Web Services Overview

Marlon Pierce
Indiana University
mpierce@cs.indiana.edu

A Note on XML

- Bryan Carpenter put together a comprehensive set of slides on XML.
  - http://www.grid2004.org/spring2004/
- Web Services make extensive use of XML, so Bryan's slides provide an excellent review.

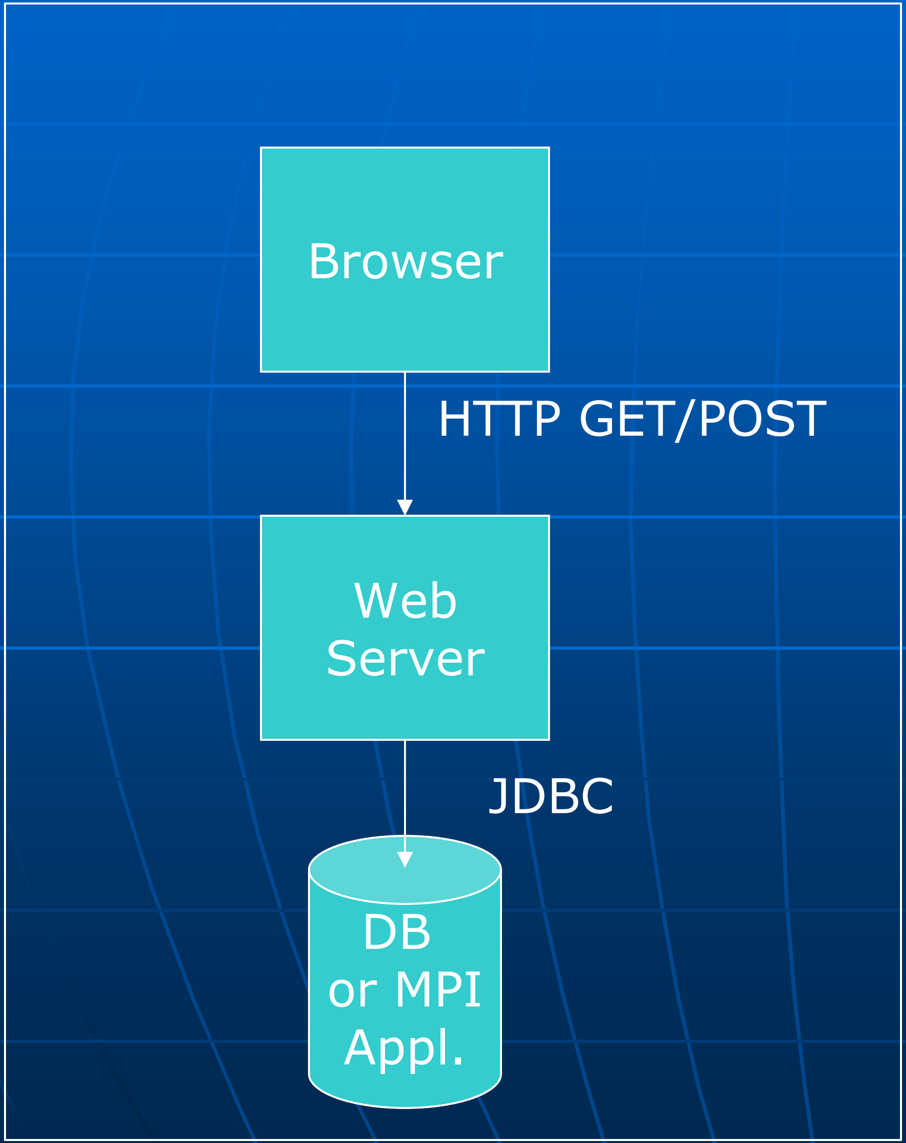
What Are Web Services?

