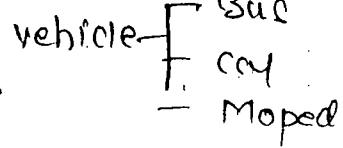
@Path("{type}")

public &<sup>Object</sup> jobsheet(String vehicleNo,

@PathParam("type") String type){

if (type.equals("car")){

return new CarResource();



}

}

public class CarResource {

@POST

@Produces("text/plain")

public String repair(String vehicleNo){

(@QueryParam("vno") String vehicleNo);

t

↑

Based on req. method it  
is called.

> jobsheet is sub resource locator if we'll

only delegate. It will only have @Path

## QueryParam

## Matrix

- Append at → Append at last. in url.
- Scope - var
- Every  $\text{Impl}$  supports

- Can appear at any place at template url
- Scope: individual Path Segment
- Model  $\text{Impl}$  supports.

$@Path("/name")$

findPersonnelInfo(@PathParam("name") String name,

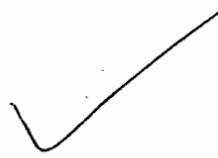
$@MatrixParam("height") \text{Int} height)$  {

return name + " " + height;

~~named~~ ~~name~~

[customer | john | name; height = 6]

→ Matrix parameters aren't supported by all  $\text{impl}$ . It's supported by Resteasy. It's in jax-rs api.



@Path("/{year}/{make}/{model}")

class CarResource {

~~Scn~~

search(@PathParam("make") String make,  
@PathParam("model") String model)

{}

}

Matrix param are at Template param scope



CRWEB/rest/cars/maruti; color=red/swift;  
year=2014

→ Matrix params are optional.

→ Anywhere in uri.

→ Won't participate resolving req to resource.

→ Add'l attr details of template param.

## Matrix Parameters

→ Not initially supported by HTTP. Initially supported only Query & Path param.

`http://localhost:8080/crudweb/customer/john; height=20/personal`

④ Path("/customers/{name}")

## class CustomerResource {

## • @GET

⑥ @GET  
⑥ Produces ("application/xml") @Path ("personal")

④ Produces ('application/json') Content-type:  
public streaming output getPersonalInfo (

@PathParam ("name") String name) {

1

© GET

④ Path (" / business )

```
public StreamingOutput getBusinessInfo()
```

→ You want to give appear give some extra info

`http://localhost:8080/cR web/customer/john;`

height = 6; color = red / personal

- You can write ~~Path~~ Template parameter at class level & at method level.
- If Template parameter is common for all methods It should be at class level. If you to address one specific method keep it at method level.

④ Path (" /firstname/{id}") } method level.  
⑤ Path (" /{id} /name") }

## Regular Expressions

- Id should be Integer only. Restrictions.

param name digits  
↓            ↓      Any no. of digits  
⑥ Path (" /customers /{id: \d+}")

req uri - 13 ✓ — output  
- - 13a — 404 Not Found

④ @GET

method1

→ /go

④ @GET

method2

→ Ambiguity

④ How to resolve request to HTTP request

method if same http method designation  
is used? [Here 1 way is mentioned]

① Anno method with @Path method()  
→ Sub resource.

@Path

class

→ Root Resource

→ If not then always the req. goes to  
first method only

@Path ("name")

@GET

m1()

↑

Subresource

@Path ("lname")

@GET

m2()

↑

src

```
+ com.bur.resource  
|   + CustomerResource  
+ com.bur.util  
|   + BurApplication
```

```
@Path("{id}/customer")  
public class CustomerResources {  
    @GET  
    @Produces(MediaType.TEXT_PLAIN)  
    public String findName(@PathParam("id") int id) {  
        return id + "-Name";  
    }  
}
```

```
@ApplicationPath("/rest")  
public class BurApplication extends Application {  
    private Set<Object> singletons;  
    public Set<Object> getSingletons() {  
        singletons = new HashSet<Object>();  
        singletons.add(new CustomerResource());  
    }  
    public Set<Object> getSingletons() {  
        return singletons();  
    }  
}
```

# How to work with Path Parameter

3 ways

- | - **Template Path Parameter**
- | - **Regular expressions**
- | - **Matrix parameters**

## Template Parameter

@Path("customer/{id}")

class CustResource {

@GET

@Produces("application/xml")

public &streamingOutput getCust (

@PathParam("id") int id) {

}

}

# Working With URI

```
http://localhost:8080/clever/treat/customer/10  
① Path("customer")  
class CustomerResource {  
    @GET  
    @Produces("application/json")  
}
```

② Path("customers/{id}")

or

③ Path("{id}/customer")

- ↑  
Path param can be anywhere in url.

Path parameter

↓  
customer ? id=10      Query param

How to send value in GET request

① → Path parameter has better readability than query param.

② → Path parameter gets space reserved in the uri. Positional parameters are reserved in uri. It's mandatory. But, query param value are optional.

③ → Path parameter may appear at any place in uri. Query params are at last of uri.

④ → Query param will not contribute in ~~for~~ mapping

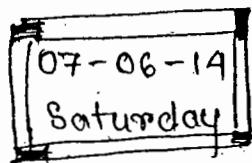
req to resource. But path param participates

1a / <sup>\* Path param</sup> 1 — req

1a            1a / x

→ ⚡ Path should be there on impl classes not on Abstract class. ⚡

→ Write anno on method level not class level.



for some B-class you can have multiple  
B-logic

### Template Method Design Pattern

abstract public class IncomeTax {

    @POST

    @Consumes("application/xml")

    public void fileITReturns(InputStream in) {

}

    public void processForm16(InputStream in);

}

    @Path("/linfoit")

    public class InfoTaxImpl  
        extends IncomeTax {

        processForm16(InputStream in)

{

}

    @Path("/tcs&lt")

    public class TaxTaxImpl extends  
        IncomeTax {

        processForm16(InputStream in)

{

}

```
public abstract class OrderResource {  
    public Response createOrder(InputStream in)  
    {  
        abstract Order buildOrder(InputStream in);  
    }  
}
```

```
@Path("/order/international")  
public class OrderResourceImpl
```

```
private Order buildOrder(InputStream in)  
{  
    // JAX-B or DOM or SAX or STAX  
    if (JAXB)  
    {  
        // & SAX  
        // & STAX  
    }  
}
```

```
@Path("/order/domestic")  
public class OrderResourceImpl
```

```
private Order buildOrder(InputStream in)  
{  
    // JAX-B  
}
```

## Working In Abstract

web.xml

[resteasy.scan]

true

[resteasy.servlet-mapping.prefix  
/rest]

package com.wiva.resource;

@Path("/address")

public interface AddressResource {

@GET

@Produces(MediaType.TEXT\_PLAIN)

String getArea(@QueryParam("zip") int zipCode);

}

public class AddressResourceImpl implements AddressResource {

public String getArea(int zipCode) {

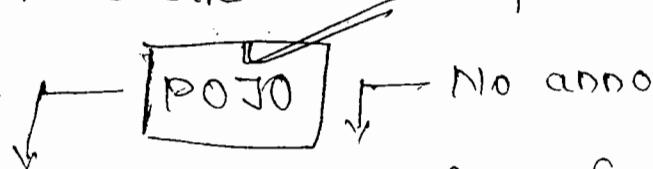
return "zip:" + zipCode + "Area :" + "Hyd";

# How to use Interfaces and Abstract classes to expose Resource classes

- If you annotate any class with jax-rs  
~~import~~ anno then logic is polluted with  
jax-rs anno. If it doesn't act as POJO.

```
@Path("/customer")
interface CustomerResource {
    @POST @Consumes("application/xml") @Produces("text/plain")
    public Response createCustomer(InputStream in);
}

class CustomerResourceImpl implements CustomerResource
```



```
public Response createCustomer(InputStream in)}
```

You can't have multiple anno b'coz  
anno are placed at interface.

```
private final class OrderStreamingOutput $impl  
    StreamlongOutput {  
    private String xml; // cons.  
    public void write(OrderStream out) {  
        PrintWriter pw = out.getWriter();  
        pw.println(xml);  
        pw.close(); // pw.flush();  
    }  
}
```

### ⑥ Application Path (" /rest")

```
public class HttpMethodApplication extends Application {
```

```
    public Set<Object> getSingletons() {  
        private Set<Object> singletons;  
        public HttpMethodApplication() {  
            singletons = new HashSet<Object>();  
            singletons.add(new OrderResource());  
        }  
        public Set<Object> getSingletons() {  
            return singletons;  
        }  
    }  
}
```

→ Advanced Resteasy →

► [http://localhost:8081/HttpmethodWeb/rest/order]

Payload.

```
<order>  
  <type>Domestic</type>  
  <price>200</price>  
</order>
```

Application XML

② Produces (MediaType.APPLICATION\_XML)  
public Stream<Output> getOrder (int id) {  
 if (orderMap.containsKey(id)) {  
 Order order = orderMap.get(id);  
 String orderXml = generateOrder(order);  
 return new OrderStreamingOutput(orderXml);  
 } else return null;  
}  
  
private String generateOrder (Order order) {  
 StringBuffer buffer = new StringBuffer();  
 if (order != null) {  
 buffer.append ("");  
 buffer.append ("<id>").append (order.getId());  
 buffer.append ("").append (order.getType());  
 buffer.append ("<price>");  
 buffer.append ("</order>");  
 }  
 return buffer.toString();  
}

```
{ else if (----- . equals ("price")) {
```

```
    }
```

```
    }
```

```
    }
```

```
    public String updateOrder
```

```
    @PUT @Consumes (MediaType.APPLICATION_XML)
```

```
    @Produces (MediaType.TEXT_PLAIN)
```

```
    Response
```

```
    public String updateOrder (InputStream in) {
```

```
        try {
```

```
            Order = buildOrder (in);
```

```
            if (orderMap.containskey (order.getId ()) {
```

```
                orderMap.put (order.getId (), order);
```

```
            } else {
```

```
                return Response.notModified () . build ();
```

```
                    L>To return diff response
```

```
status code.
```

```
}
```

```
} catch (Exception e) {
```

```
    return Response.serverError () . build ();
```

factory for  
creating Response

```
    return Response.ok () . build ();
```

```
        ok ("order . getId () 'updated ' );
```

```
}
```

06-06-14  
Friday

```
<order>
<id> 2 </id>
<type> Domestic </type>
<price> 23 </price>
</order>
```

```
private Order buildOrder (InputStream in) {
    DocumentBuilderFactory factory = null;
    DocumentBuilder builder = null;
    Document doc = null;
    Node root = null;
    factory = DocumentBuilderFactory.newInstance ();
    builder = factory.newDocumentBuilder ();
    doc = builder.parse (in);
    if (doc != null) {
        Order order = new Order ();
        root = doc.getFirstChild ();
        NodeList children = root.getChildNodes ();
        for (int i = 0; i < children.getLength (); i++) {
            Node child = children.item (i);
            if (child.getNodeName ().equals ("id")) {
                order.setId (Integer.parseInt (child.getTextContent ()));
            } else if (child.getNodeName ().equals ("type")) {
                order.setType (" - - ");
            }
        }
    }
}
```

@Path("/order")

```
public class OrderResource {
```

```
    private Map<Integer, Order>  
        orderMap;
```

```
    private AtomicInteger counter;
```

```
    public OrderResource()
```

```
        orderMap = new ConcurrentHashMap<
```

```
< Integer, Order >();
```

```
        counter = new AtomicInteger(0);
```

```
}
```

@POST

@Consumes(MediaType.APPLICATION\_XML)

public String newOrder(@RequestBody Order order)

@Produces(MediaType.TEXT\_PLAIN)

public String newOrder(InputStream in) {

Response response; → modify code

public String createOrder(InputStream in) {

```
        Order order = buildOrder(in); // try catch
```

```
        order.setId(counter.incrementAndGet());
```

```
        orderMap.put(order.getId(), order);
```

```
        return order.getId() + " " + order.getId();
```

} // catch

// ResponseServerError build()

→ http://localhost:8080/reqweb/rest/order?id=1

To Avoid

IllegalThreadStateException

use concurrent

HashMap

→ No two threads  
can access it at  
a time. Better  
than sync. block

→ Google Chrome → settings → Extensions  
↳ Apps tab → Web Store → search

Simple Resteasy Client or  
Advanced Resteasy Client and  
Cookies

Basic Example (http methods) | E.g.

Order — domain (DTO)

Order Resource

→ Don't use Serializable obj it will not  
be interoperable.

Http Methods Web

src

com.hbm.domain

Order

id  
type  
price

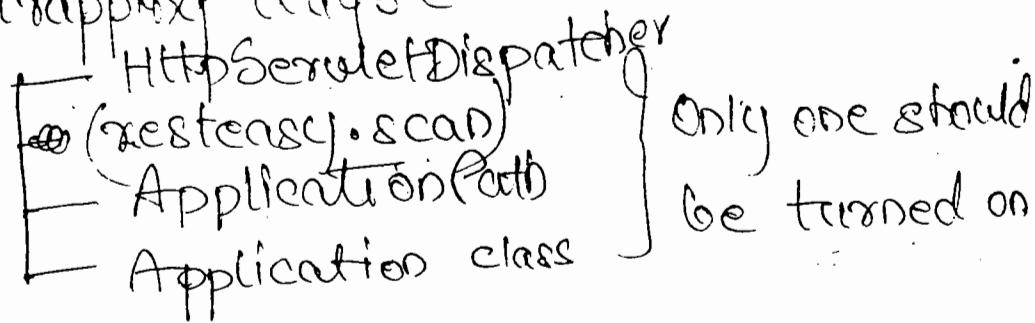
// getters  
getters

com.hbm.resource

orderResource

com.hbm.util

## Bootstrapping Ways (Few here)



→ By default scan = "false"

→ If both are there then one obj / req

strategy of resteasy.scan will take precedence.

public

private Set<Object> singletons; } use only one

private Set<Class<?>> classes; }

→ From JEE 3.0 web.xml is optional.

@ApplicationPath("/rest") ← No web.xml

public class UAApplication extends Application

}

```
@Path("/enquiry")
public class RailEnquiryResource {
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String getPnrStatus(String pnr) {
        (@QueryParam("pnr") String pnr) {
            // returns "conf";
        }
        // const
        // op
    }
}
```

### web.xml

```
<8-n> resteasy </8-n>
<s-c> HttpservletDispatcher </s-c>
<8-o>
```

// write Application class.

→ In ~~Singletons~~ getSingletons() you can return a set of objects, only one resource object will be used for processing any no. of req coming to that resource.

