XML and Web Services for Astronomers

Roy Williams
California Institute of Technology
roy@caltech.edu

Robert J. Brunner
University of Illinois
bigdog@uiuc.edu

ASASS 2002, 13 October 2002, Baltimore

Web Services for Astronomers

- What are Web Services
- Web Service Architecture
- Building Web Services
- The Future of Web Services

What are Web Services?

- Web (From Dictionary.com)
  1. A latticed or woven structure
  2. Something intricately contrived, especially something that ensnares or entangles.
  3. A complex, interconnected structure or arrangement

Shorthand for the World Wide Web

What are Web Services?

- Service (From Dictionary.com)
  1. The performance of work or duties for a superior or as a servant
  2. An act or a variety of work done for others, especially for pay
  3. Assistance; help

Slang terms not suitable for print.

What are Web Services?

- Web Service
  - Distributed Computing Model
  - Self-Contained Modular Applications
  - Platform Independent
  - Language Independent

Or

  - An unpaid act of performing intricately contrived work for others that ensnares all?

Hello World

```
public class HelloWorld {
    public java.lang.String getMessage() {
        return "Hello World!" ;
    }

    public static void main(String[] args) {
        HelloWorld hw = new HelloWorld() ;
        System.out.print(hw.getMessage()) ;
    }
}
```

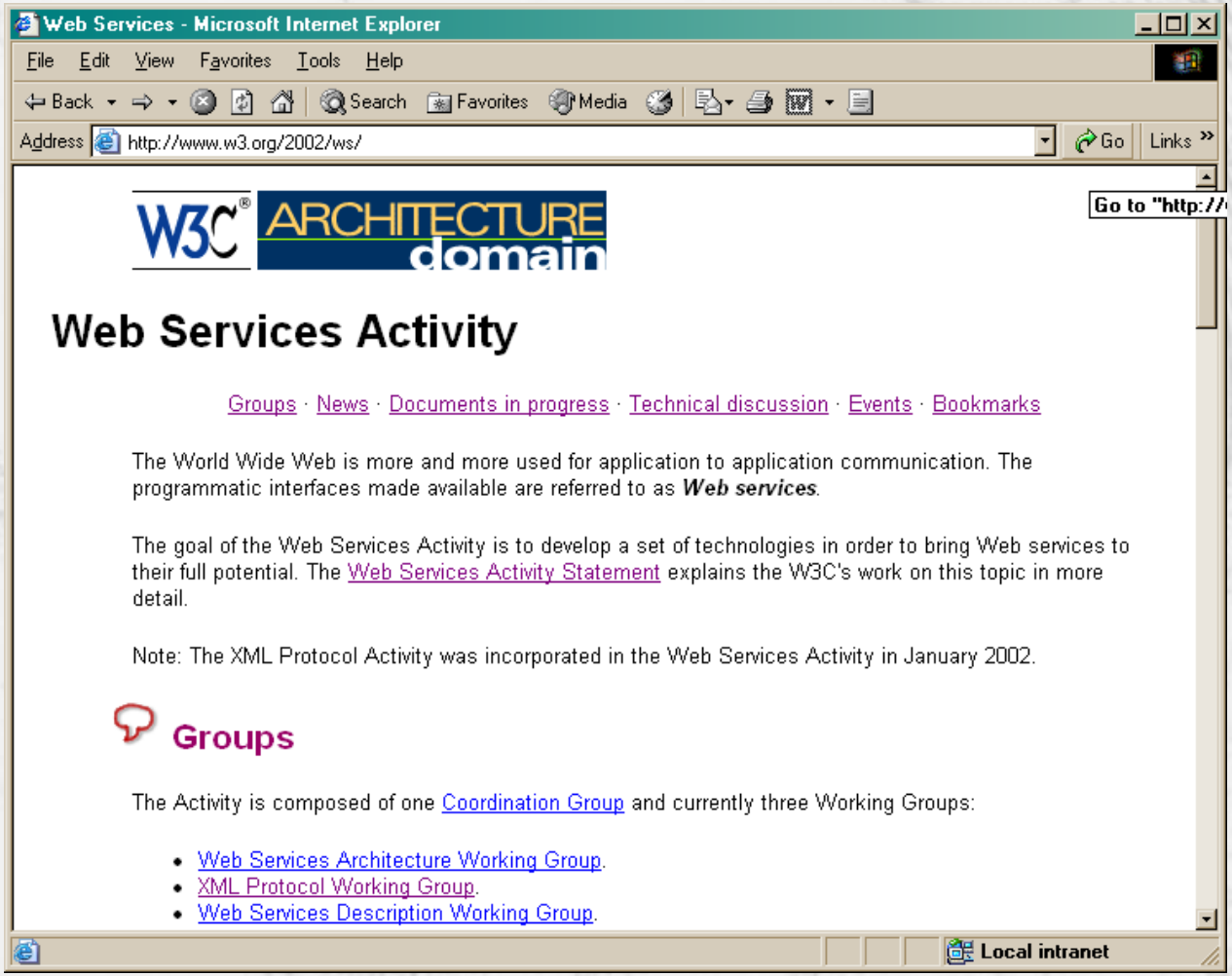
What are Web Services?

