**WEB-SERVICES**

1. The distributed technologies are corba, RMI, EJB etc uses Binary format , hence the objects are serialized into stream of bytes.
2. Web services uses XML format instead of binary format .Hence in web services the data is exchanged as XML message.
3. The XSD(xml schema) supports data types. Hence validation can be done. In web services the objects are serialized into XML message.
4. The web services technologies such as SOAP ,WSDL , UDDI are represented in XML.
5. Web services makes distributed application easier . It makes implementation easier. It is the implementation of service oriented architecture. (SOA) concept.

Architecture :-

Service Requester 🡨-------🡪 Service Registry 🡨------------🡪 Service provider

1. Service provider creates web services .it is contains some business logic functionality in their organization .
2. Service Registry is central location where the service provider can list its web services .
3. Service Requester can invoke the central registry and use the services.

**SOAP:- simple object access protocol**

Soap is a standard messaging protocol in web services.

Soap messaging are represented in XML with additional features such as

1. Set of rules for sending message
2. Network protocol such as HTTP.
3. A way to represent protocol and application faults.
4. It is a standard format for messaging.
5. W3c vendor of soap.

**Structure of soap**

1)<SOAP :envelope> Root elements of soap message

2)<SOAP :Header>

It contains XML elements that describes security credentials ,digital signatures, Transactions, routing instruction, debugging info.

3)<SOAP :Body>

Contains the actual application data in XML format.

1)XML document fragment

2)procedure call with parameters

3)fault message

**WSDL :- web service description language**

WSDL documents are represented in XML . It is used to provide interface name, method name, parameter types, return type etc in XML format.

It provides message format such as SOAP.

It provides messaging protocol such as HTTP.

Messaging style such as document or RPC.(remote procedure call)

Encoding style such as soap encoding.

End point address URL that client can communicate with a service.

It is well suited for code generation tools, which can read a WSDL document and generate a programmatic interfaces and network stubs.

The WSDL is a schema based document .It contains 7 important elements.

1)<wsdl :definitions>

2)<wsdl:types>

3)<wsdl:message>

4)<wsdl:portTypes>

5)<wsdl:operation>

6)<wsdl:binding>

7)<wsdl:service>

**JAXP :- Java API for XML processing**

1)it is specification from W3C.

2)It is a API from sun.

3)using JAXP API we can process XML documents in two methodologies .1)DOM 2)SAX

DOM :- Dom stands for document object model.

It is tree based model.

At a time total document will be loaded into the memory as a tree structure.

JAXB :- java architecture for XML binding

Jaxb is a java standard that defines how java objects are converted to XML.

Programmer doesnot have to deal with XML directly.

JAXB parser is faster than SAX parser.

JAXB also provides random access like DOM.

It provides compiler to compile XML schema to java class.

JAXB is high level language while JAXP/SAX/DOM are assembly language for XML document management.

**UDDI :- Universal description discovery and integration.**

UDDI is the registry where all the WSDL documents are registered.

It is also developed in XML.

It is also called XML Registry.

WSDL is the document which explains the services information. This WSDL documents has to be placed in some location from where consumer can access them.

SOAP🡨-------XML🡨------CONSUMER 🡨-------------HTTP,FTP,SMTP ---🡪PROVIDER ->WSDL ------->UDDI

**1)SOAP**

**2)REST :- Representational State Transfer.**

Rest is an architectural approach.It is not a protocal.

It uses HTTP protocal.

we build REST services with both XML and JSON.The data transfermation is done either using JSON or XML.

We have a resource to represent the data in either XML,HTML or JSON format.

Restful webservices are platform and language independent.

We have some methods in HTTP which we used in Rest webservices to enter data,save data,update data or delete data.

1)GET :-It is used to read a resource and display the data.

2)PUT:- It is used to update an existing resource.

3)POST:- It is used to create a new resource.

4)DELETE:-It is used to delete the resource.

example:-

GET/users :- It will retrive all the details of all users.

GET/users/{id} :- It will retrive the details of the users whose id is mentioned.

POST/users :- It will create a user.

DELETE/users/{id} :- It will delete the details of the users whose id is mentioned.

PUT/users/{id} :- It will UPDATE the details of the users whose id is mentioned.

404 ---RESOURCE NOT FOUND

200---success

500---server error

401---Unauthorized

webservices :-mobile app bookmy show,amazon,flipcard,telephone bill payment.

paytm :- all services are put together to make customers satisfied.

amazon:- online shopping,utility bils payment.

In this data tranformation will be done by using JSON,text,Html,xml

It can be reused.we can develop webservices using .net,java,php etc.

We can comunicate with the services.

we have service provider,service consumer,service repository.