Since there's no `[resteasy-seal]` the `(init-param)` of `HttpServletDispatcher` will be searched to get the instantiation process.

using App Web

└ src

  └ com.eia.resource      └ RailEnquiryResource

    └ com.eia.util

      └ UAAApplication

→ ~~HttpServletDispatcher~~

## web.xml

< servlet >

< s-n > testeasy < /s-n >

< s-c > ... < /s-c >

< init-param >

< p-n > . . . Application < p-n >

< p-v > . . . FirstTRSApplication < p-v >

< /i-p >

< l-o-s > & < /l-o-s >

< /servlet >

④ → You can write a class extending

From Application class. It has two

concrete methods getClasses() &

getSingletons. returning Set.

→ You can return Set of classes in

getClasses, where those resource classes

will be used as per request one object.

05-06-14  
Thursday

Instantiation strategy

Info. abt resources

Application class

class GreetApp

class FirstRSApplication extends Application

per req. 1 obj

Set<Class> getClasses() { . . . }

concrete  
methods not  
abstract.

Set<Object> getSingletons() { . . . }

all req 1 obj

// for singleton

private Set<Object> singletons;

public FirstRSApplication()

singletons = new HashSet<Object>();

singletons.add(new GreetResource());

Set<Object> getSingletons()

{ return singletons;

}

Choosing Instantiation

Strategy Using

Application class

## Customizing url pattern

### HttpServlet Dispatcher

1+

→ Here, all req. will be mapped to its  
So, In app. it will receive ~~all~~ all the  
req. then prob comes in web apps.

HttpServletDispatcher  
[servlet]  
init parameter

- |- name = resteasy.mapping.prefix
- |- value = /rest
- [servlet-mapping]
- |- url-pattern = /rest/\*

Specific to Resteasy  
impl

servlet. ↴

http://localhost:8081/firstResteasyweb/rest/greet?  
name=john.

resteasy.scan = true

↓

### HttpServletDispatcher

ctrl	classname
ctrl 2	classname2
...	...

Map  
Data Structure

⑧ Response returned by the resource method will be dispatched back to the client by the HttpServletDispatcher by setting the appropriate content-type.

→ whenever a req. comes to Resource class it can act in two ways:-

### Instantiation Strategies

↳ Singleton. (single instance)

↳ non-singleton. (per-request new instance)

→ Singleton acts as stateful. (1 Resource class obj/all request)

→ non-singleton acts as stateless.

### HttpServletDispatcher

↳ Per request new instance (by default)

```
public CorrectResource(){}
```

```
    System.out.println("Hi");
```

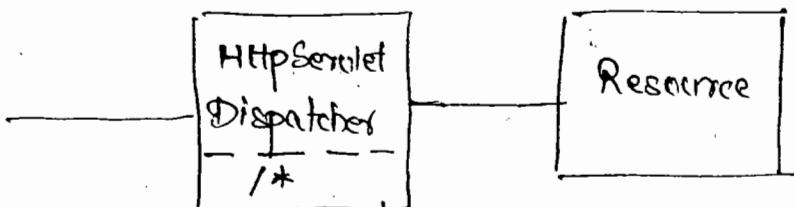
ctrl + F5 → Hard refresh.

→ Checks query parameters and gets Input then  
→ check MIME & sets contentType accordingly

[HttpServletDispatcher] /\*  
↓ ↓ ↓ ↓

- ① Always request comes to HttpServletDispatcher b'coz path is /\*.
- ② It will get extra path info of the request, with that it will try to identify a Resource class. ~~where~~
- ③ Path annotation contains extra path.
- ④ Instantiates obj of corresponding class.
- ⑤ It identifies the request method (i.e. request.getRequestMethod), for this method it will find a method on resource class with is annotated with <sup>respective</sup> ~~response~~ annotation ~~produces~~. (GET / POST ..)
- ⑥ Reads that metadata, evaluates if everything my method requires is available as part of the request or not.
- ⑦ Accept header with method MIME type (produces), If matching, extracts the info from Input request (req param, cookie param), passes as Input to the method

## Bootstrapper / engine of application.



Http supports only  
synchronous req/reply

So, we need to close  
the socket & open new

Http Processor

Extracts extra  
path info / greet  
& identifies if it's  
annotated with

Don't use /\*  
it's suspicious  
url pattern

@Path & tries to create obj  
of resource class (so, class  
must have default cons).  
Since, Servlet uses extra path  
to locate resource so no  
web1 / web2 reqd. ~~reqd~~

→ After creating the object it searches for resources  
in the class, & servlet says req calls service(-, -) if  
in it request.getRequestMethod(). Then after identifying  
method metadata then looks @Producers then  
checks if meta for accept header (what type of

## Uninstalling Options

<Servlet>

```

< servlet-name > resteasy < /servlet-name >
< servlet-class > org.jboss.resteasy ... HttpServletDispatcher < /servlet-class >
< load-on-startup > 1 < /load-on-startup >

```

<Servlet-mapping>

```

< servlet-name > resteasy < /servlet-name >
< url-pattern > /* < /url-pattern >

```

✓< servlet-class > → Scan resource classes.  
 < init-param >  
 < param-name > resteasy.scan < /param-name >  
 < param-value > true < /param-value >  
 < limit-param >  
 < load-on-startup > - - -

Right click on Server → Add & Remove projects

→ Add → ~~Finish~~ → 8080 is already bind

→ ~~①~~ Change port.

Sandbox → jboss → standalone → configuration

→ standalone.xml → search 8080 →  
change to 8081 → save.

→ Close server & double click. (change)

http://localhost:8081/firstResteasyWeb?

@Path("/greet")

public class GreetResource {

Entity providers  
↓  
(data)

@Produces(MediaType.TEXT\_PLAIN)

@GET → HTTP method designators

public String greet(@QueryParam("name") String person){

return "Hello" + person + "!" ;

}

} . . .

→ Every Resource class should have min one resource method.

→ The above POJO can't be accessed over the n/w.

So, wrap it in a servlet by cfg if it's called

Bootstrapping code

. Until we don't have

servlet my resource won't be exposed.

web.xml

HttpServletDispatcher → Provided by Impl.

→ Class annotated with @Path is called Root Resource class.

- Convention: class name ends with Resource.
- Restful services completely rely on annotations.
- Any class annotated with `@Path` → Root class.
- we can send GET, POST, PUT, DELETE

`@Path("/greet")` → uri for addressing it.  
 public class GreetResource {  
`greet?person=John`

`@GET`

```
public String greet(@String person){  

  @QueryParam("name") String person}
```

`@GET` } HTTP method  
`@POST` request method  
`:` designations

→ In which way the parameter value will be comp.

→ Can be anything

which type of response method will send.

`@Produces("text/plain")`

↳ MIME/media type.  
 Instead of writing it as  
 its so jax-rs provided constants.

`@Produces(MediaType.TEXT_PLAIN)`

Always a GET req,  
 will sent data as string.

[Help → Eclipse Marketplace → search → Jboss Community  
→ select & install]

→ In Restful services only one endpoint exists  
i.e. Servlet endpoint.

New → Dynamic Web Project → FirstResteasyWeb →  
Target Runtime (b'coz resteasy jars were added) →

jboss 7.1 → create a new, local server →

Browse to jboss directory → next → finish next

→ next  Generate (web.xml) → finish

[check build path]

→ There's no wsdl so no contract first / last.

So, we'll start from java itself. We'll develop web  
resource not web service as it's accessible directly

over n/w.

com.fws.resource

| GreetResource

109-05-14  
Wednesday

jboss 8 is called ~~wildfly~~  
wildfly.

[cfg Env.]

(Runtime env).

→ DVD → jboss-as...final.zip →

sandbox folder (Extract to here).

→ This has old impl jars. Go to ~~jboss~~

extracted file → modules → [Jar Bundles →

resteasy → copy it → Extract to here →

open folder → resteasy-jboss-modules-3.0.5.Final]

→ paste → delete ~~resteasy~~ resteasy jars

→ Extract here → overwrite

[Developing App]

→ Use Eclipse Kepler

→ cfg jboss 7.1.1 runtime

window → preferences → servers →

new server runtime env.

JBoss

JBoss Community ✓

## Working with Resteasy ↴ impl

→ Eclipse (Kepler) IDE.

→ can integrate with JBoss AS 7.1.1  
(recommended)

→ jax-rs api. has been designed independent of app server (AS). It can work with any Servlet 2.5. api compliant server.

→ jdk 1.5 + (Annotations)

If there were no anno then more interface  
are reqd. So, eipi is very flexible.

(for running in Tomcat some config is reqd.)

→ Rest Easy binary distribution.

→ jboss 7.1.0 AS is pre cfg with jax-rs api 1.9

Replace server jars with latest jars.

→ Override server runtime libraries with latest

rest easy samp jars provided by binary distribution

## The supporting Restful Service

↳ Apache axis 2

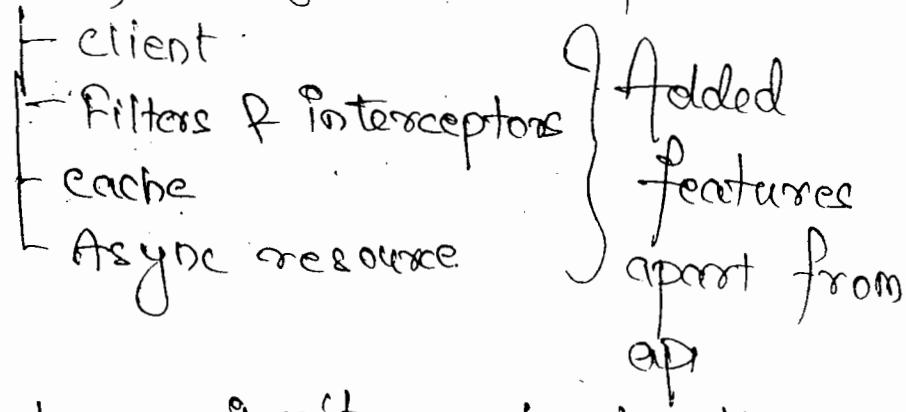
→ Apache CXF

↳ Spring f/w (muc)

Jersey 1.x for jax-rs 1.1 api

Jersey 2.0 for jax-rs 2.0 api

Resteasy for jax-rs 1.1 api



→ So, Resteasy isn't customization

now it's official.

## Bootstrapping Code

↳ Servlet cfg for getting the request.

→ If req. URL doesn't match with any of the existing resource it's treated as new req.

200, 201, 204.

## Restful Services

→ It's not specific to Java. It's a tech. by Roy Fielding.

→ Java released Jax-RS API (~~JAX-RS~~-8H)  
Initial version Jax-RS 1.1 (Only provider can be built)  
|  
+---+  
+ No Filters  
+ No Interceptors

→ Recent release Jax-RS 2.0 (~~JAX-RS~~-311)

Impl of Jax-RS API

|  
+---+  
+ Jersey (Sun) (popular)  
+ Resteasy (JBoss) — Can integrate with Spring fw.

## Http Status Code

### ⑤ POST

→ 1

exit

→ The action performed by post req mayn't result in ~~any~~ addressable resource. In this case,

it return either 200 (OK) or 204 (Content)  
w. res code. Success & response returned.

→ If new resource has been created 201 (Created).

↑  
Ops has finished  
successfully but  
doesn't return any  
response

→ 303 redirect user agent to retrieved cacheable resource. (Here, response is  
acted in client side)

→ 2

→ 3

→ 4

→ 5

→ 6

→ 7

### ⑥ PUT :

The req URI refers to an already existing resource then the req should be treated as for updated & either the 200 (OK) or 204 (No content) is returned to indicate success

http://localhost:8080/CRCoWeb/emp/update?Id=10

#### ④ HEAD:

- Similar to `[get]` instead of returning response body it returns http code associated with req. response.

#### ⑤ POST:

- Non-idempotent and non-safe method of HTTP
- Each POST req. modifies service in unique way.
- You may/mayn't sent info with req & receive info as response.

#### ⑥ OPTIONS:

- For req. info about comm options of the resource you're inserted in. Idempotent resource method.

`!Idempotent?`

- The side effects of N>0 req's same as sending 1 req.

Idempotent: Doing 1 time or any no. of time outcome will stay

of server will be same.

### ③ PUT:

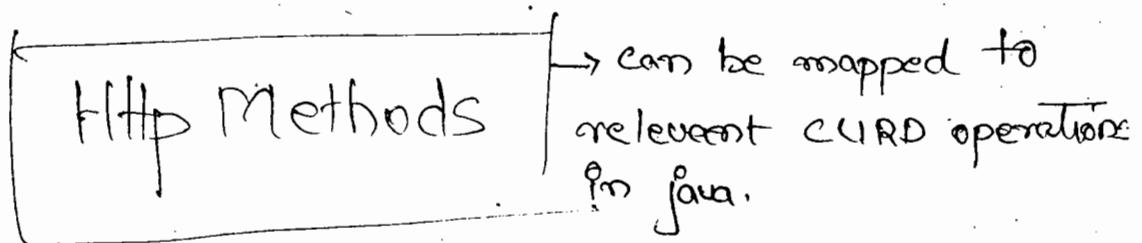
- Usually modelled as an Insert/update.
- Idempotent. Client remove Identity of res he's trying to <sup>create</sup> Insert / update.
- As part of PUT req. we have to send Identity of resource [empId].
- If empId = 1 then any no. of these updates it will have no effect on the underlying service, as only that obj gets affected.

### ④ DELETE:

- For removing the resource.
- Idempotent. If once deleted next time it will have no effect.

08-06-19  
Monday

→ Sun gave WAPL (Web Application Description Language). It describes services of web resources in an application.



## ① GET:

- Read - only operation.
- Used to query the info. from server.
- It's both idempotent & safe operation.
- Safe means invoking a GET doesn't change the state of server at all.

How data is sent

GET	POST
Query parameter string	Req body
Cookie	
Path parameters	
Matrix parameters	Binary + text + files sent
Request data	

02-06-14  
Monday

→ Sun gave WADL (Web Application Description Language). It describes services of web resources in an application.

## Http Methods

### ① GET:

- Read - only operation.
- Used to query the info. from server.
- It's both idempotent & safe operation.
- Safe means invoking a GET doesn't change the state of server at all.

How data is sent

POST

GET

Query parameter  
Path parameters  
Matrix parameters  
Only text

Req body  
Binary + text + files can be sent

02.06  
Monday

## (6) Hypertext As The Engine of Application State

(HATEOAS)

→ Unless you deeply study well you can't tell which method ~~to use~~ will suit your purpose.

→ Results are ~~result~~ returned based on search with hyperlinks.

→ Lets the data formats drive the state transitions for your application.

↳ one state to another state

## REST Architecture

→ Completely HTTP centric. No other protocol can be used.

→ Anyone who can send http req & receive http res can access. No ~~method~~ "This is terms called consumer."