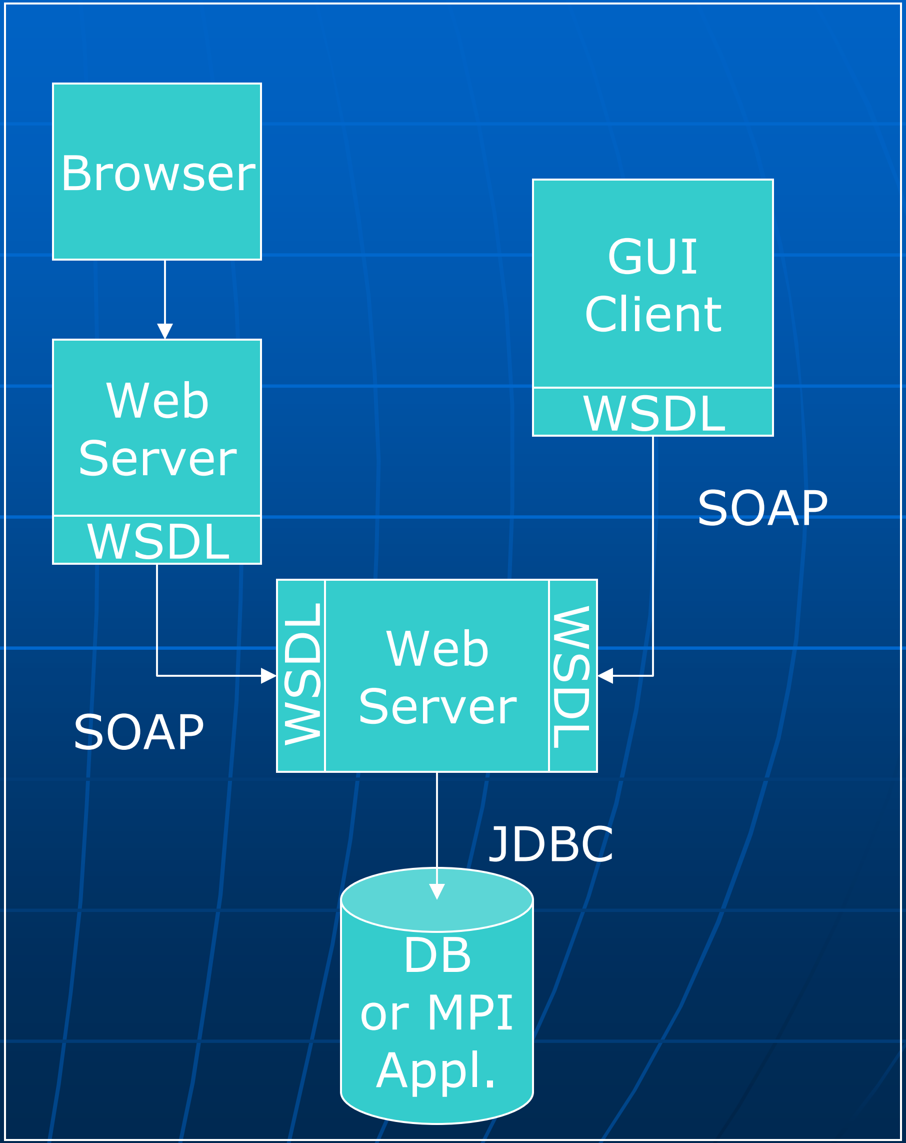
- Web services framework is an XML-based distributed services system.
  - SOAP, WSDL, UDDI
  - WS-Interoperability
  - Intended to support machine-to-machine interactions over the network using messages.
- Basic ideas is to build a platform and programming language-independent distributed invocation system out of existing Web standards.
  - Most standards defined by W3C, Oasis (IP considerations)
  - Interoperability really works, as long as you can map XML message to a programming language type, structure, class, etc.
    - We regularly use Java-C++ and Java-Perl communication
- Very loosely defined, when compared to CORBA, etc.
- Inherit both good and bad of the web
  - Scalable, simple, distributed
  - But no centralized management, not high performance, must be tolerant of failures.

Web Services Compared to MPI

- WSDL is a API definition language
  - Your programs have been using the MPI API
  - Your codes on the Grid Farm machines have been using the LAM-MPI implementation of MPI.
    - Prof. Andrew Lumsdaine, Indiana U and the Open Systems Lab
- SOAP is an envelope for transferring messages.
  - You can build messaging systems (“MOMs”) with SOAP.
- For the most part, WS and MPI apply to very different domains.
  - Web Services are loosely coupled
    - Use (typically) HTTP to carry messages.
  - No shared memory
  - Millisecond (or longer) message communication speeds instead of microsecond.