A Service that is accessed via the Web!

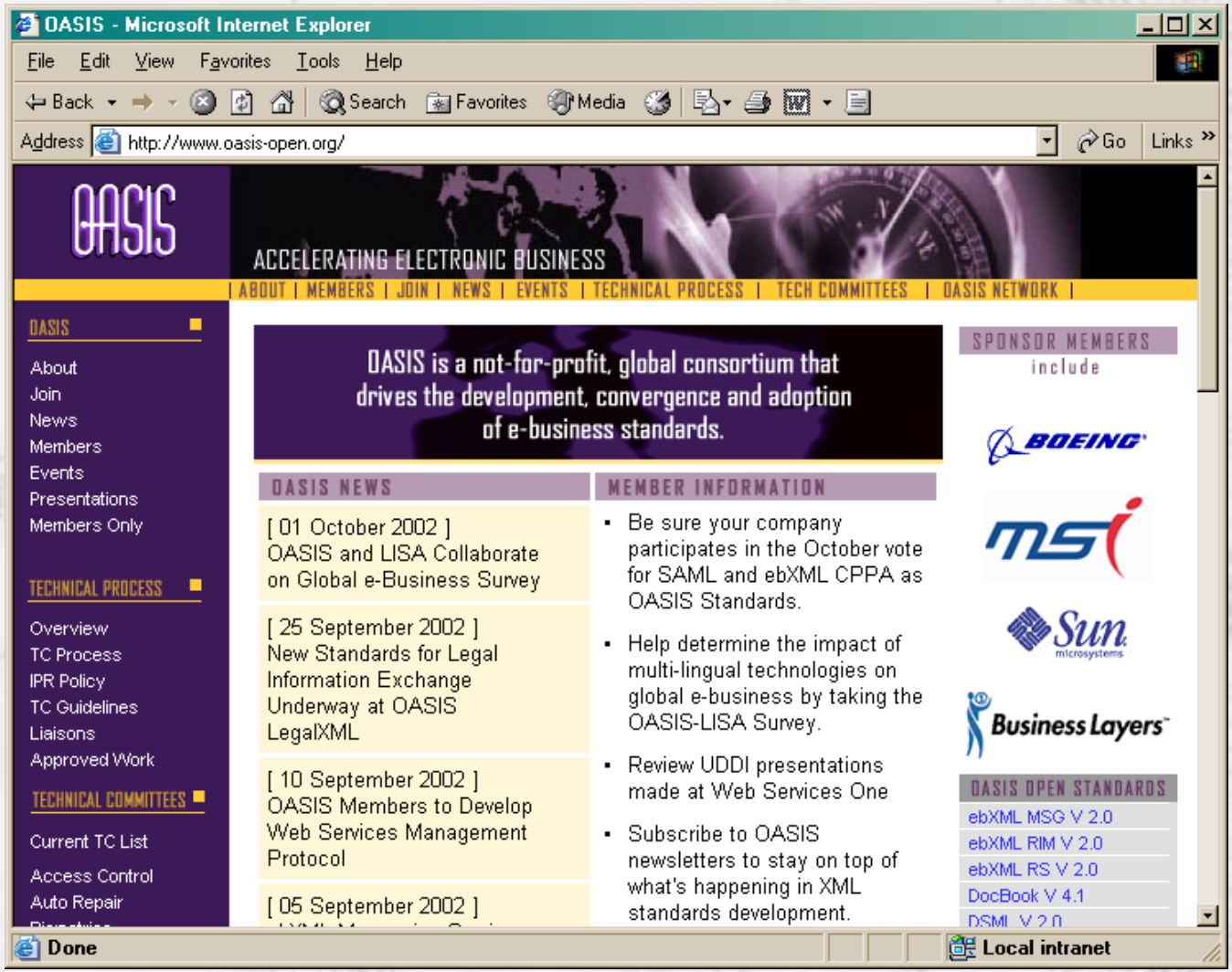
Who is in Control?

- W3C (www.w3c.org)
  - WSDL
  - SOAP/XML Protocol
  - Web Service Activity
- Oasis (www.oasis-open.org)
  - ebXML
  - UDDI
- WS-I (www.ws-i.org)