→ Sun

Language

in

① G

→ R

→ L

→ D

→ Sate

st

C  
e  
F  
F  
F

Only te

→ If provider is SOAP 1.2 then consumer should use SOAP 1.2 which makes interoperability difficult.

→ In HTTP fixed methods are there making REST interoperable.

(7). No versioning b'coz of HTTP 1.0

→ Scalability is increased, we know which

method is doing what (Idempotent) by caching.

the req. data, at client side not server side.

### ③ Representation Oriented

→ Diff. clients may need diff. formats. Logic is same.

but representation can be varied.

→ Client tells through http header as Info using

accept-headers in which he tries to get the response. (Server uses content-type = " ")

→ This is called content negotiation.

### ④ Communication Statelessly

→ Nothing is stored on server rather at client.

(Less memory, no session mgmt → Performance high)  
at Local cache.

→ (Think in clustered env)

## Q) What is Restful Services?

(A) In Book.

→ Google & Amazon around 2007 shifted to Restful Services from WebServices.

(Representational State Transfer)

Set of architectural principles

→ Following principles defines REST:

→ In WS we need to use unit soap actions.

① Addressable resource. (every resource <sup>addressed by</sup> has URI)

② A uniform, constrained, interface.

→ Use a small set of well defined methods to manipulate your target resource.

③ Familiarity (no reqd. to see web to know service)

④ Interoperability (http protocol supported methods)

⑤ Scalability.

→ In web service we have web b'coz set of methods aren't limited.

constrained Interface = Limited methods.

→ In WebServices & CORBA aren't distributed

when following WS → b'coz vendor may or  
mayn't support it. It's Interoperable

→ There are two ways to secure your app impl security.

Custom  
WS Security.

① Security

Provider  
Consumer

→ WS-Security & SAML tells how to ~~call service~~ access

secured service. You have to continuously use Handler.

RESTFUL  
SERVICES

REST is principle used by Internet.

Services.xml

</operation>

<module ref="transport"/>

Given in daf.

→ There

⇒ Secur

E

01-05-14  
Sunday

→ WS-S

secur

① How you secured Web service in your project?

Used more

- (A)
- Authentication → WS-Security
  - Authorization → (not used generally)
  - Transport level Security (SSL)(Https) → Used more
  - Data Protection or Message level security, (Security) → (Encryption)  
(Message level Integrity) → (public & private key)
  - Digital Signatures (If msg is recd. from correct sender)

X

If you expose our service to flipkart & DTDC

We're exposing our service to flipkart & DTDC

DTDC then Authorization is failure.

RES

Runtime is calling it back

Passwort

class PasswordCallbackHandler & impl CallbackHandler {

Auth mech. cfg are passed as array  
public void handle (Callback[] callbacks) throws

UnsupportedCallbackException {

for (int i=0 ; i< callbacks.length ; i++)

if ((WSPasswordCallback) callbacks[i]) {

If (callbacks[i] == WSPasswordCallback::MSCHIEBER)

If (((WSPasswordCallback) callbacks[i]).getUsage() == )

username = ((WSP--) callbacks[i]).getIdentifier();

username

pwd = (""). getPassword();

if (username != null && username == ) {

return;

else

throw new UnsupportedCallbackException (callbacks[i], "msg");

}

## Apache Axis 2 Security

Passwort  
classe

→ Apache Axis 2 doesn't comes with Web Service

Security by default

→ Therefore 2 modules for security

[rampart-dist-1.6.1-bin]

└ modulus

{rabin-1.6.1

} 2 modules

rampart-1.6.1

→ Copy both modules goto runtime env.

Apache Tomcat → Axis2 → ~~WEB-INF~~

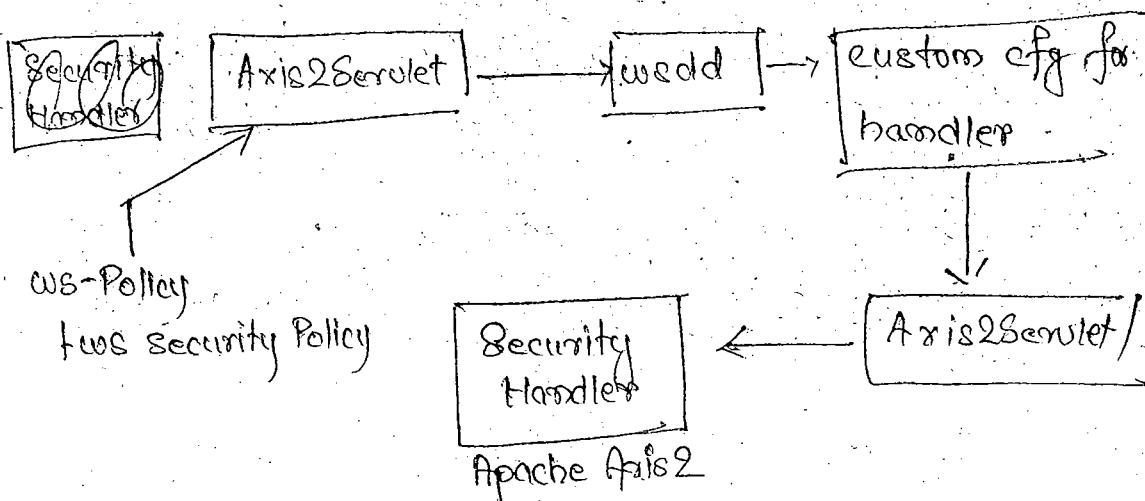
modules → paste

→ Paste rampart jars in runtime env.

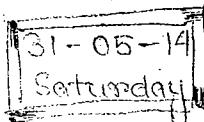
Contract First

Secured\_Passport

→ For containers independent impl: we need to use another mechanism. We need to write our **Security Handler** class.



→ we need to config one module for it.



Create **SecureTrainWS**

`oracle/wss-saml-or-username-token-service-policy`

**Security Policy**

OWSM Policies



② Oracle WebService

Security Manager

WLS Policy



WebLogic Server

No policy

Security Realms → User & Groups → Add

- Ⓐ Where can you see security cfg?
- Ⓐ wsdd
- Ⓐ Can I disable security?
- Ⓐ Can I disable security? at wsdl
- Ⓐ Yes, remove anno or at wsdl  
do it from admin console.

(definitions)

```
<wsp:policy id = "wsp1">
```

```
</wsp:policy>
```

```
<wsp:policy>
```

```
</wsp:policy>
```

```
<types>
```

```
<messages>
```

```
<portType>
```

```
<binding name = " --- ">
```

```
<wsp:policyReference id = "#wsp1">
```

→ For

use ↗

Security

Security Handler

WS-P "c"

fws s

→ Use ↗

31-05-  
Setup ↗

JBoss

create

Se ↗

O O W ↗

?

Secu ↗

→ In contract - last approach after running the tools write security cfg.

→ In contract - first

Follow the above

→ write in wsdl at start.

→ Container independent, like Apache Axis2

If deployed in Tomcat won't work

b'coz WS-\* specs relies on

container features (JAS... ) which Tomcat doesn't have.

WSDL

→ Web Service Security manager converts interoperable  
data to JAAS format.

① How consumer knows that service is secured?

④ WSDL, WS-Security policy detail will be  
cfg in <services> in wsdls & in corresponding  
<binding> as it may bind to multiple  
ports (either some of which might be secured)

<definitions> ~~<types>~~

<types/>

<message/>

<portType name = "pt1">

</portType>

<binding name = "ptsoap11Binding">

</binding>

<binding name = "ptsoap12Binding">

</binding>

<service>

</service>

</definitions>

## WS Auth Types

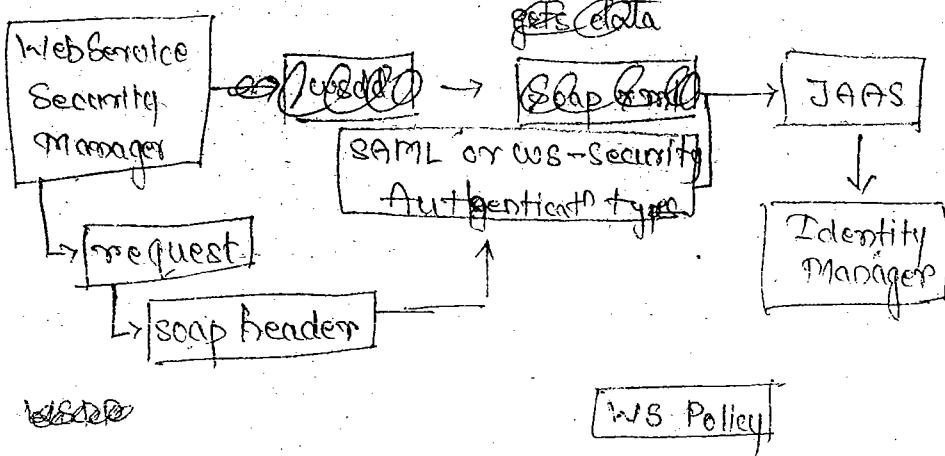
User Token policy over plaintext.

Passed as part of SOAP header

User Token policy over digest, encrypted with 80 bits

User Token over timestamp.

wsdd



→ WS Policy is set of standard

tags. It provides standard many

cfg how to ~~add~~ assertion (impose) on your

service.

→ WS Security Policy provides tags to tell ~~how~~

type of security. (How to impose)

WS Policy

Standard Markup language across impl.

wsPolicy  
<  
date

(@) How

(A) (1) E

cfg

< bios

port

< defini

ctyp

< me

< po

< pe

< bios

< h

< h's

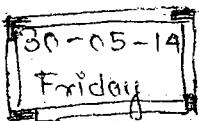
< h's

< em

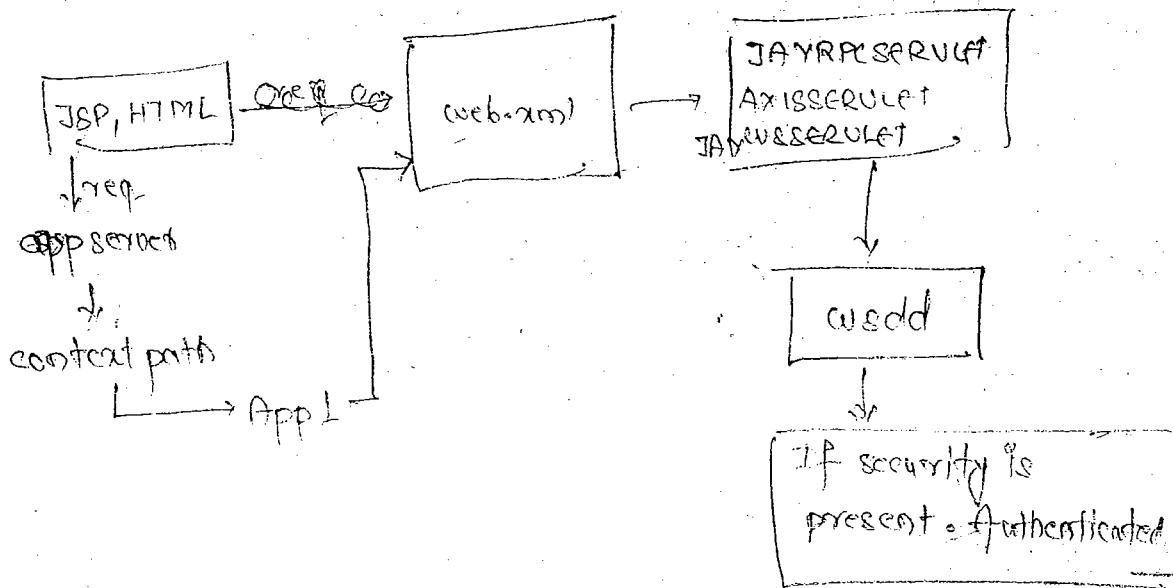
< s

< da

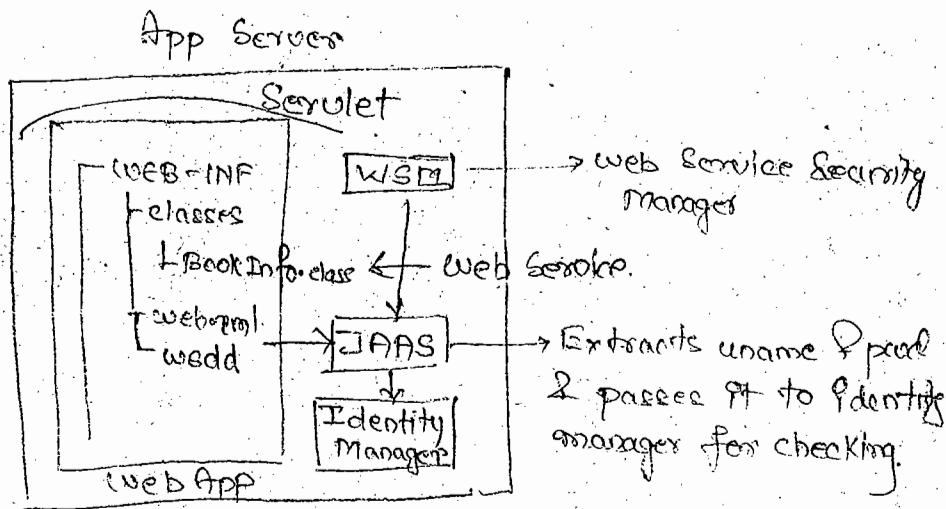
→ If .Net app. is trying to access JEE app.  
it can't pass b/c of diff. semantics of  
JEE security. So, WS-Security was given.



→ In JEE we write security for web.xml &  
for web services in wsdd.



→ Every app server has one component called  
Web Service Security Manager. Any app server supporting  
Web Service provides it.



→ If it can't

JEE se

30-0  
Friday

→ Since, our BookInfo webservice is pojo so it

can't get direct ref. from server. So, we will wrap it in Servlet (wrapper).

→ Impl. vendors will provide to user the wrapper & cfg in web.xml.

→ Info abt web service will be cfg in wsdd file?

This wsdd info will be provided to servlet.

→ For same servlet you can add multiple url pattern.

→ Auth. mechanism here is JEE. So, other app can't understand. It's not interoperable.

→ We can't leverage JEE security mechanism for webservice.

→ In JEE for

JSP, HT  
free  
app server  
context

→ Every  
Web 8

WebSer

## Authentication

- Basic
- Form Based
- Digest

username & pwd passed with ~~http body~~ <sup>header</sup>

written in

web.xml

username, pwd send from  
form req. body

↳ encoded/encoded  
unencoded

→ Server locates the resource & checks if there

any validation. Then control goes to JAAS, it needs

reads the authentication written in web.xml.