Basic Architectures: Servlets/CGI and Web Services





Explanation of Previous Slide

- The diagram on the left represents a standard web application.
  - Browsers converse with web servers using HTTP GET/POST methods.
  - Servlets or CGI scripts process the parameters and take action, like connect to a DB.
  - Examples: Google, Amazon
- On the right, we have a Web services system.
  - Interactions may be either through the browser or through a desktop client (Java Swing, Python, Windows, etc.)
  - Examples: Google, Amazon

Some Terminology

- The diagram on the left is called a client/server system.
- The diagram on the right is called a multi-tiered architecture.
- SOAP: Simple Object Access Protocol
  - No longer an abbreviation in SOAP 1.2
  - XML Message format between client and service.
- WSDL: Web Service Description Language.
  - Describes how the service is to be used
  - Compare (for example) to Java Interface.
  - Guideline for constructing SOAP messages.
  - WSDL is an XML language for writing Application Programmer Interfaces (APIs).

Amazon and Google Experiment with Web Services

- Both Google and Amazon have conducted open experiments with Web services.
- Why? To allow partners to develop custom user interfaces and applications that work Google and Amazon data and services.
- You can download their APIs and try them.
  - http://www.google.com/apis/
  - http://www.amazon.com/webservices

More Examples of Web Services

- Geographical Information Systems are perfect candidates for WS
  - The Open Geospatial Consortium defines several relevant standards
    - Geographic Markup Language (GML) exchanges info.
    - Web Feature Service works with abstract GML feature data.
    - Web Map Service creates maps (images)
- XMethods
  - Lots and lots of contributed examples, live demos
  - Try them
    - http://www.xmethods.com/

Web Service Architectures

- The following examples illustrate how Web services interact with clients.
- For us, a client is typically a JSP, servlet, or portlet that a user accesses through browser.
- You can also build other clients
  - Web service interoperability means that clients and services can be in different programming languages (C/C++, python, java, etc).

Before Going On…

- In the next several slides we'll go into the details of WSDL and SOAP.
- But in practice, you don't need to work directly with either.
    - Most tools that I'm familiar with generate the WSDL for you from your class.
    - SOAP messages are constructed by classes.
    - Generated client stubs will even hide SOAP classes behind a local "façade" that looks like a local class but actually constructs SOAP calls to the remote server.

Developing Web Services

Using Apache Axis to develop Java implementations of Web services.

Web Service Development Tools

- Web service toolkits exist for various programming languages:
  - C++, Python, Perl, various Microsoft .NET kits.
- We’ll concentrate on building Java Web services with Apache Axis.
- Language and implementation interoperability is addressed through WS-I.
  - http://www.ws-i.org/

Apache Axis Overview

- Apache Axis is a toolkit for converting Java applications into Web services.
- Axis service deployment tools allow you to publish your service in a particular application server (Tomcat).
- Axis client tools allow you to convert WSDL into client stubs.
- Axis runtime tools accept incoming SOAP requests and redirect them to the appropriate service.