--------------------------------------------Example-1-------------------------------

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBootExample13Application {

public static void main(String[] args) {

SpringApplication.run(SpringBootExample13Application.class, args);

}

}

-----------------------------------Controller--------------------------------------

package com.example.demo.controller;

import java.util.\*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import com.example.demo.model.Book;

import com.example.demo.service.BookService;

@RestController

public class BookStoreController

{

@Autowired

private BookService bookservice;

@GetMapping("/bookstore/books")

public List<Book> getAllBook()

{

List<Book> allBookList=bookservice.getAllBook();

return allBookList;

}

@GetMapping("/bookstore/books/{bookid}")

public Book getBookById(@PathVariable String bookid)

{

Book bookDetails=bookservice.getBookById(Integer.parseInt(bookid));

return bookDetails;

}

@RequestMapping(method=RequestMethod.POST, value="/bookstore/books")

public void addBook(@RequestBody Book book)

{

bookservice.addBook(book);

}

@RequestMapping(method=RequestMethod.PUT, value="/bookstore/books/{bookid}")

public void editBook(@RequestBody Book book,@PathVariable String bookid)

{

bookservice.editBook(book,Integer.parseInt(bookid));

}

@RequestMapping(method=RequestMethod.DELETE, value="/bookstore/books/{bookid}")

public void deleteBook(@RequestBody Book book,@PathVariable String bookid)

{

bookservice.deleteBook(Integer.parseInt(bookid));

}

}

package com.example.demo.model;

public class Book

{

private long bookid;

private String title;

private double price;

public long getBookid() {

return bookid;

}

public void setBookid(long bookid) {

this.bookid = bookid;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public String getAuthorname() {

return Authorname;

}

public void setAuthorname(String authorname) {

Authorname = authorname;

}

private String Authorname;

public Book(long bookid, String title, double price, String authorname) {

super();

this.bookid = bookid;

this.title = title;

this.price = price;

Authorname = authorname;

}

@Override

public String toString() {

return "Book [bookid=" + bookid + ", title=" + title + ", price=" + price + ", Authorname=" + Authorname + "]";

}

}

package com.example.demo.service;

import java.util.List;

import com.example.demo.model.Book;

public interface IBookService

{

public void addBook(Book book);

public void editBook(Book book,int bookid);

public boolean deleteBook(int bookid);

public Book getBookById(int bookid);

public List<Book> getAllBook();

}

package com.example.demo.service;

import java.util.\*;

import com.example.demo.model.Book;

import org.springframework.stereotype.Component;

@Component

public class BookService implements IBookService

{

private static List<Book> books = new ArrayList<Book>();

static

{

Book book1=new Book(1,"Core Java",300.00, "Shubham");

Book book2=new Book(2,"Adv Java",600.00, "Trupti");

Book book3=new Book(3,"Spring Boot",700.00, "Madhu");

books.add(book1);

books.add(book2);

books.add(book3);

}

public void addBook(Book book)

{

books.add(book);

}

public void editBook(Book book, int bookid)

{

Book record=getBookById(bookid);

books.remove(record);

book.setBookid(bookid);

books.add(book);

}

public boolean deleteBook(int bookid) {

Book record =getBookById(bookid);

books.remove(record);

return Boolean.TRUE;

}

public Book getBookById(int bookid)

{

return books.stream().filter(b ->b.getBookid() == bookid).findFirst().get();

}

public List<Book> getAllBook()

{

return books;

}

}

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jersey</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

properties

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

server.port=8090

**output :-**

http://localhost:8090/bookstore/books(get)

[ {

"bookid": 1,

"title": "Core Java",

"price": 300.0,

"authorname": "Shubham"

},

{

"bookid": 2,

"title": "Adv Java",

"price": 600.0,

"authorname": "Trupti"

},

{

"bookid": 3,

"title": "Spring Boot",

"price": 700.0,

"authorname": "Madhu"

}

]

http://localhost:8090/bookstore/books/1(get)

{

"bookid": 1,

"title": "Core Java",

"price": 300.0,

"authorname": "Shubham"

}

---------------------------------------------POST-------------------------------

body--->raw--->JSON

http://localhost:8090/bookstore/books(post) ----adding new Records

{

"bookid": 4,

"title": "Java Advance",

"price": 700.0,

"authorname": "Trupti"

}

------------------------------PUT----------------------------------------

http://localhost:8090/bookstore/books/5(put) ----modify the data

{

"bookid": 5,

"title": "Java j2ee",

"price": 650.0,

"authorname": "Trupti Patel"

}

-----------------------------DELETE-------------------------------------

http://localhost:8090/bookstore/books/5(delete) ----delete the data

{

"bookid": 5,

"title": "Java j2ee",

"price": 650.0,

"authorname": "Trupti Patel"

}