B/cuz JAAS engine is separate entity that manages  
security.

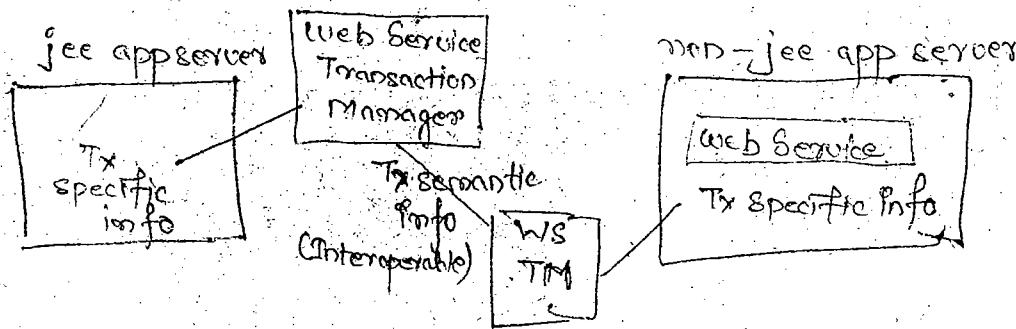
↓ JEE server component

→ JAAS passes control to Identity Manager which is  
running in LDAP server which stores username & pwd

is using hashing technique. Pwd in LDAP server is

encrypted & can't be easily converted to text. LDAP  
server is part of JEE container itself.

→ If username & pwd is matching then control is passed  
to servlet. LDAP is an obj. of Identity Manager.



→ WS - Transaction Manager was introduced by WS - Atomic Transaction.

Air b  
I  
L

→ @ Sec

anywhere

reads

B/c of

secure

→ JAAS

running

in user

encrypt

server

→ If ..

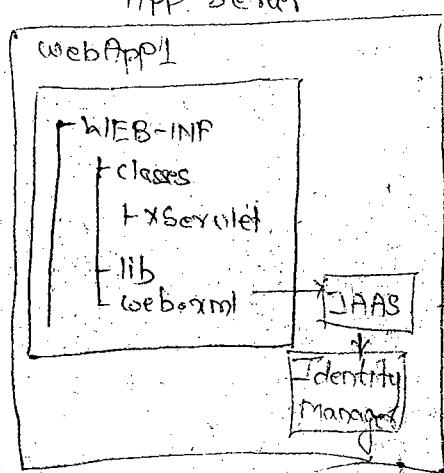
to ser

## How JEE App Security Works ?

→ You can impl security for web apps in 2 ways,  
in a env !.

① You can create your login form maintain  
a user-table & check for authenticated  
session if not redirect to login page.  
This is Custom security mechanism.

② If you're deploying in JEE app servers,



Java Authentication &  
Authorization Service.

29-05-14  
Thursday

within current transaction.

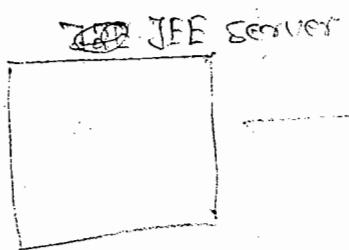
→ Servlet calls EJB, using JRMPI/IOP passing

transaction semantics if the EJB app fails it

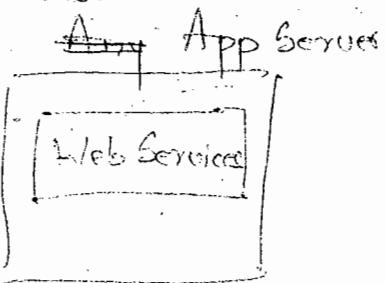
will tell servlet to rollback passing same

transaction semantics back to servlet.

### For HTTP



Other



can't understand JEE

transactional info by JEE

APP SERVER

→ If you want to enforce Tx mgmt then we ~~can't~~

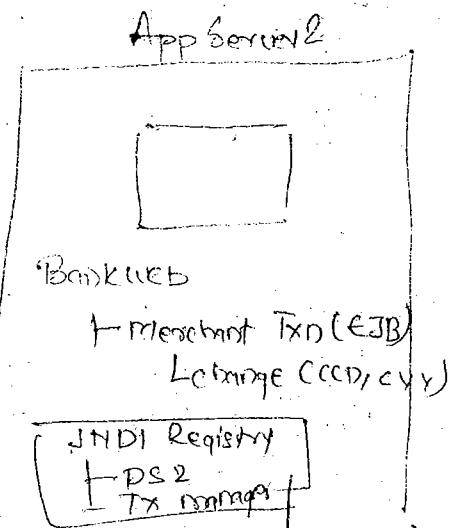
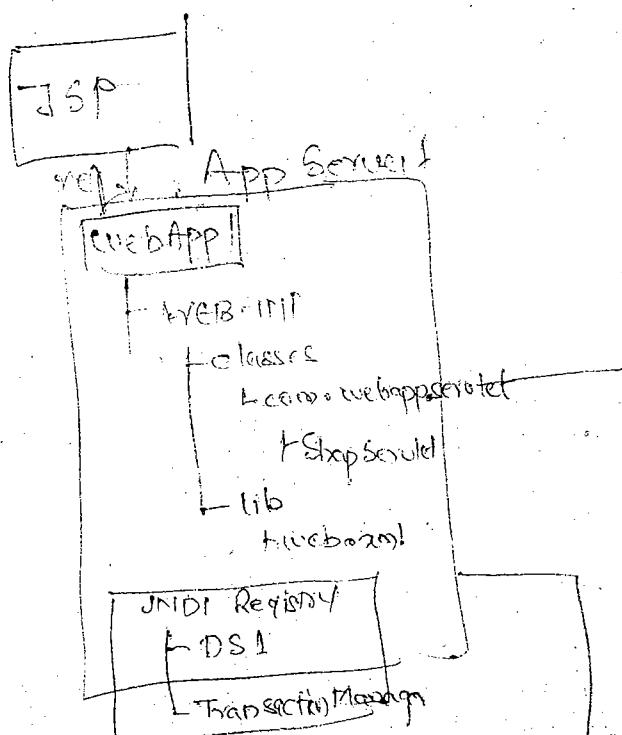
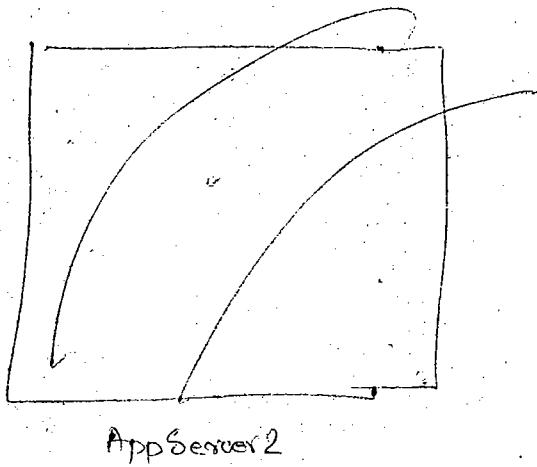
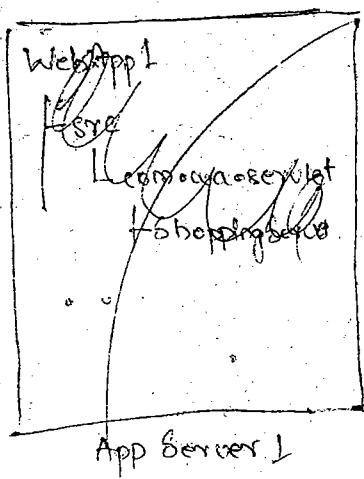
use JEE Server we have to use Interoperable

Tx Info.

→ Oracle give Web Service Atomic Tx Mgmt (WSAT)

[Organization for Advanced Information Structure]

Use Case: Transactional Boundaries exists  
in two diff application servers.



EJB uses RMP, IIOP  
protocol (stateful)

199-0  
Thrusi

→ Service

transac

wifl

transac

For

→ TPL

use

Tx P..

→ Oracle

Organ

→ We have to use JndiFactory to get jndi obj.

try {

InitialContext ic = new InitialContext();

UserTransactionManager tx = (UserTransactionManager) ic.lookup(  
("javax/UserTransaction"));

tx.beginTransaction();

ds1 = ic.lookup("jndi ds1");

ds2 = --- (ds2);

// get con, create stmt, execute stmt.

flag = false;

} catch (Exception e) {

flag = true;

} finally {

if (flag == true)

tx.rollback();

else

tx.commit();

}

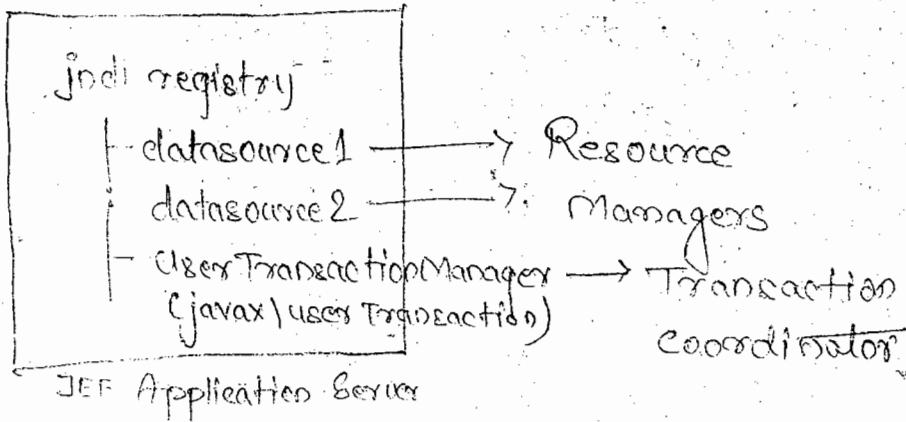
} catch (Exception e) {

From  
JTA

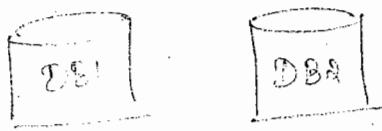
## Global Transaction in Typical JEE App

- Only JEE compliant server will support Global Transaction.
- To impl it in Tomcat 6 you need JTOM

↓↓↓↓



JEE Application Server



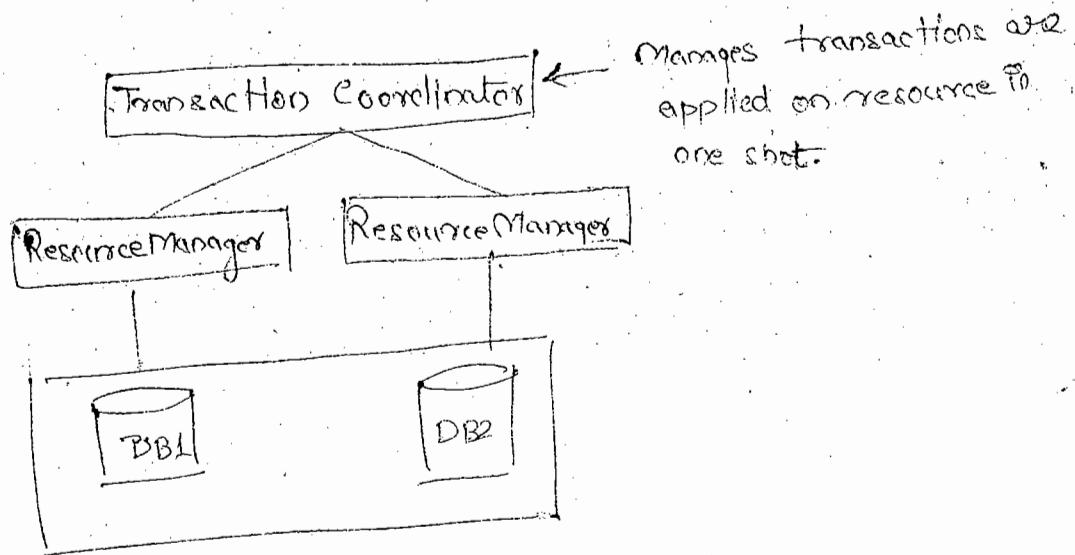
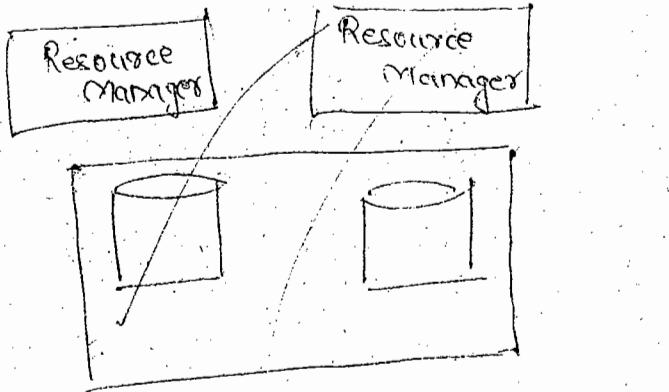
→ with JNDI registry for any server can object exists User Transaction Manager with JNDI name:

User Transaction javax/userTransaction

↑  
Transaction Coordinator

→ We h  
obj.  
try {  
Pointe  
UserT  
tx.t  
ds.  
ds.  
H.g.  
fing  
j.c.  
j.jc

locates



- Before using resource we have to begin tell Transaction Coordinator to begin Tx and with it the con obj created by Resource Manager needs to be passed.
- Transaction Coordinator will commit & rollback.

28-05-14  
Wednesday

## Jee Transactionality

- 2 ways to work with transactions in java

Local Tx -- Java jdbc api

↳ one resource participates in the transaction boundary

Global Tx (XA) -- JTA (Java Transaction Mgmt API)

↳ More than one resource participates

Multiple DB are involved in one transaction.

→ Before

Trans

gt

mecc

→ Tran

## Global Transaction (XA)

- In core java apps. you can ~~not~~ manage multiple resources in one transaction boundary.

- In Global Tx never a resource can commit if rollback by itself. Connection obj represents a DB

Third-party

means imposed

WSAML Specification (Security Assertion mark-up language)

→ SAML became popular. So, Microsoft event to IBM and released specifications

WS-\* Specification

- WS - Security
- WS - Atomic Transaction
- WS - Addressing
- WS - Reliable Messaging
- WS - Policy
- WS - ~~Security~~ Policy.
- WS - Trust.

} each spec addresses one enterprise class solution.

Microsoft + IBM found  
Oasis & released  
WS-\* specification.

Organization for Advanced Information  
Structure

→ jax-ws api and jax-ws api don't have enterprise class solution.

→ Apache Axis2 is based on jax-ws so you can add add'l features as modules.