Developing and Deploying a Service

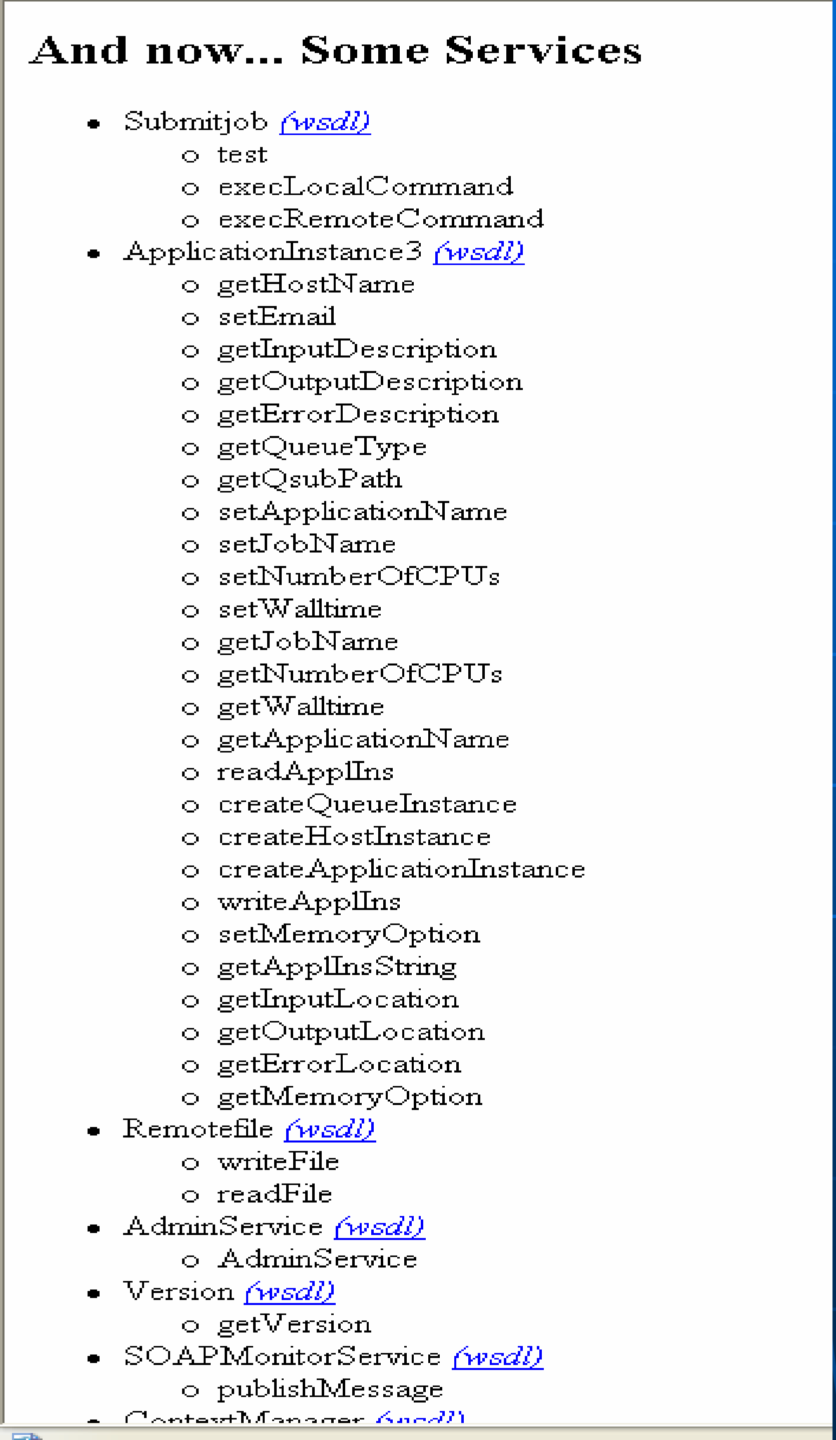
- Download and install Tomcat and Axis.
- Write a Java implementation
  - Services are just Java programs
  - Compile it into Tomcat's classpath.
- Write a deployment descriptor (WSDD) for your service.
  - Will be used by Axis runtime to direct SOAP calls.
- Use Axis's AdminClient tool to install your WSDD file.
  - The tells the axis servlet to load your class and direct SOAP requests to it.
- That's it.
  - Axis will automatically generate the WSDL for your service.

Sample WSDD

```
<deployment name="Submitjob"
      xmlns="http://xml.apache.org/axis/wsdd/"
      xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
      <service name="Submitjob" provider="java:RPC">
            <parameter name="scope" value="request"/>
            <parameter name="className"
                        value="WebFlowSoap.SJwsImp"/>
            <parameter name="allowedMethods"
                        value="execLocalCommand"/>
      </service>
</deployment>
```