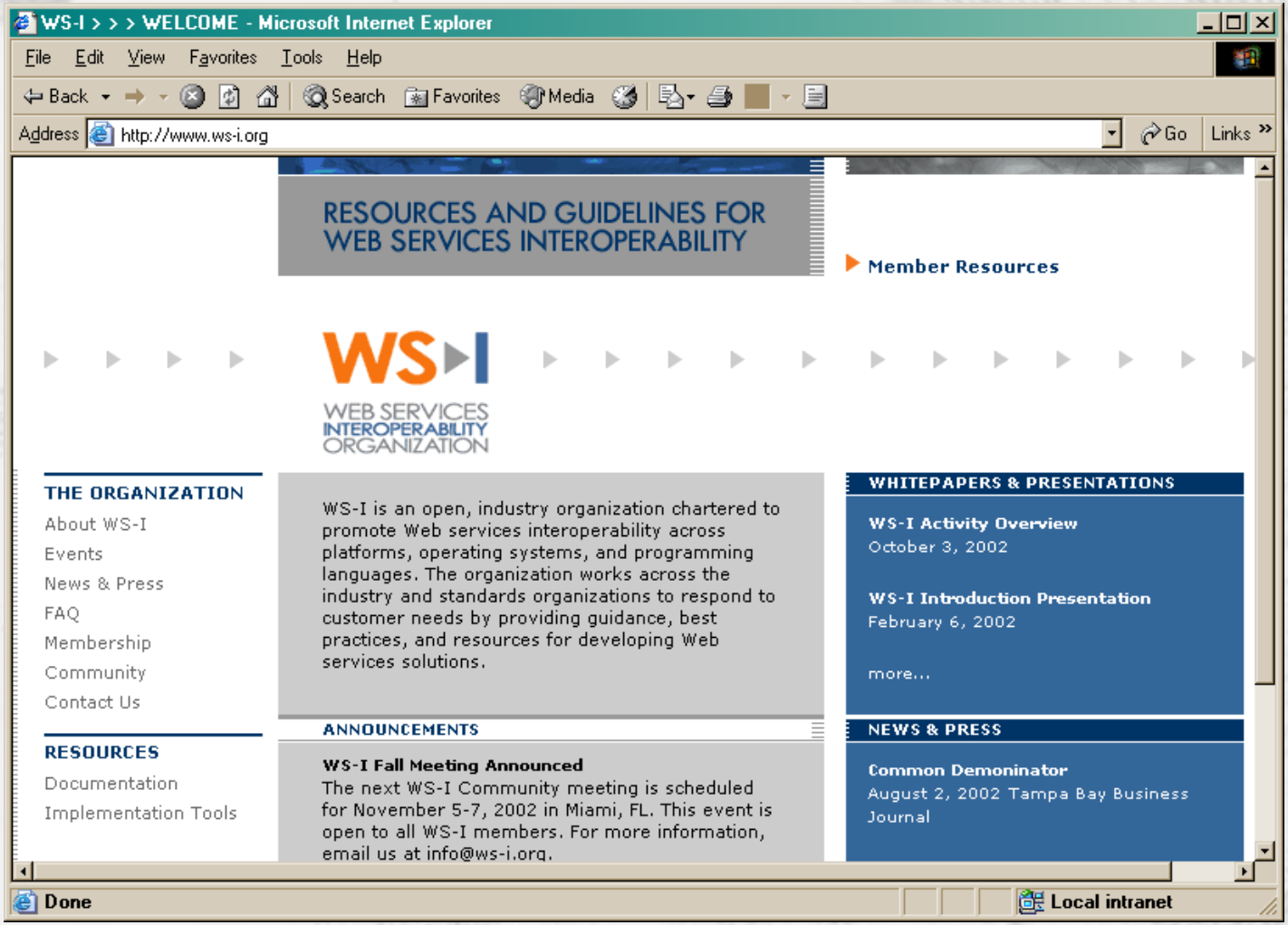
W3C Web Services Activity



OASIS


OASIS - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.oasis-open.org/
Go Links
OASIS
ACCELERATING ELECTRONIC BUSINESS
ABOUT | MEMBERS | JOIN | NEWS | EVENTS | TECHNICAL PROCESS | TECH COMMITTEES | OASIS NETWORK
OASIS
About
Join
News
Members
Events
Presentations
Members Only
TECHNICAL PROCESS
Overview
TC Process
IPR Policy
TC Guidelines
Liaisons
Approved Work
TECHNICAL COMMITTEES
Current TC List
Access Control
Auto Repair
OASIS is a not-for-profit, global consortium that drives the development, convergence and adoption of e-business standards.
OASIS NEWS
[01 October 2002]
OASIS and LISA Collaborate on Global e-Business Survey
[25 September 2002]
New Standards for Legal Information Exchange Underway at OASIS LegalXML
[10 September 2002]
OASIS Members to Develop Web Services Management Protocol
[05 September 2002]
MEMBER INFORMATION
Be sure your company participates in the October vote for SAML and ebXML CPPA as OASIS Standards.
Help determine the impact of multi-lingual technologies on global e-business by taking the OASIS-LISA Survey.
Review UDDI presentations made at Web Services One
Subscribe to OASIS newsletters to stay on top of what's happening in XML standards development.
SPONSOR MEMBERS include
BOEING
Sun microsystems
Business Layers
OASIS OPEN STANDARDS
ebXML MSG V 2.0
ebXML RIM V 2.0
ebXML RS V 2.0
DocBook V 4.1
Done
Local intranet