## Evolution Of Web Services

### W3C - I Organization

BP 1.0      } Recommendation of standards that  
 BP 1.1      every prog lang. should support  
               to develop interoperable distributed  
               apps, enterprise class level app

BP 1.0

Jax-RPC API

BP 1.1

Jax-WS API

→ Web Services are used by others, so its

enterprise class level solution.

→ BP 1.0/1.1 addresses core concepts not  
enterprise level like Security, ~~Except~~ Transaction;

→ People realized it later.

⇒ OASIS (Organization for Standardized Advanced

→ SAML (Security Assertion Markup Language)

specification was designed by third party.

for how to implement security in webservice. The  
tags are standard XML tags transferred over

SOAP.

Third  
FSA

→ SAM

IBM

W<sup>r</sup>

→ W<sup>r</sup>

→ W<sup>r</sup>

→ W<sup>r</sup>

→ W<sup>r</sup>

→ W<sup>r</sup>

→ Jax

enterprise

→ Apis

you can

## Jax-rpc api

→ It's api so it's partial. We need impl + tools

to facilitate the development of web services

impl + tools  $\xrightarrow{\text{bundled}}$   
in s/w

For jax-rpc si and jax-rpc ri it's bundled  
in jwsdp.

→ c:\Oracle\Middleware\user-projects\domain

+ mydomain

+ lib  
+ bin

+ bat (tools)

Oracle ships bundle in

application server itself.

→ Some bundles it in binary distribution.

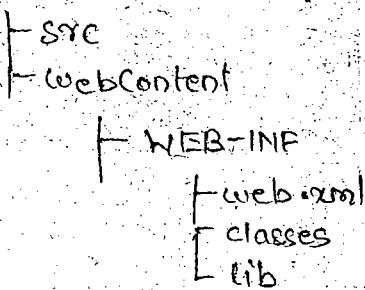
→ Oracle is commercial for (Weblogic server). So, it

provides infrastructure & add-on for app development.

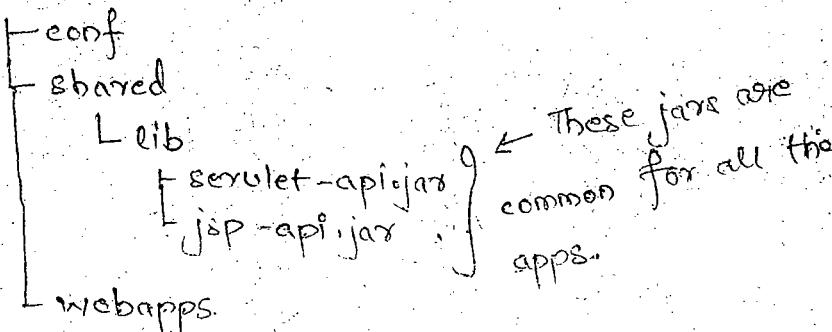
Bug fix support exists very well.

→ The jars shipped with Weblogic runs with Weblogic  
only.

## WebApp1



Apache - tomcat - 6.0.0



- In Eclipse if you don't config Target Runtime then the (common) shared jars won't be there in build path.
- Tomcat 6.0 supports only (Servlet 2.5) Dynamic web module version 2.5.
- In project classpath the Runtime jars will be available.
- Project Facets to know project web module version

Jax - >

→ It's ok

to forward

impl +

form to

in JSP

>c:\Oracle

Oracle

app's

→ Server

→ Oracle

provider

Bug

The J

only.

Pmr no : 119

Gret Pmr Status

27-05-14  
Tuesday

## Web Services - Security

### Implementation Types

→ Broadly classified into 2 types:

↳ j2ee container independent impl

↳ j2ee container specific impl.

j2ee container independent impl.

jax-rpc si

jax-ws ri

apache axis

apache axis2

apache cxf

↳ Any Servlet 2.5+ container  
will run it.

j2ee container specific impl

Oracle weblogic web services

IBM websphere web services

JBoss web services [open source]

Runtime has jars so  
server is a part  
in the impl name.

→ Next → SOAP1.2 Binding → MEF? - Document Utilities

→ Next → Available Methods:  → Next

→ Additional class & I/O classes → Security:  NO

→ Handlers:  → Finish.

→ Add @WebParam & @WebResult to web service

method ID & EI.

http://localhost:7101/console

Username:

Password:  ~~pass~~

## Deployment:

Right click on your project → Deploy web services

→ Application Server → Integrated Weblogic Server → next →

next → Admin Console → Deployments

+ web service app - Routing Policy:

↓ Service

✓ wsdl

→ ~~test~~ / testclient

→ whenever you create Application it asks you which project you want to develop.

→ New → Application → [New Folder] → [Developer] →

[Web Service Application] → Generic Application →

Application Package Prefix : [com.wsapp] → Next →

Project name : [RailEnquiry] → Project Technologies : [Web Services]

→ Next → Finish.

[It hides empty folders]

Right click on RailEnquiry → new → Java → class

class : [ProEnquiry], package : [com.wsapp.service]

public class ProEnquiry {

    public String getProStatus (String pos) {

        return "confirmed";

}

[ProEnquiry] → Right click → create web service

JAX-WS Anno → next →

Service Name : ProEnquiry Service

port : [ ]

sei : [ ]

For integrated weblogic server default port is  
7001.

→ Whenever you create a new project

Default username: weblogic.

→ New ->

Default password: weblogic1 or welcome1

Web S...

If iTunes is installed, weblogic fails to start. Bcoz it adds an env var `-DQTZIP` pointing to `.../qt.zip`.

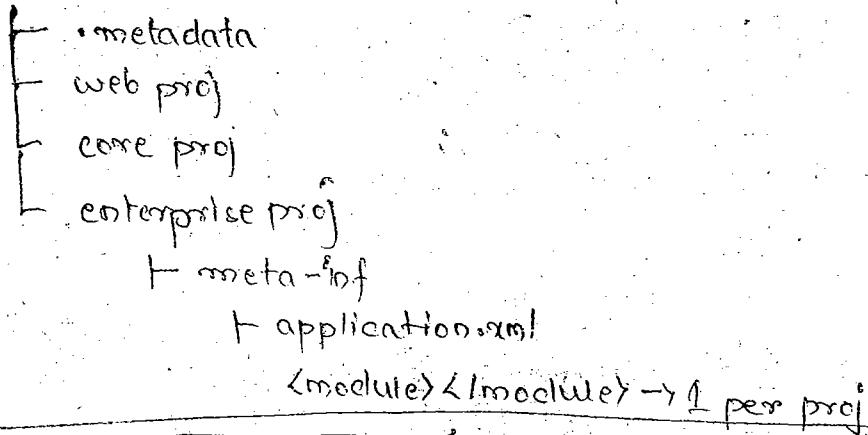
Applicat...

Project

→ DS

### Eclipse workspace

c:\workspace (we develop app here)



[At hi]

Righ...

ctrls

publi...

c:\workspace ← [JDeveloper]

Application (Enterprise app) (1 project = 1 Application)

web

java

web

→ J2EE encourages Enterprise Application development.

Princi...

→ If you collectively want to deploy use above

structure else remove Application folder.

→ jdele JDeveloper 11.1.1.6.0. (batch release) Generic

It works on any platform.

→ Double click jvm will directly execute the jar.

→ Go to start → Oracle Fusion Middleware → JDeveloper Studio

→ Go to menu → Start Server Instance (Integrated Weblogic)

→ Dialog window →

Tomcat

+ webapps

All apps have same  
uname & pwd & everyone  
can see it.

Middleware

+ wlserver (weblogic server installation)  
+ bin  
+ config.xml (for creating  
domain)

• Domain contains specific server for app to run  
(separate security, datasource...)

For each separate apps you can create separate  
domain for uname & pwd. Security

Middleware\user-projects\domain

+ mydomain (our created domain)

+ bin

+ server

You can create domain anywhere by default its in  
user-projects

26-05-14  
Monday

→ jboss

gt.

→ Dev

→ Gia :

→ Glassfish

→ by

[jwsdp-2.0] (jdk 1.5)

| saaj (1.2) + jax-p  
| jax-b(1.x)  
| Jaxrpc-si  
| jaxws - si (1.x)

[jwsdp 2.0] (jdk 1.6) — upgraded APIs

| jax-p  
| jax-b(2.x)  
| saaj (1.3)

All APIs  
uname  
can :

④ Dev

For  
dev

Mick

You  
use

→ Since, jdk API's are different, web services developed on jwsdp 2.0 + jdk 1.5 won't work in jwsdp 2.0 + jdk 1.6.

→ For jwsdp 2.0 + jdk 1.6 based web services are directly supported by GlassFish server. Tools are there in bin directory & Netbeans.

→ Netbeans & GlassFish are replaced by JDeveloper & Intellicode.

DURGA SOFTWARE SOLUTIONS

# Additional Notes

Apache CXF

**Mr. Sriman**

As part of this content we are going to how to work on Apache CXF Provider as well as consumer side using Spring Integration.

## Contents

<b>1 JAX-WS API (APACHE CXF IMPLEMENTATION) .....</b>	<b>3</b>
1.1 UNDERSTANDING APACHE CXF DISTRIBUTION.....	3
1.2 BUILDING PROVIDER.....	4
1.3 BUILDING CONSUMER .....	18

# **Apache**

# **CXF**

## 1 JAX-WS API (Apache CXF Implementation)

Apache CXF is an open source web service framework. CXF helps you build and develop web services using frontend programming APIs, like JAX-WS and JAX-RS.

It can work on various protocols like SOAP/HTTP, XML/HTTP, RESTful/HTTP, or CORBA and work with variety of protocols like HTTP, JMS and JBI.

Some of the features/advantages of Apache CXF:

- a) It supports various frontends like JAX-WS and JAX-RS
- b) Ease of use, strong tooling support is there to work with contract-first or contract-last approach.
- c) Little amount of code is being generated when compared with other frameworks.
- d) Eventually going to replace Apache Axis2
- e) Easy to integrate with Spring framework (makes spring as a first-class citizen)
- f) CXF supports variety of Web Service Standards including SOAP, the WS-I Basic profile, WSDL, WS-Addressing, WS-Security (WSSE), WS-Policy, WS-ReliableMessaging, WS-SecurityPolicy, WS-SecureConversation, WS-Trust etc.
- g) It can be deployable on any Web Application Container (Container Independent implementation)

### 1.1 Understanding Apache CXF distribution

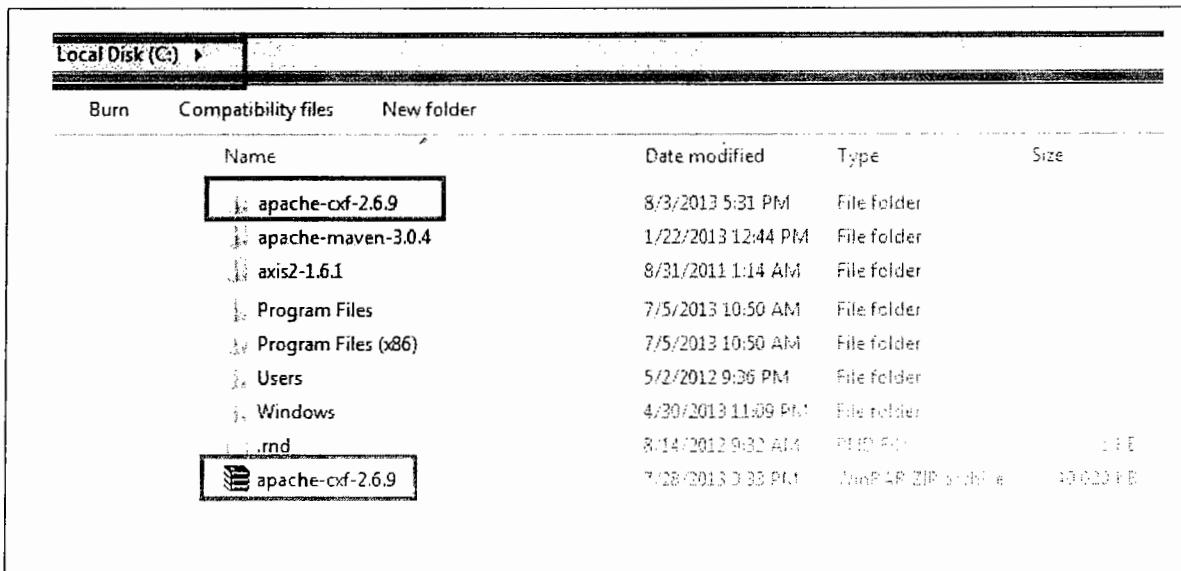
If you want to develop Apache CXF based web service or consumer, you need to download the Apache CXF Binary Distribution. It is a zip containing Jar's and Tool's that are required to develop Apache CXF based provider or consumer.

Here is the link to download Apache CXF distribution.

→ <http://cxf.apache.org/download.html>

As we mentioned the current release is 2.7.6. We are using in this examples the 2.6.9 version.

Once you download the binary zip distribution apache-cxf-2.6.9.zip, copy it to the c:\drive and extract it.



The binary distribution will contain the following directory structure.

```
apache-cxf-2.6.9
| - bin      - Contains all the tools wsdl2java.bat, java2wsdl.bat,
  wsdlvalidator.bat etc
| - docs    - java docs
| - etc
| - lib      - All the jars
| - modules
| - samples - all the samples of apache cxf
```

You have completed setting up the tools/environment; to build apache cxf based web services.

### 3.2 Building Provider

In building a provider always there are two approaches. Contract First and Contract Last. Let's try to build a contract first based developer using Apache CXF.

Now we are trying to develop JAX-WS API, Apache CXF implementation based provider using Servlet endpoint, using Contract First approach, using Sync Req-Reply based provider using document-literal based web service.

