Lecture 5: Web Services: JAX-WS 2.0
JAX-WS 2.0

- Part of Java EE.
- New in Java SE 6.
- API stack for web services.
- Replaces JAX-RPC.
- New API’s:
  - JAX-WS, SAAJ, Web Service metadata
- New packages:
  - javax.xml.ws, javax.xml.soap, javax.jws
Writing A Web Service

```java
package loanservice;

import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.xml.ws.Endpoint;

@WebService
public class LoanApprover {
    @WebMethod
    public boolean approve(String name) {
        return name.equals("Mike");
    }
}
```
public static void main(String[] args) {
    LoanApprover la = new LoanApprover();
    Endpoint endpoint = Endpoint.publish("http://localhost:8080/loanapprover", la);
}
}
Compile The Service

Create a myservice directory.

From the directory just above loanservice, run Java’s Annotation Processing Tool (APT):

C:\>apt -d myservice loanservice/LoanApprover.java

This populates a directory named myservice. The directory holds the compiled package as well as a new directory (package) called jaxws. The new jaxws package holds classes associated with the parameters to and from each web service method. Use the -s switch to generate the source code.
Publish the Service

From a directory just above myservice:

C:\>java -cp myservice loanservice/LoanApprover

To view the WSDL, visit the service with a browser at http://localhost:8080/loanapprover?wsdl
Generate Stub Code

Make a client directory.


This populates the client subdirectory with .class and .java files.
package client;

class ApproverClient {

    public static void main(String args[]){
        LoanApproverService service = new LoanApproverService();

        LoanApprover approverProxy = service.getLoanApproverPort();

        boolean result = approverProxy.approve("Mike");

        if(result) System.out.println("Approved");
        else System.out.println("Not approved");
    }
}

Compile & Run the Client

C:\>javac -cp . client/ApproverClient.java

C:\>java -cp . client/ApproverClient
Approved

Demo files under :
m m6/www/95-843/JDK6_WebServices/Demo
ServerCounter is a Singleton

@WebService
public class ServerCounter {
    int ctr = 0;
    public int getCtr() {
        ctr++;  // What happens?
        return ctr;
    }
}

What happens?
A single object holds the count and every client shares it. Each visit generates a new updated count.