WS-I


WS-I > > > WELCOME - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.ws-i.org
Go Links
RESOURCES AND GUIDELINES FOR WEB SERVICES INTEROPERABILITY
Member Resources
WS-I
WEB SERVICES INTEROPERABILITY ORGANIZATION
THE ORGANIZATION
About WS-I
Events
News & Press
FAQ
Membership
Community
Contact Us
RESOURCES
Documentation
Implementation Tools
WS-I is an open, industry organization chartered to promote Web services interoperability across platforms, operating systems, and programming languages. The organization works across the industry and standards organizations to respond to customer needs by providing guidance, best practices, and resources for developing Web services solutions.
ANNOUNCEMENTS
WS-I Fall Meeting Announced
The next WS-I Community meeting is scheduled for November 5-7, 2002 in Miami, FL. This event is open to all WS-I members. For more information, email us at info@ws-i.org.
WHITEPAPERS & PRESENTATIONS
WS-I Activity Overview
October 3, 2002
WS-I Introduction Presentation
February 6, 2002
more...
NEWS & PRESS
Common Demoninator
August 2, 2002 Tampa Bay Business Journal
Done
Local intranet

How is this different?

- RPC Model Exists!
  - CORBA
  - COM/DCOM
  - RMI
  - …
- Web Services use XML!!!!!

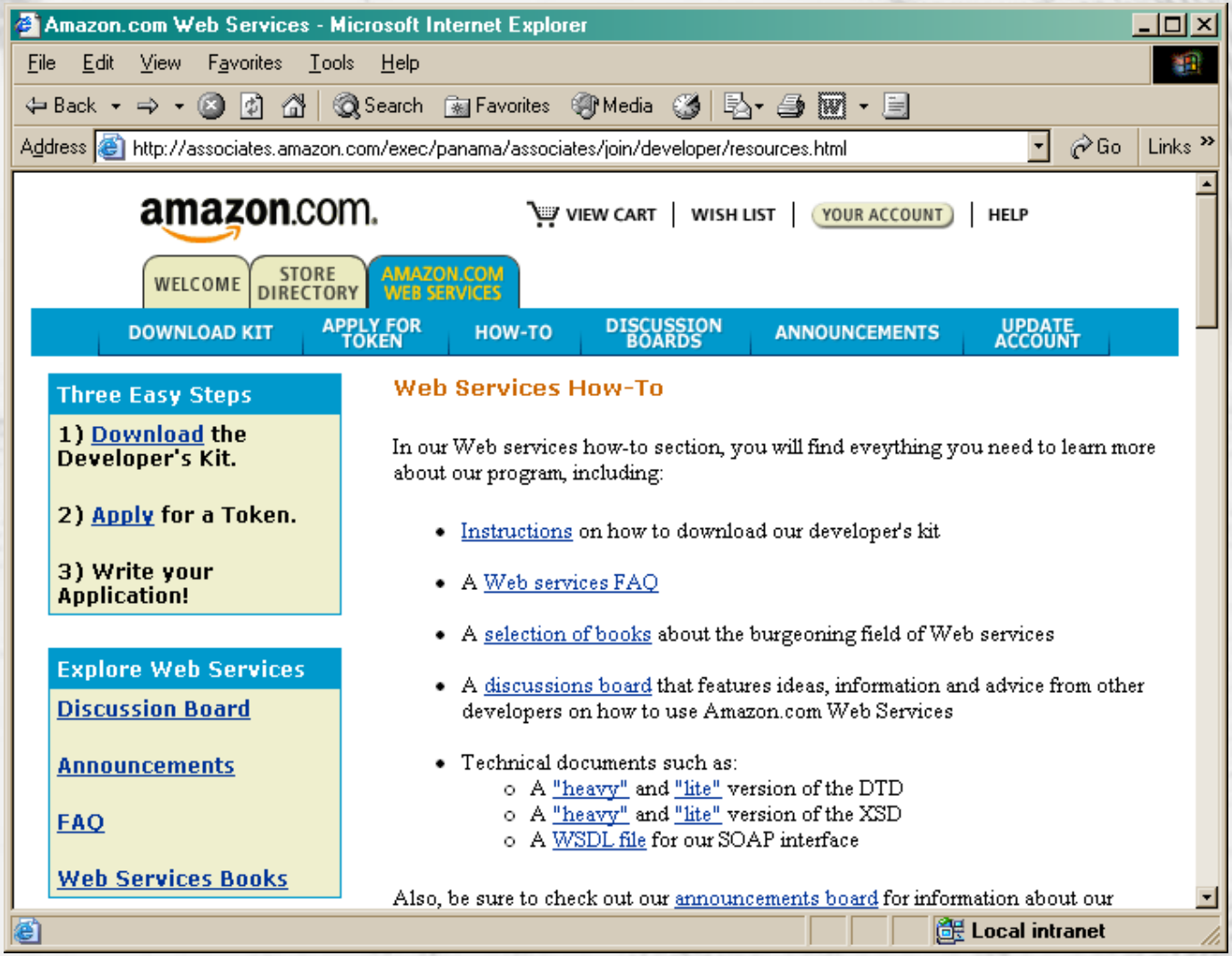
Practical Examples

- Business to Business
  - Inventory Records
  - Bill of Laden
  - Purchase Orders
- Business to Consumer
  - Financial Data
  - Spelling/Searching
  - Product Listings
  - Airline Reservations