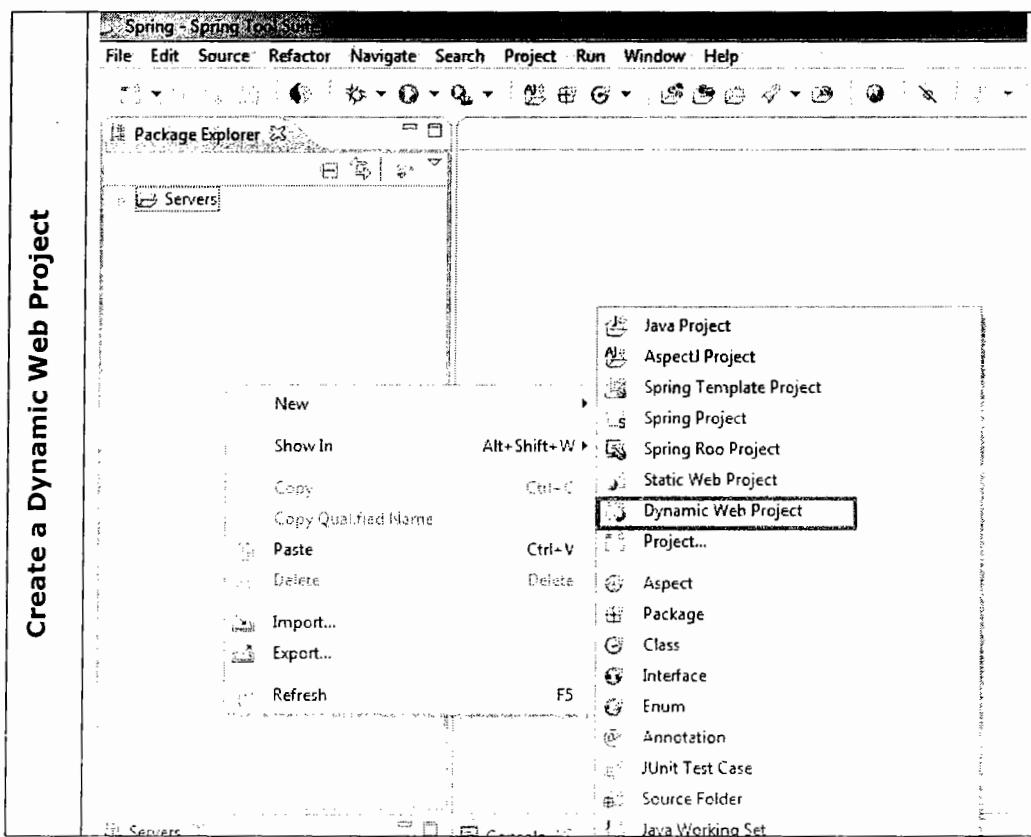
- Write WSDL Document** – As it is contract first based approach always the development will starts with WSDL document. Now we need to write the WSDL document with document-literal as the message exchanging pattern.

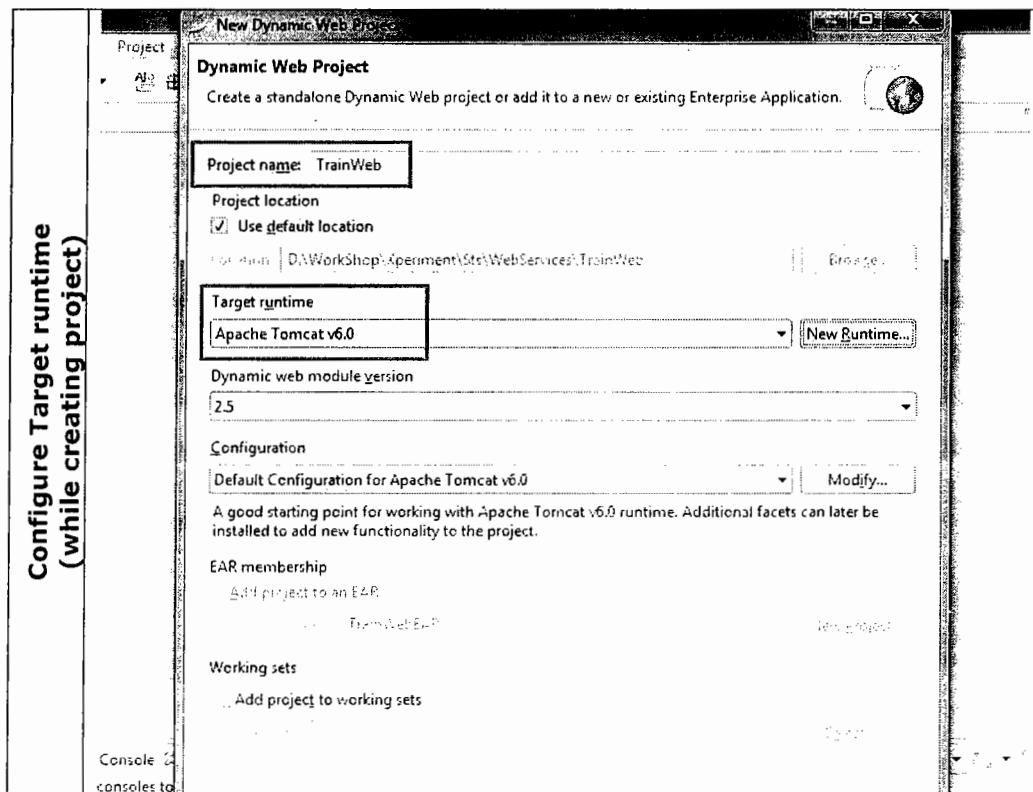
Let us consider the below traininfo.wsdl document for developing the service.

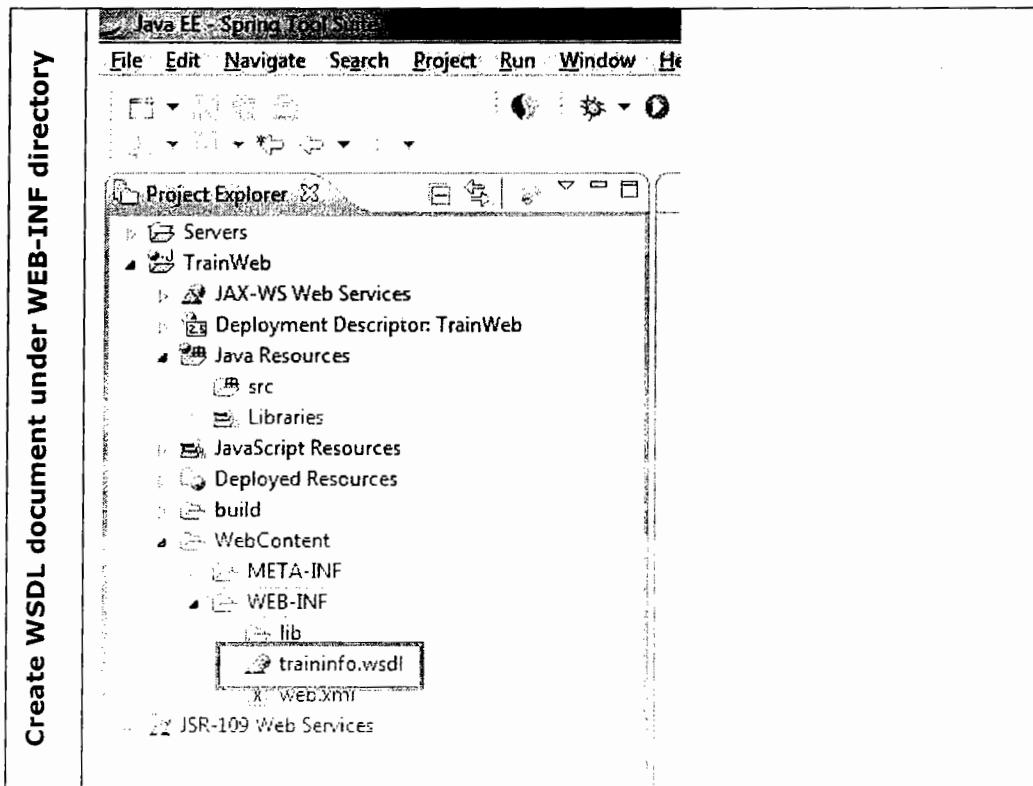
**traininfo.wsdl**

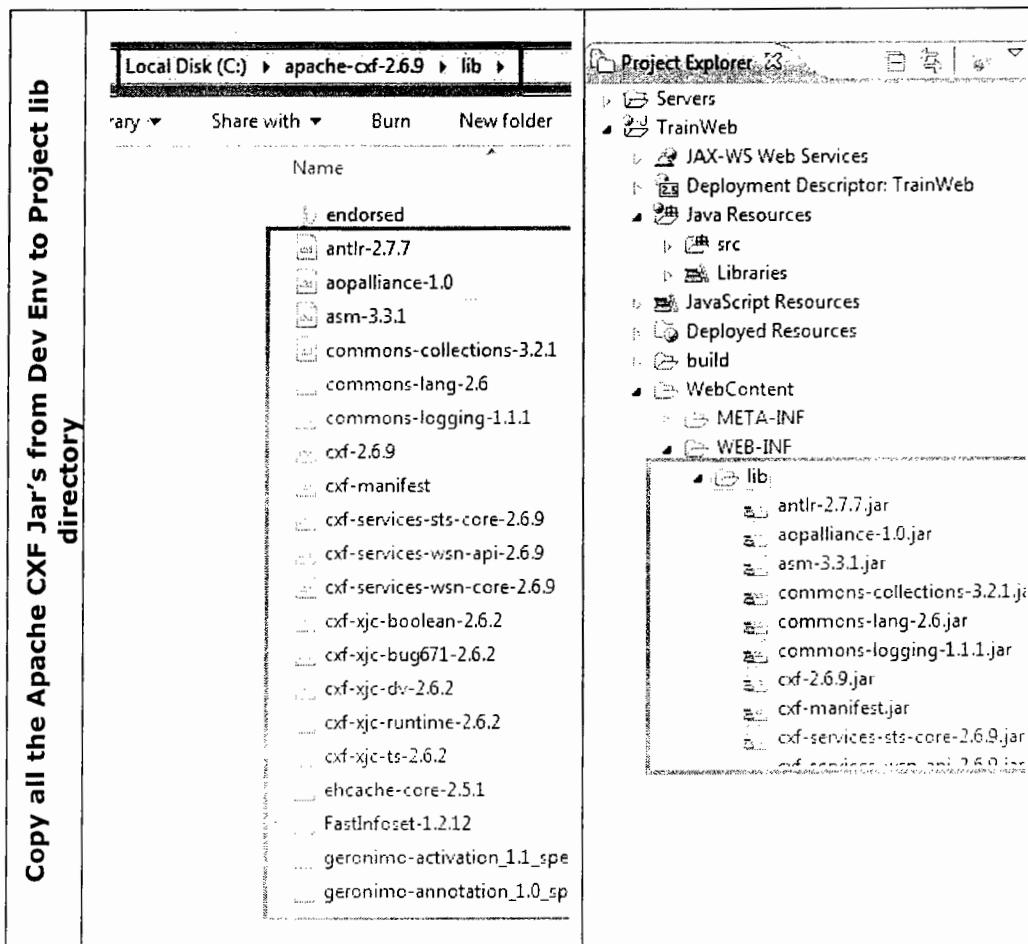
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://irctc.co.in/reservation/wsdl"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="traininfo"
    targetNamespace="http://irctc.co.in/reservation/wsdl"
    xmlns:it="http://irctc.co.in/reservation/types">
    <wsdl:types>
        <xsd:schema
            targetNamespace="http://irctc.co.in/reservation/types">
            <xsd:element name="ReservationInfo">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="ssn"
                            type="xsd:string" />
                        <xsd:element name="name"
                            type="xsd:string" />
                        <xsd:element name="trainNo"
                            type="xsd:string" />
                        <xsd:element name="src"
                            type="xsd:string" />
                        <xsd:element name="dest"
                            type="xsd:string" />
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Ticket">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="pnr"
                            type="xsd:string" />
                        <xsd:element name="coach"
                            type="xsd:string" />
                        <xsd:element name="berth"
                            type="xsd:string" />
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:schema>
    </wsdl:types>
```

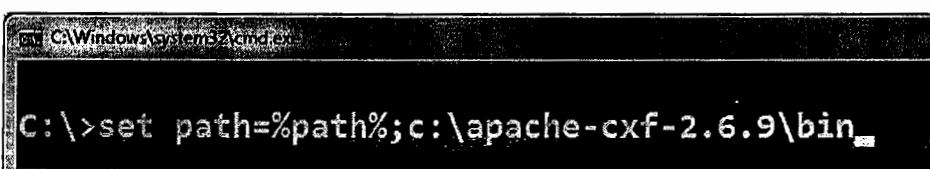
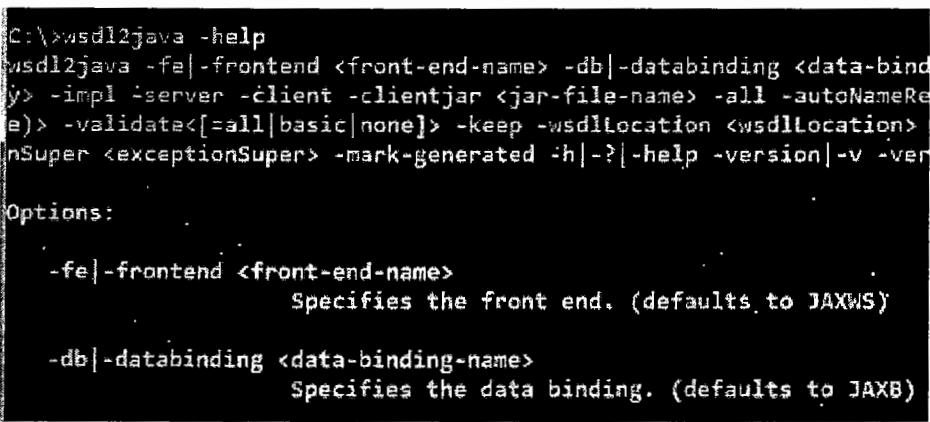
```
<wsdl:message name="reserve">
    <wsdl:part element="it:ReservationInfo" name="in" />
</wsdl:message>
<wsdl:message name="reserveResponse">
    <wsdl:part element="it:Ticket" name="out" />
</wsdl:message>
<wsdl:portType name="TrainInfo">
    <wsdl:operation name="reserve">
        <wsdl:input message="tns:reserve" />
        <wsdl:output message="tns:reserveResponse" />
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="TrainInfoSOAPBinding" type="tns:TrainInfo">
    <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="reserve">
        <soap:operation
            soapAction="http://irctc.co.in/reservation/wsdl#reserve" />
        <wsdl:input>
            <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal" />
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="TrainInfoService">
    <wsdl:port binding="tns:TrainInfoSOAPBinding"
      name="TrainInfoSOAPPort">
        <soap:address location="http://www.example.org/" />
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

