Introduction to Web Services

Bina Ramamurthy

Literature Surveyed

- IBM's alphaworks site:
http://www-106.ibm.com/developerworks/webservices/

http://www-3.ibm.com/software/solutions/webservices/pdf/WSCA.pdf

Web Services

- *Web Services* is a technology that allows for applications to communicate with each other in a standard format.
- A *Web Service* exposes an interface that can be accessed through XML messaging.
- A Web service uses XML based protocol to describe an operation or the data exchange with another web service. Ex: SOAP
- A group of web services collaborating accomplish the tasks of an application. The architecture of such an application is called Service-Oriented Architecture (SOA).

Web Services Suite of Protocols

- A suite of protocols define the Web Services Technology.
- These are used to describe, publish, discover, deliver and interact with services.
- The information about the protocols is from IBM's developerworks.

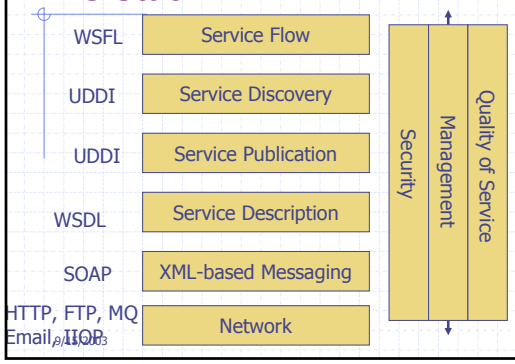
WS Suite of Protocols

- Messaging protocol Simple Object Access Protocol (SOAP) encodes messages so that they can be delivered over the transport protocols HTTP, SMTP or IIOP.
- Web Services Definition Language (WSDL) is used to specify the service details such as name, methods and their parameters, and the location of the service. This facilitates the registering and discovery of the service.
- For services to locate each other, the Universal Description, Discovery and Integration (UDDI) protocol defines a registry and associated protocols for locating and accessing services.

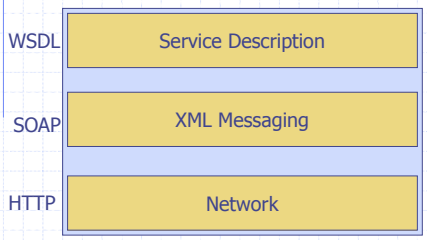
WS Suite of Protocols (contd.)

- The WS-Transaction and WS-Coordination protocols work together to handle distributed transactions.
- The Business Process Execution Language for Web Services (BPEL4WS) defines workflow operations.
- WS-Security is a family of protocols that cover authentication, authorization, federated security, secure communications, and delivery.
- WS-Policy is another group of protocols that define the policy rules behind how and when Web services can interact with each other.
- WS-Trust defines how trust models work between different services.
- These protocols are for e-business. Are there any available for e-science?

WS Stack



WS Interoperability Infrastructure

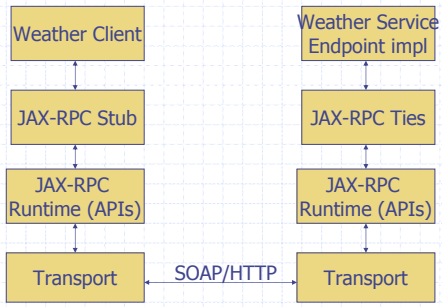


Do you see any platform or language dependencies here?

JAX-RPC

- **JAX-RPC: Java API for XML-based Remote Procedure Call (RPC).**
- **An API for building Web Services and Web Services clients.**
- **Some important concepts in JAX-RPC are:**
  - **Type-mapping system (WSDL to Java)**
  - **Service endpoint**
  - **Exception handling**
  - **Service endpoint context**
  - **Message handlers**
  - **Service clients and service context**
  - **SOAP with attachments**
  - **Runtime services**
  - **JAX-RPC client invocation models**

Application Architecture



Approaches to Web Service Implementation

- Top down: Start with WSDL and map onto Java
- Bottom up: Start with Java and end up all the supporting classes needed.
- We used the second approach for our RMI example.

WS Development Lifecycle

- Build:
  - Definition of service interface
  - Definition of service implementation
    - New services
    - Existing application into WS
    - Composing a WS out of other WS and applications
  - Source compiled and Stubs and Ties are generated.
- Deploy:
  - Publication of the service interface and service implementation to service registry or service requestor.
  - Deployment of executables in an execution environment.

WS Development Lifecycle (contd.)

- Run: A WS is available for invocation. Requestor can perform find and bind operation.
- Manage: on going management and administration for security, availability, performance, QoS, and business processes.

A Simple Example from Sun Microsystem

- HelloWorld distributed application:
- Files of interest:
  - **HelloIF.java: service definition interface**
  - **HelloImpl.java: Service definition implmentation.**
  - **HelloClient.java: remote client to invoke the service.**
  - **config-interface.xml: configuration file used by wscompile**
  - **jaxrpc-ri.xml: a configuration file used by wsdeploy**
  - **web.xml: a deployment descriptor for the web component that dispatches to the service.**
  - **build.xml for running the various steps such as compile, wscompile, deploy etc. Used by the ant tool.**
  - **build.properties: contains the details of varuious context roots or paths.**

Building and Deploying the Service

- Code the service definition interface and implementation class.
- Compile the service definition code written above.
- Package the code in a WAR (web archive) file.
- Generate the ties and WSDL files.
- Deploy the service.

Coding the interface and implementation classes

- The interface extends java.rmi.Remote interface.
- No constant declarations allowed.
- Methods must throw java.rmi.RemoteException
- Method parameters and return types must be supported by JAX-RPC types.

Compiling and Packaging

- To compile:
  - ant compile-server
- To pacakge:
  - ant setup-web-inf
  - ant package
- These two commands will generate and place the executables in appropriate directories. (Details will be given to you later in another handout).

Generating Ties and WSDL file and deploy the service

- To generate Ties and WSDL:
  - ant process-war
  - Will invoke *wsdeploy* to generate the tie classes and the WDSL file MyHello.wsdl
- To deploy the service:
  - ant deploy
- To verify deployment:
  - http://localhost:8080/hello-jaxrpc/hello
  - The details of the web service will be displayed.
- To undeploy:
  - ant undeploy

Building and Running the client

- Generate the stubs.
- Code the client.
- Compile the client code.
- Package the client classes into a JAR file.
- Run the client.

Client steps

- To generate stubs:
  - ant generate-stubs
  - This will call *wscompile* to generate the stubs.
- Coding the client: is a stand alone program. It calls the service through the generated stub which acts as a proxy for the remote service.

Clients steps (contd.)

- Compile the client code:
  - ant compile-client
- Package the client:
  - ant jar-client
- Running the client:
  - Ant run

Iterative Development

- Test the application.
- Edit the source files.
- Execute *ant build* to create and deploy war files.
- Execute *ant redeploy* to undeploy and deploy the service.
- Execute *ant build-static* to create jar files with static stubs.
- Execute *ant run* to run the client.

Other features

- Read about the types supported by JAX-RPC
- An advanced feature of interest is the dynamic proxy.
- Read about the directory structure and paths.

Reading Material

- Introduction to Web Services Ch.1.
- Building Web Services with JAX-RPC Ch. 11.
- Ant build tool details.
- XML, XML Schema and SOAP1.1.