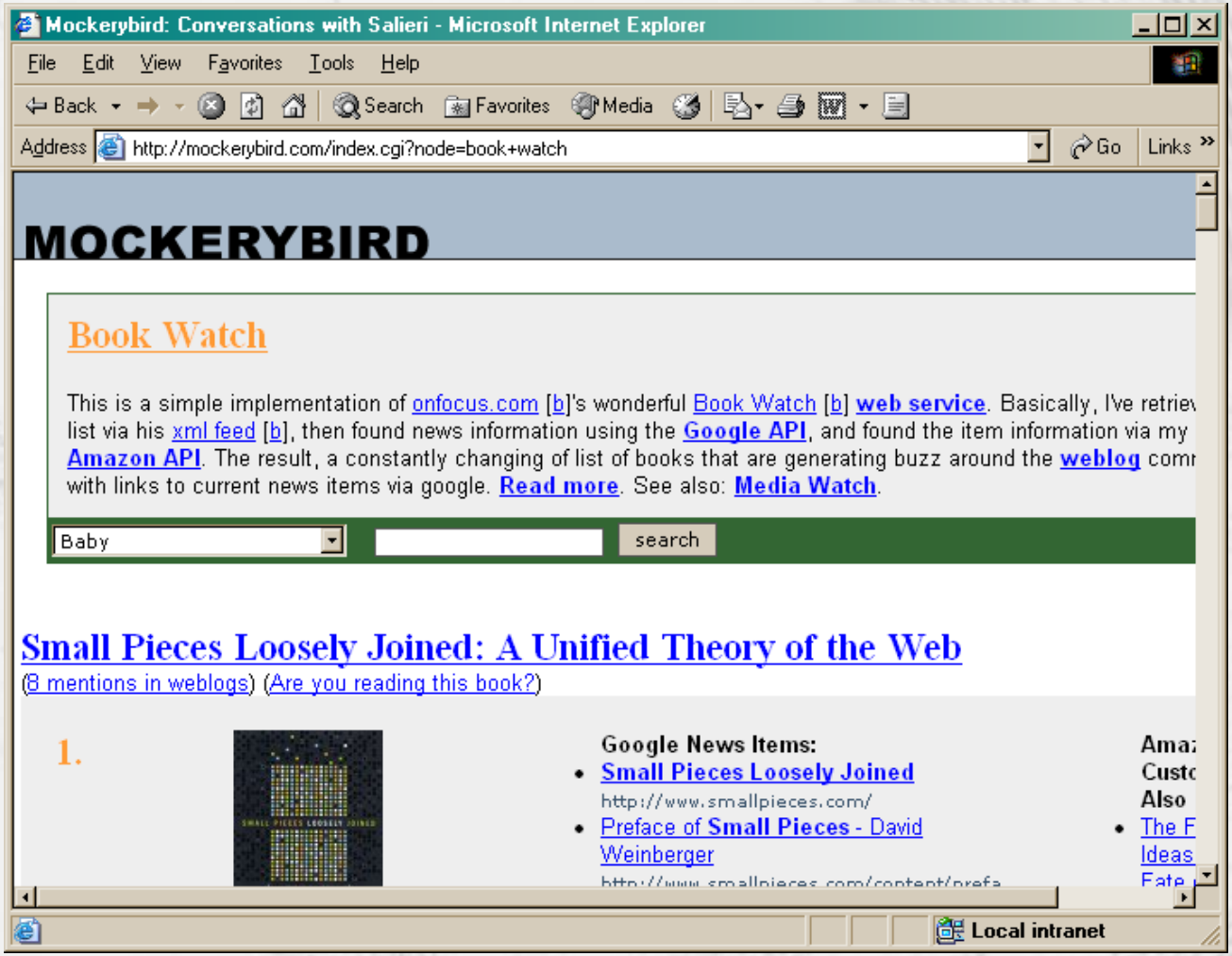
Google


Google Web APIs - Home - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.google.com/apis/
Go Links
Google
Google Web APIs (beta)
Home
All About Google
Google Web APIs
Overview
Download
Create Account
Getting Help
API Terms
FAQs
Find on this site:
Search
Develop Your Own Applications Using Google
With the Google Web APIs service, software developers can query more than 2 billion web documents directly from their own computer programs. Google uses the SOAP and WSDL standards so a developer can program in his or her favorite environment - such as Java, Perl, or Visual Studio .NET.
To start writing programs using Google Web APIs:
1
Download the developer's kit
The Google Web APIs developer's kit provides documentation and example code for using the Google Web APIs service. The download includes Java and .NET programming examples and a WSDL
With Google Web APIs, your computer can do the searching for you.
Server and developer's kit updates
Example Visual Basic .NET client
Better support for multilingual queries
Java client supports HTTP proxies
Done
Local intranet