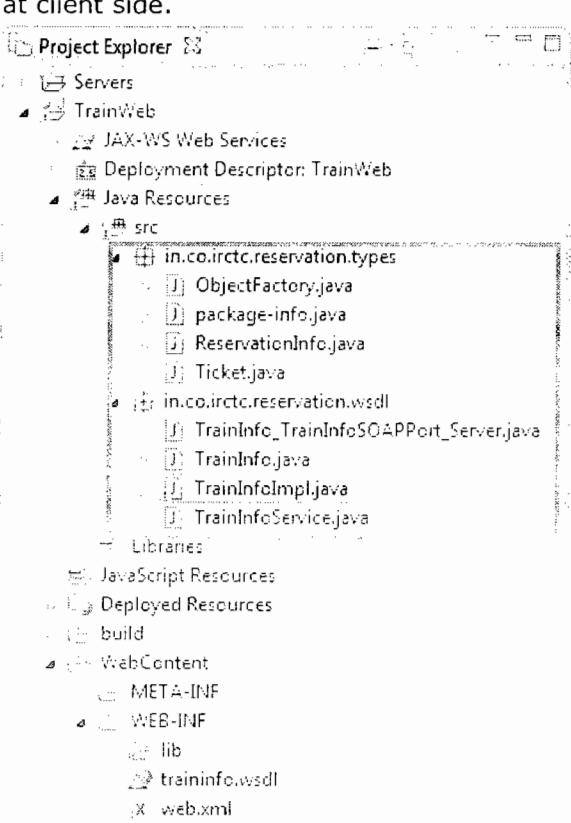


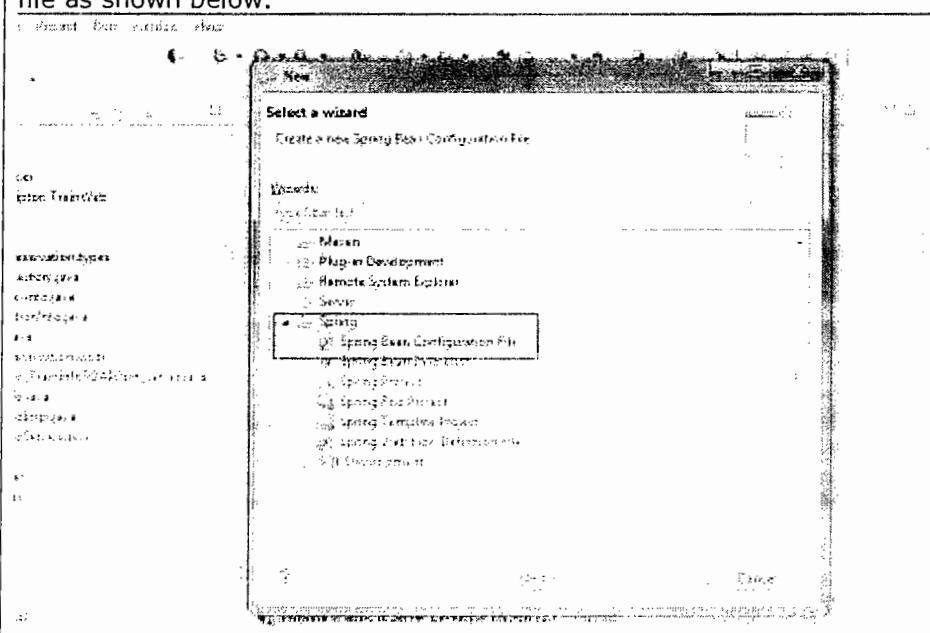






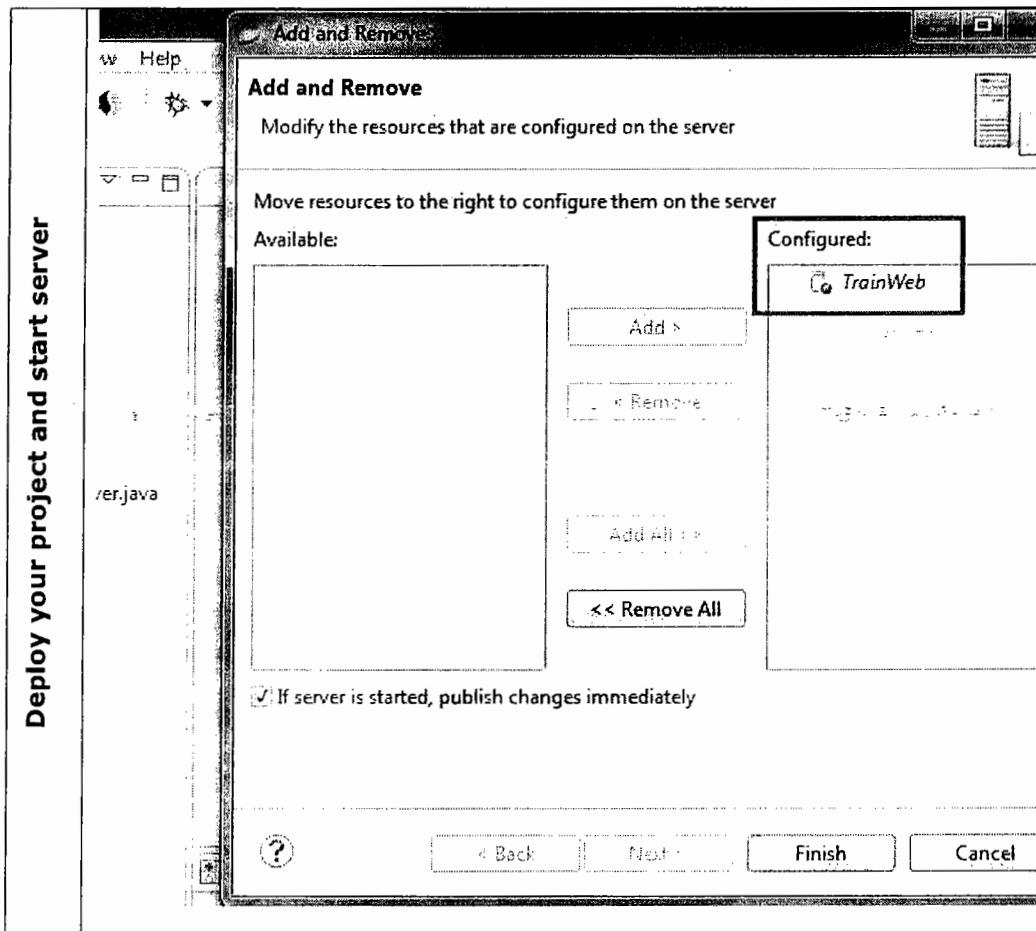
<b>Run WSDL2Java</b>	<p>As it is contract first approach, once we are done with writing the WSDL document, we need to generate binding classes, to generate binding classes from WSDL, we need to run WSDL2Java tool that is provided by Apache CXF.</p> <p>This tool is provided to us as part of Binary Distribution which we configured as development environment in c:\ drive. Now setup the path pointing to the directory c:\apache-cxf-2.6.9\bin</p>  <p>Verify whether you are able to run the tool or not</p> 								
<b>Switches</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;">-d src</td><td style="padding: 5px;">Specify the directory, where you want to generate those files under</td></tr> <tr> <td style="padding: 5px;">-impl</td><td style="padding: 5px;">A dummy impl class is generated</td></tr> <tr> <td style="padding: 5px;">-server</td><td style="padding: 5px;">Server side code is generated</td></tr> <tr> <td style="padding: 5px;">-frontend jaxws21</td><td style="padding: 5px;">By default the api we use is jax-ws2.1. If you want to use jax-ws2.2 then you need to copy the endorsed jars as well. To skip the Jax-ws 2.2 specific methods use this switch</td></tr> </table>	-d src	Specify the directory, where you want to generate those files under	-impl	A dummy impl class is generated	-server	Server side code is generated	-frontend jaxws21	By default the api we use is jax-ws2.1. If you want to use jax-ws2.2 then you need to copy the endorsed jars as well. To skip the Jax-ws 2.2 specific methods use this switch
-d src	Specify the directory, where you want to generate those files under								
-impl	A dummy impl class is generated								
-server	Server side code is generated								
-frontend jaxws21	By default the api we use is jax-ws2.1. If you want to use jax-ws2.2 then you need to copy the endorsed jars as well. To skip the Jax-ws 2.2 specific methods use this switch								

<b>Change to the Project directory and run the tool</b>	<pre>C:\&gt;cd D:\WorkShop\Xperiment\Sts\WebServices\TrainWeb</pre> <p><b>Command:</b> wsdl2java -d src -server -impl -verbose -frontend jaxws21 WebContent\WEB-INF\traininfo.wsdl</p> <pre>D:\WorkShop\Xperiment\Sts\WebServices\TrainWeb&gt;wsdl2java -d src -se Loading FrontEnd jaxws ... Loading DataBinding jaxb ... wsdl2java -d src -server -impl -verbose WebContent\WEB-INF\traininf wsdl2java - Apache CXF 2.6.9  D:\WorkShop\Xperiment\Sts\WebServices\TrainWeb&gt;</pre>
<b>Generates SEI Interface, Impl and Input/output classes</b>	<p>It generates Input/Ouput classes, SEI Interface, Dummy Implementation class, xxx_Server.java to publish the service and one Service class used at client side.</p>  <p>The screenshot shows the Eclipse IDE's Project Explorer view. The 'TrainWeb' project is expanded, revealing its structure. Under 'Java Resources &gt; src', there is a package named 'in.co.irctc.reservation.types' containing four files: ObjectFactory.java, package-info.java, ReservationInfo.java, and Ticket.java. Below this package is another named 'in.co.irctc.reservation.wsdl' containing four files: TrainInfo_TrainInfoSOAPPort_Server.java, TrainInfo.java, TrainInfoImpl.java, and TrainInfoService.java. There are also 'Libraries', 'JavaScript Resources', 'Deployed Resources', and 'build' folders. Under 'WebContent', there are 'META-INF', 'WEB-INF' (containing a 'lib' folder with 'traininfo.wsdl'), and a 'web.xml' file.</p>

<p><b>Provide the Implementation in Impl class</b></p>	<pre> /*  * (non-Javadoc)  *  * @see  * in.co.irctc.reservation.wsdl.TrainInfo#reserve(in.co.irctc.reservation  * .types.ReservationInfo in )  */ public in.co.irctc.reservation.types.Ticket reserve(     in.co.irctc.reservation.types.ReservationInfo in) {     LOG.info("Executing operation reserve");     try {         in.co.irctc.reservation.types.Ticket _return = new Ticket();         _return.setPnr("Pnr1131");         _return.setBerth("S1");         _return.setCoach("24");         return _return;     } catch (java.lang.Exception ex) {         ex.printStackTrace();         throw new RuntimeException(ex);     } } </pre>
<p><b>Create spring beans configuration file</b></p>	<p>Right click on WEB-INF directory and create spring beans configuration file as shown below.</p> 

<b>Give the name of the file as cxf-services.xml and import the jaxws namespace</b>	
<b>Import two additional xml files provided by cxf</b>	<pre> &lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;beans xmlns="http://www.springframework.org/schema/beans"        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:jaxws="http://cxf.apache.org/jaxws"        xsi:schemaLocation="http://www.springframework.org/schema/beans                            http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd"&gt;      &lt;bean id="cxf" resource="classpath:META-INF/cxf/cxf.xml" /&gt;     &lt;bean id="cxf-servlet" resource="classpath:META-INF/cxf/cxf-servlet.xml" /&gt;  &lt;/beans&gt; </pre>

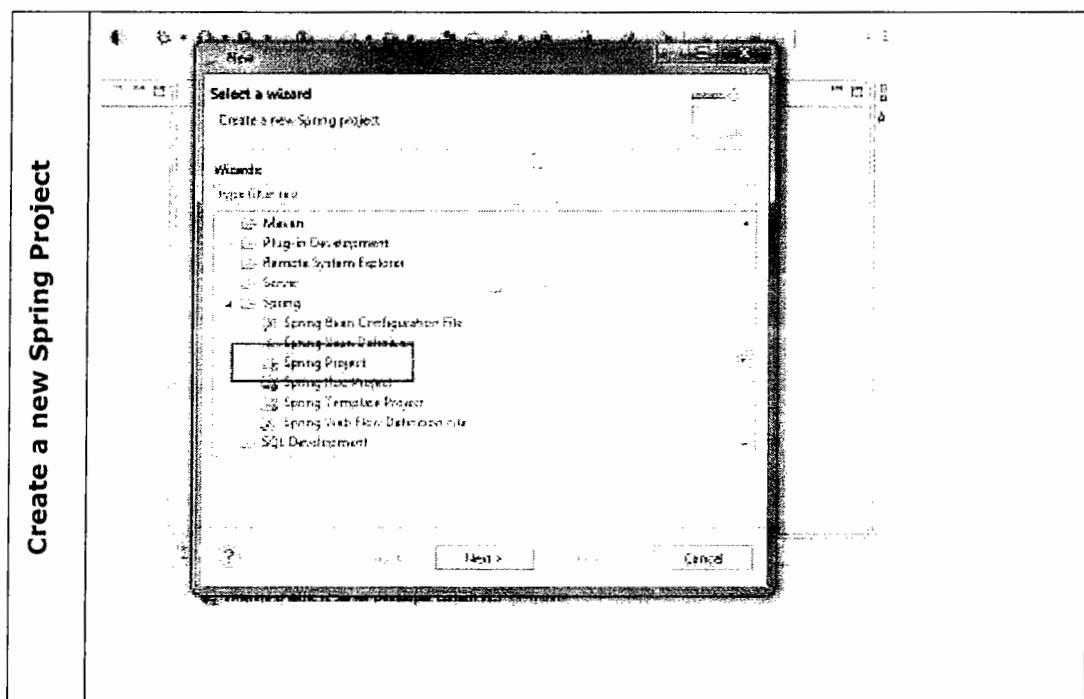
<b>Configure Web.xml</b>	<p><b>Configure the implementation as jaxws: endpoint bean</b></p> <pre> &lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"        xmlns="http://java.sun.com/xml/ns/jaxws" xmlns:web="http://java.sun.com/xml/ns/javaee/web"        xsi:schemaLocation="http://java.sun.com/xml/ns/jaxws http://java.sun.com/xml/ns/javaee/web        http://java.sun.com/xml/ns/jaxws_2_2" version="2.2"&gt;     &lt;jaxws:endpoint name="TrainWeb/TrainInfoService"&gt;         &lt;jaxws:parameters&gt;             &lt;param name="contextConfigLocations"/&gt;         &lt;/jaxws:parameters&gt;         &lt;jaxws:bindings&gt;             &lt;jaxws:binding name="jaxws:jaxws-configuration"&gt;                 &lt;jaxws:implementation-class&gt;org.springframework.web.context.ContextLoaderListener&lt;/jaxws:implementation-class&gt;             &lt;/jaxws:binding&gt;             &lt;jaxws:binding name="jaxws:cxf-service-servlet"&gt;                 &lt;jaxws:implementation-class&gt;org.apache.cxf.transport.servlet.CXFServlet&lt;/jaxws:implementation-class&gt;                 &lt;jaxws:properties&gt;                     &lt;property name="CXFServlet&amp;gt;&lt;/property&gt;                     &lt;property name="CXFServlet&amp;gt;&lt;/property&gt;                 &lt;/jaxws:properties&gt;             &lt;/jaxws:binding&gt;         &lt;/jaxws:bindings&gt;     &lt;/jaxws:endpoint&gt; &lt;/beans&gt; </pre>
	<p><b>Configure web.xml</b></p> <ol style="list-style-type: none"> <li>1) Configure ContextLoaderListener which will takes the cxf-services.xml as input to expose your jaxws endpoint as bean</li> <li>2) Configure CXFServlet which will acts as an servlet endpoint in processing the request and dispatching the response as shown below.</li> </ol>