Explanation

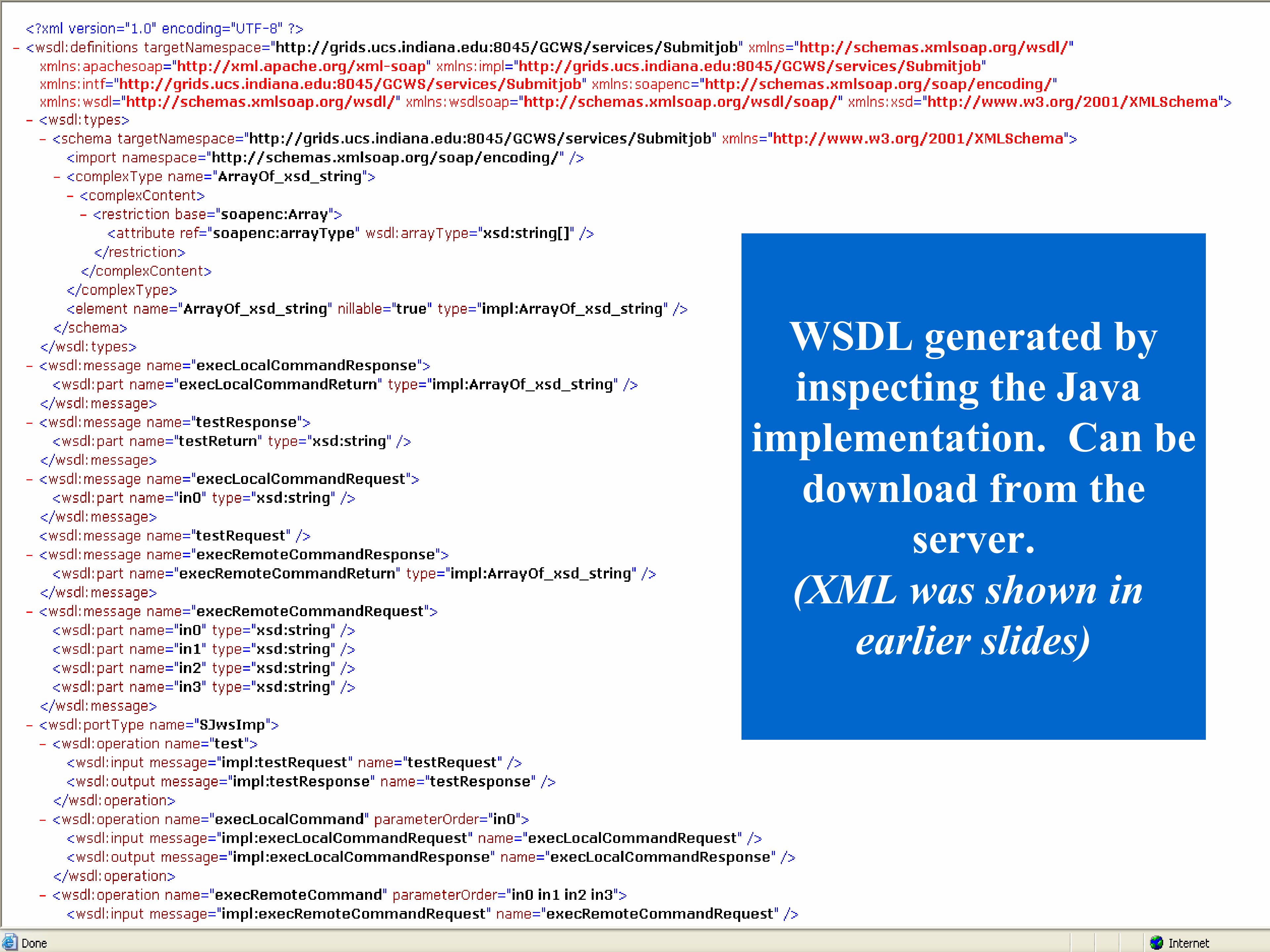
- Use Axis’s command-line AdminClient tool to deploy this to the server.
- Axis will create a service called
  - http://your.server/services/SubmitJob
- WSDL for service is available from
  - http://your.server/services/SubmitJob?wsdl
- A list of all services is available from
  - http://your.server/services



And now... Some Services

- Submitjob (wsdl)
  - test
  - execLocalCommand
  - execRemoteCommand
- ApplicationInstance3 (wsdl)
  - getHostName
  - setEmail
  - getInputDescription
  - getOutputDescription
  - getErrorDescription
  - getQueueType
  - getQsubPath
  - setApplicationName
  - setJobName
  - setNumberOfCPUs
  - setWalltime
  - getJobName
  - getNumberOfCPUs
  - getWalltime
  - getApplicationName
  - readApplIns
  - createQueueInstance
  - createHostInstance
  - createApplicationInstance
  - writeApplIns
  - setMemoryOption
  - getApplInsString
  - getInputLocation
  - getOutputLocation
  - getErrorLocation
  - getMemoryOption
- Remotefile (wsdl)
  - writeFile
  - readFile
- AdminService (wsdl)
  - AdminService
- Version (wsdl)
  - getVersion
- SOAPMonitorService (wsdl)
  - publishMessage
- ContextManager (wsdl)