Amazon


Amazon.com Web Services - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://associates.amazon.com/exec/panama/associates/join/developer/resources.html
Go Links
amazon.com.
VIEW CART | WISH LIST | YOUR ACCOUNT | HELP
WELCOME
STORE DIRECTORY
AMAZON.COM WEB SERVICES
DOWNLOAD KIT
APPLY FOR TOKEN
HOW-TO
DISCUSSION BOARDS
ANNOUNCEMENTS
UPDATE ACCOUNT
Three Easy Steps
1) Download the Developer's Kit.
2) Apply for a Token.
3) Write your Application!
Explore Web Services
Discussion Board
Announcements
FAQ
Web Services Books
Web Services How-To
In our Web services how-to section, you will find eveything you need to learn more about our program, including:
Instructions on how to download our developer's kit
A Web services FAQ
A selection of books about the burgeoning field of Web services
A discussions board that features ideas, information and advice from other developers on how to use Amazon.com Web Services
Technical documents such as:
A "heavy" and "lite" version of the DTD
A "heavy" and "lite" version of the XSD
A WSDL file for our SOAP interface
Also, be sure to check out our announcements board for information about our
Local intranet

Multiple Invocations



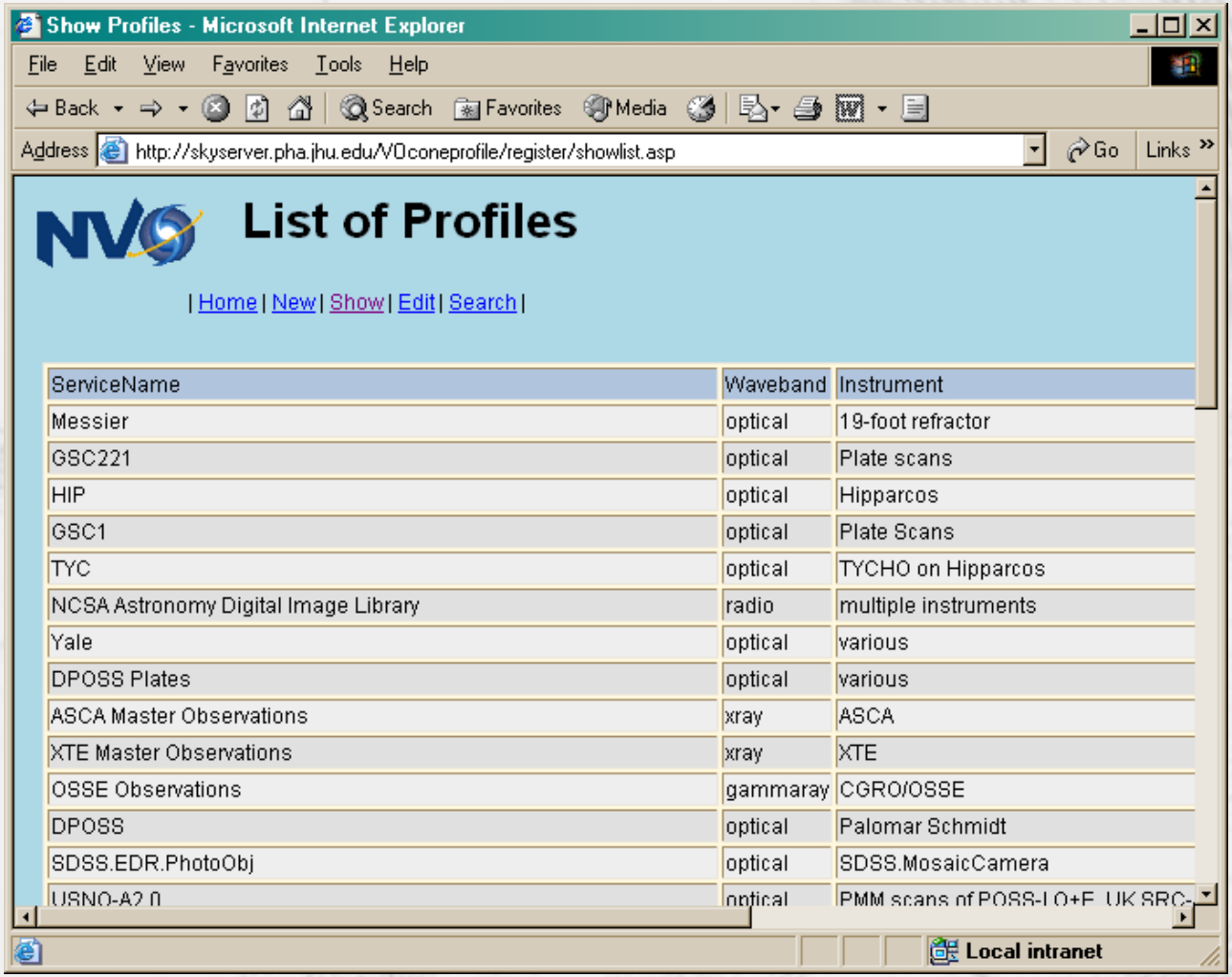
Practical Benefits

- Programmatic Access
- Platform/Language Independent
- Compose/Distribute

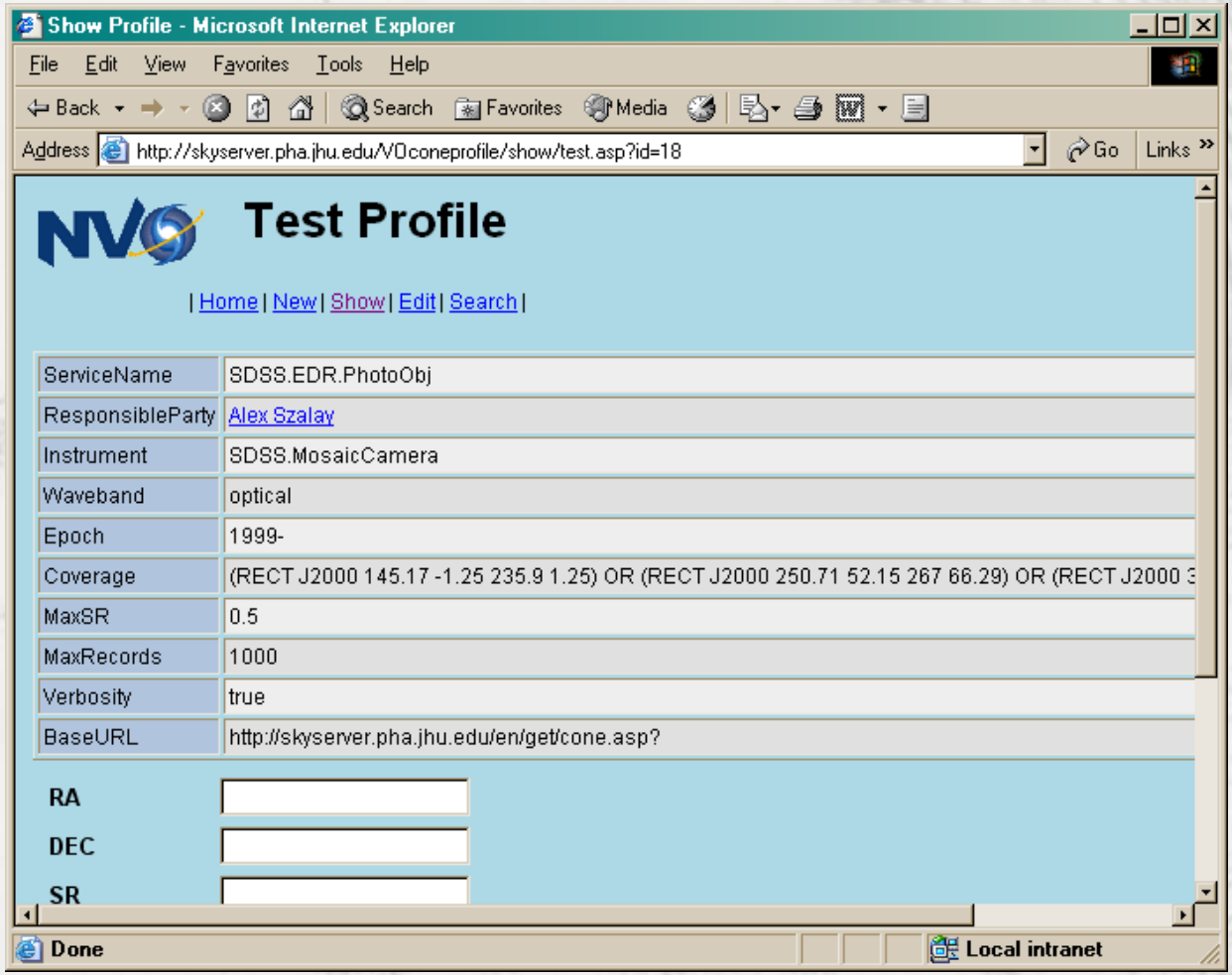
What about Astronomy

- Name Resolution
  - NED/SIMBAD Models
- Image Access
  - virtualsky
- Catalog Access
  - Intelligent Archive Queries
- Catalog Joins
  - Cross Identification Servers

Cone Search Profiles



SDSS EDR Cone Search



Web Service Paradigm

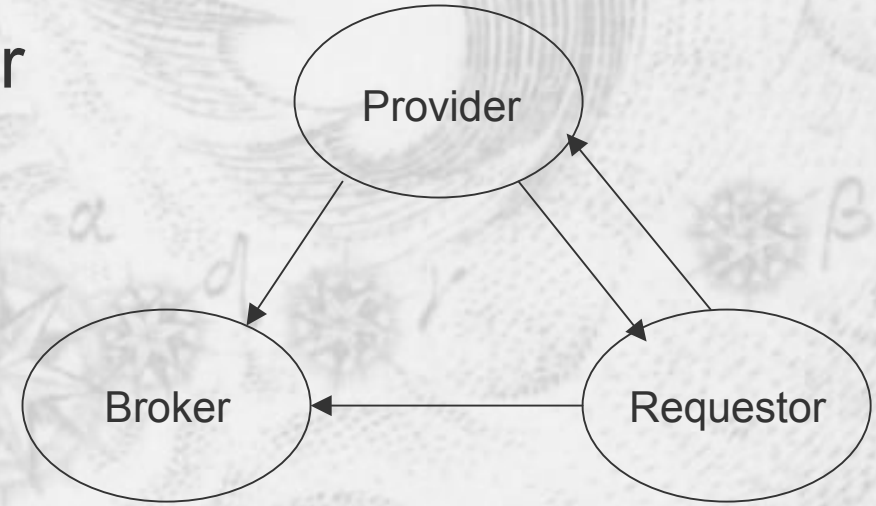
- Service Oriented Programming
  - Dynamically Locate Services
  - Services are “ON” the Network
  - Services can be coupled
- Multiple Transport Protocols
  - HTTP, SMTP, FTP, …
- Multiple Message Encodings
  - SOAP, XML-RPC, XP(?), …