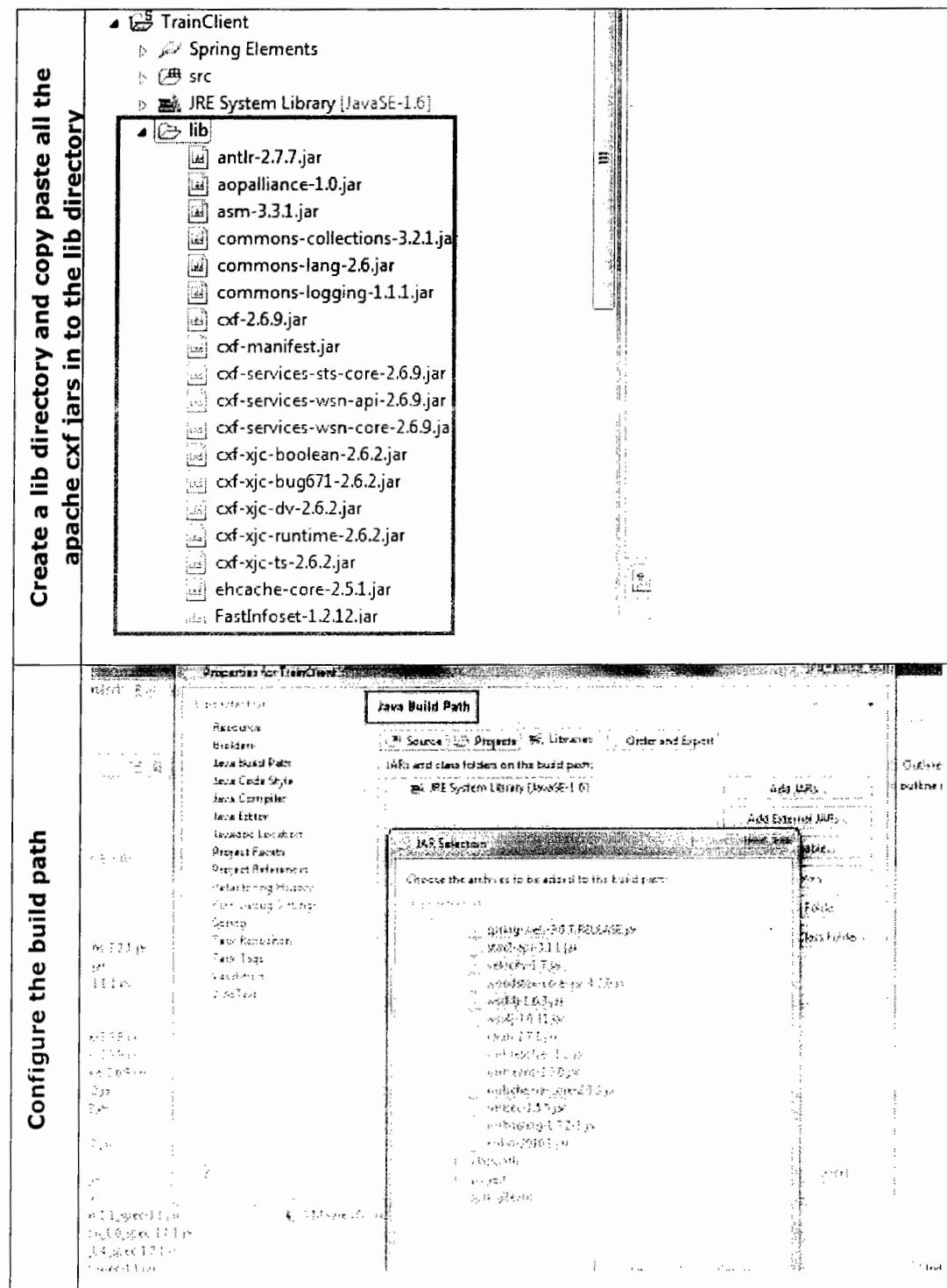


<b>Test by accessing the wsdl url</b>	<pre> &lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;wsdl:definitions targetNamespace="http://irctc.co.in/reservation/wsdl" name="traininfo" xmlns:it="http://schemas.xmlsoap.org/wsdl/types/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://irctc.co.in/reservation" xmlns:xsd="http://www.w3.org/2001/XMLSchema"&gt;     &lt;wsdl:types&gt;         &lt;xsd:schema targetNamespace="http://irctc.co.in/reservation/types" xmlns:it="http://irctc.co.in/reservation" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://irctc.co.in/reservation" xmlns:xsd="http://www.w3.org/2001/XMLSchema"&gt;             &lt;xsd:element name="ReservationInfo"&gt;                 &lt;xsd:complexType&gt;                     &lt;xsd:sequence&gt;                         &lt;xsd:element name="ssn" type="xsd:string"/&gt;                         &lt;xsd:element name="name" type="xsd:string"/&gt;                         &lt;xsd:element name="trainNo" type="xsd:string"/&gt;                         &lt;xsd:element name="src" type="xsd:string"/&gt;                         &lt;xsd:element name="dest" type="xsd:string"/&gt;                     &lt;/xsd:sequence&gt;                 &lt;/xsd:complexType&gt;             &lt;/xsd:element&gt;             &lt;xsd:element name="Ticket"&gt;                 &lt;xsd:complexType&gt;                     &lt;xsd:sequence&gt;                         &lt;xsd:element name="pnr" type="xsd:string"/&gt;                         &lt;xsd:element name="coach" type="xsd:string"/&gt;                         &lt;xsd:element name="berth" type="xsd:string"/&gt;                     &lt;/xsd:sequence&gt;                 &lt;/xsd:complexType&gt;             &lt;/xsd:element&gt;         &lt;/xsd:schema&gt;     &lt;/wsdl:types&gt; </pre>
<b>Test service using SOAP UI</b>	<p>The screenshot shows the SoapUI interface with a request message defined:</p> <pre> &lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns2="http://irctc.co.in/reservation/wsdl"&gt;     &lt;soapenv:Header/&gt;     &lt;soapenv:Body&gt;         &lt;ns2:Ticket xmlns:it="http://schemas.xmlsoap.org/wsdl/types/"&gt;             &lt;pnr&gt;Pnr1131&lt;/pnr&gt;             &lt;coach&gt;24&lt;/coach&gt;             &lt;berth&gt;S1&lt;/berth&gt;         &lt;/ns2:Ticket&gt;     &lt;/soapenv:Body&gt; &lt;/soapenv:Envelope&gt; </pre>

### 1.3 Building Consumer

In order to access a provider always the input for the consumer is WSDL. Check whether the WSDL is available. Follow the steps in accessing the provider.





**Run the wsdl2java tool to generate the client side classes**

Open the command prompt and configure the path pointing to the directory where wsdl2java tool is there.

```
C:\Windows\system32\cmd.exe
C:\>set path=%path%;c:\apache-cxf-2.6.9\bin
C:\>
```

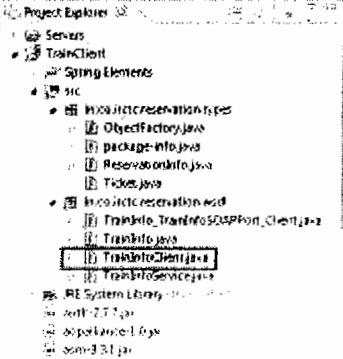
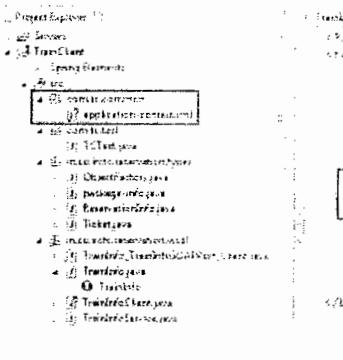
Switch to the project directory

```
D:\WorkShop\Xperiment\Sts\WebServices\TrainClient>...
```

Run the tool with the following switches

**Command:** wsdl2java -d src -client -verbose -frontend jaxws21 wsdlurl

```
D:\WorkShop\Xperiment\Sts\WebServices\TrainClient>wsdl2java -d src -client -verb
Loading FrontEnd jaxws21 ...
Loading DataBinding jaxb ...
wsdl2java -d src -client -verbose -frontend jaxws21 http://localhost:8081/Trainw
wsdl2java - Apache CXF 2.6.9
```

<p><b>Create a Client class in the project</b></p>	<p>Refresh the project and create one more class TrainInfoClient bean, into which you need to inject TrainInfo as port.</p>  <pre>package in.co.irctc.reservation.wsdl;  import in.co.irctc.reservation.types.ReservationInfo; import in.co.irctc.reservation.types.Ticket;  public class TrainInfoClient {     private TrainInfo trainInfoPort;      public void doReservation(ReservationInfo in) {         Ticket t = trainInfoPort.reserve(in);         System.out.println("Pnr : " + t.getPnr());     }      public void setTrainInfoPort(TrainInfo trainInfoPort) {         this.trainInfoPort = trainInfoPort;     } }</pre>
<p><b>Configure classes as beans</b></p>	<p>Configure TrainInfo SEI Interface as JAXWS:consumer endpoint</p> <p>Configure TrainInfoClient also as bean and inject TrainInfo consumer endpoint as dependent via setter injection.</p>  <pre>&lt;!-- TrainClient.xml --&gt; &lt;beans&gt;     &lt;!-- TrainClient consumer configuration --&gt;     &lt;bean id="trainClient" class="in.co.irctc.reservation.TrainInfoClient"&gt;         &lt;!-- TrainInfo consumer endpoint configuration --&gt;         &lt;property name="trainInfoPort" ref="trainInfo"/&gt;          &lt;!-- TrainInfo consumer configuration --&gt;     &lt;bean id="trainInfo" class="in.co.irctc.reservation.TrainInfo"&gt;         &lt;!-- TrainInfo consumer endpoint configuration --&gt;         &lt;property name="trainInfoPort" ref="trainInfoPort"/&gt;      &lt;/beans&gt;</pre>
<p><b>Create a Test class and run</b></p>	 <pre>&lt;!-- TCTest.xml --&gt; &lt;beans&gt;     &lt;!-- Application context configuration --&gt;     &lt;import resource="classpath:META-INF/cxf/cxf.xml" /&gt;     &lt;import resource="classpath:META-INF/cxf/cxf-service.xml" /&gt;      &lt;!-- TrainInfoClient consumer configuration --&gt;     &lt;bean id="trainClient" class="in.co.irctc.reservation.wsdl.TrainInfoClient"&gt;         &lt;!-- TrainInfo consumer endpoint configuration --&gt;         &lt;property name="trainInfoPort" ref="trainInfo" /&gt;           &lt;!-- TrainInfo consumer configuration --&gt;     &lt;bean id="trainInfo" class="in.co.irctc.reservation.TrainInfo"&gt;         &lt;!-- TrainInfo consumer endpoint configuration --&gt;         &lt;property name="trainInfoPort" ref="trainInfoPort" /&gt;      &lt;/beans&gt;</pre>

## SOAP VS REST

**REST:**

- It is not a protocol rather architectural style
- Completely stateless
- Provide good caching mechanism over Http protocol (can be used to scale-up when the data is not frequently modified on the resource)
- It is Web Style (SOAP is distributed technology)
- There is no standard definition language to expose the interface of resource to the client. They need to have common understanding of content need to be passed (depends on response content container hypermedia)
- It is suitable for few applications such as Mobile platform or PDA's (as SOAP is eliminated)
- These are easy to integrate with existing Web applications, they don't need any changes in existing architecture
- It doesn't enforce the message format, it can be XML or JSON

**SOAP:**

- SOAP is a XML-based messaging protocol
- It has a specification, but REST has none
- SOAP has specification for stateful implementation
- No Caching support
- It has a standard description language WSDL using which consumer and provider can exchange the data over standard interface
- SOAP requires less plumbing code as transactions, security, addressing etc has been provided.
- SOAP is XML based protocol and will not supports any other formats like JSON



# XML VS JSON

**Abbreviation:**

- XML Stands for Extensible Markup language
- Json stands for Javascript object notation

**Meaning:**

- XML is a markup language that defines set of rules for encoding documents in the format that is both human readable and machine readable

Json is a text-based open standard designed for human-readable data interchange. It is derived from JavaScript scripting language for representing simple data structures and associated arrays, called objects. Json format is often used for serializing and transmitting structured data over the network connection. Primarily to transmit data between server and a web application serving an alternative to XML.

**Type of format:**

- XML - Markup language
- Json - Data interchange

**Extended from:**

- XML - SGML
- Json - JavaScript.

**Developed By:**

- XML - World Wide Web Consortium
- The Json format was originally specified by Douglas Crockford for using it at State Software.

**Data types:**

- XML – Does not provide any notation of data types. One must rely on XML schema for adding type information.
- Json – Provides scalar data types and the ability to express structured data through arrays and objects.

**Support for Arrays:**

- XML – Arrays have to be expressed by conventions, for example through the use of outer placeholder element that models the arrays content as inner elements
- Json – Native array support

**Support for Objects:**

- XML - Objects has to be represented through conventions, often through a mixed use of attributes and elements
- Json - Native Objects support

**Null support:**

- XML - Requires the use of xsi:nil on elements in an XML instance document plus an import of the appropriate namespace
- Json - Natively recognized the null

**Comments:**

- XML – Native support and usually available through API
- Json – Not supported

**Namespaces:**

- XML – Supports namespaces which eliminates the risk of name collisions when combining the documents
- Json – No concept of namespaces. Naming collisions are avoided by nesting objects or using prefix in an object member

**Formatting decisions:**

- XML – Requires a greater effort to decide how to map application types to XML elements and attributes
- Json – Simple provides a much more direct mapping of application data

**Size:**

- XML – Documents tends to be lengthy in size
- Json – Syntax is very terse and yields formatted text where most of the space is consumed

**Parsing in Javascript:**

- XML – Requires XML DOM implementation and additional application code to map text back to Javascript objects
- Json – No additional code is required apart from using eval() function

**Learning curve:**

- XML – Generally tends to use of several technologies in concert: Xpath, XML Schema, XSLT, DOM, XML Namespaces etc.
- Json – Already familiar with developers through Javascript

**Tools:**

- XML – Enjoy a mature set of tools widely available
- Json – Rich tool support - scarce