Check your Tomcat Server for a list of deployed Services:
http://localhost:8080/axis/services



```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="http://grids.ucs.indiana.edu:8045/GCWS/services/Submitjob" xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://grids.ucs.indiana.edu:8045/GCWS/services/Submitjob"
   xmlns:intf="http://grids.ucs.indiana.edu:8045/GCWS/services/Submitjob" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <wsdl:types>
    - <schema targetNamespace="http://grids.ucs.indiana.edu:8045/GCWS/services/Submitjob" xmlns="http://www.w3.org/2001/XMLSchema">
        <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      - <complexType name="ArrayOf_xsd_string">
        - <complexContent>
          - <restriction base="soapenc:Array">
              <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]" />
            </restriction>
          </complexContent>
        </complexType>
        <element name="ArrayOf_xsd_string" nillable="true" type="impl:ArrayOf_xsd_string" />
      </schema>
    </wsdl:types>
  - <wsdl:message name="execLocalCommandResponse">
      <wsdl:part name="execLocalCommandReturn" type="impl:ArrayOf_xsd_string" />
    </wsdl:message>
  - <wsdl:message name="testResponse">
      <wsdl:part name="testReturn" type="xsd:string" />
    </wsdl:message>
  - <wsdl:message name="execLocalCommandRequest">
      <wsdl:part name="in0" type="xsd:string" />
    </wsdl:message>
    <wsdl:message name="testRequest" />
  - <wsdl:message name="execRemoteCommandResponse">
      <wsdl:part name="execRemoteCommandReturn" type="impl:ArrayOf_xsd_string" />
    </wsdl:message>
  - <wsdl:message name="execRemoteCommandRequest">
      <wsdl:part name="in0" type="xsd:string" />
      <wsdl:part name="in1" type="xsd:string" />
      <wsdl:part name="in2" type="xsd:string" />
      <wsdl:part name="in3" type="xsd:string" />
    </wsdl:message>
  - <wsdl:portType name="SJwsImp">
    - <wsdl:operation name="test">
        <wsdl:input message="impl:testRequest" name="testRequest" />
        <wsdl:output message="impl:testResponse" name="testResponse" />
      </wsdl:operation>
    - <wsdl:operation name="execLocalCommand" parameterOrder="in0">
        <wsdl:input message="impl:execLocalCommandRequest" name="execLocalCommandRequest" />
        <wsdl:output message="impl:execLocalCommandResponse" name="execLocalCommandResponse" />
      </wsdl:operation>
    - <wsdl:operation name="execRemoteCommand" parameterOrder="in0 in1 in2 in3">
        <wsdl:input message="impl:execRemoteCommandRequest" name="execRemoteCommandRequest" />
```

WSDL generated by inspecting the Java implementation. Can be download from the server.

(XML was shown in earlier slides)

Building a Client with Axis

- Obtain the WSDL file.
- Generate client stubs
  - Stubs look like local objects but really convert method invocations into SOAP calls.
- Write a client application with the stubs
  - Can be a Java GUI, a JSP page, etc.
- Compile everything and run.

Sample Java Client Code

```
/**Create SubmitJob client object and point to the
   service you want to use */
SubmiJob sjws = new
   SubmitJobServiceLocator().getSubmitjob(new

   URL(http://your.server/services/SubmitJob));
/** Invoke the method as if local. */
String[] messages =
        sjws.execLocalCommand(command);
```

Two Notes On Client Stubs

- Axis stubs convert method calls into SOAP requests but WSDL does not require the use of SOAP.
  - Web Service Invocation Framework (WSIF) from IBM allows flexibility of protocols. (Alek Slominski, IU)
- Client stubs introduce versioning problems.
  - We are developing dynamic (stubless) clients that construct SOAP messages by inspecting WSDL at runtime.

Some Web Service URLs

- Apache Axis (Java and C++)
  - http://xml.apache.org/axis/
- NaradaBrokering
  - Java support for Reliability, Eventing, etc.
- WS/XSUL from Indiana University Extreme Labs
  - http://www.extreme.indiana.edu/xgws/xsul/index.html
- gSOAP: C++ SOAP toolkit
  - http://www.cs.fsu.edu/~engelen/soap.html
- Python Web Services:
  - http://pywebsvcs.sourceforge.net/
- Perl:
  - http://www.soaplite.com/