Web Services for Astronomers

- What are Web Services
- Web Service Architecture
- Building Web Services
- The Future of Web Services

Web Service Architecture

- Three Primary Roles
  1. Service provider
  2. Service requester
  3. Service broker



Web Service Architecture

- Framework must support
  1. Publishing Service
  2. Finding a Service
  3. Binding a Service

Web Service Lifecycle

1. Service Must be Created
2. Service Must Be Published
3. Service Must be Easily Located
4. Service Must be Invoked/Called
5. Service must be Unpublished

Service Provider

- Creates the Service
  - New Service
  - Wrap Legacy Service
  - Wrap “Other” Services
- Publishes the Service
  - Registries
  - Standard Hierarchies
- Supports the Web Service
- Unpublishes the Service

Service Broker

- Maintains Service Registry
- Simplifies Service Location
  - Categorization
  - Query Support

Service Requestor

- Locates Service
- Invokes Service
  - Direct Request
  - Indirect Request

The Big Three

- Service Description – WSDL
  - The most important, everything else derives from this
- Service Invocation – SOAP

  Dominant Communication Protocol (XML Protocol)
- Service Publication – UDDI

  Being Pushed Hard, but future not clear. (OGSA)

Describing a Service

- Web Services Description Language (WSDL)
  http://www.w3.org/2002/ws/desc/
  - XML Document that provides the public interface to a Web Service
    - Public Methods
    - Data Type Information (IN/OUT)
    - Transport Protocol Binding Information
    - Service Location
  - The What, Where, and How!

Invoking a Service

- Simple Object Access Protocol (SOAP)
  - Although as of V1.2 SOAP is no longer an acronym

  http://www.w3.org/2000/xp/Group/
- XML protocol for exchanging messages
  - Platform/Language Independent
- Different Transport Protocols (General Case)
  - HTTP/HTTPR
  - SMTP
  - FTP
  - BEEP
  - …

Publishing a Service

- Universal Description, Discovery, and Integration (UDDI)

  http://www.uddi.org (Now under OASIS)
- Technical specification for building WSDL document repositories
  - Documents can be published
  - Document can be searched
  - Formal Hierarchy
- UDDI Registry implements the specification
  - IBM, Microsoft, SAP, etc. have public Registries
  - astrouddi.org (?)

Hello World (WSDL Style)

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
  targetNamespace="http://localhost:8080/axis/HelloWorld.jws"
  xmlns:impl="http://localhost:8080/axis/HelloWorld.jws"
  xmlns:intf="http://localhost:8080/axis/HelloWorld.jws"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://localhost:8080/axis/HelloWorld.jws">
      <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="mainRequest">
    <wsdl:part name="args" type="impl:ArrayOf_xsd_string"/>
  </wsdl:message>
  <wsdl:message name="getMessageResponse">
    <wsdl:part name="getMessageReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="getMessageRequest">
  </wsdl:message>
  <wsdl:message name="mainResponse">
  </wsdl:message>
  <wsdl:portType name="HelloWorld">
    <wsdl:operation name="main" parameterOrder="args">
      <wsdl:input name="mainRequest" message="impl:mainRequest"/>
      <wsdl:output name="mainResponse" message="impl:mainResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getMessage">
      <wsdl:input name="getMessageRequest" message="impl:getMessageRequest"/>
      <wsdl:output name="getMessageResponse" message="impl:getMessageResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="HelloWorldSoapBinding" type="impl:HelloWorld">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="main">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="mainRequest">
        <wsdlsoap:body use="encoded"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://localhost:8080/axis/HelloWorld.jws"/>
      </wsdl:input>
      <wsdl:output name="mainResponse">
        <wsdlsoap:body use="encoded"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://localhost:8080/axis/HelloWorld.jws"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getMessage">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getMessageRequest">
        <wsdlsoap:body use="encoded"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://localhost:8080/axis/HelloWorld.jws"/>
      </wsdl:input>
      <wsdl:output name="getMessageResponse">
        <wsdlsoap:body use="encoded"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://localhost:8080/axis/HelloWorld.jws"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="HelloWorldService">
    <wsdl:port name="HelloWorld" binding="impl:HelloWorldSoapBinding">
      <wsdlsoap:address location="http://localhost:8080/axis/HelloWorld.jws"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

WSDL Definitions Element

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
  targetNamespace="http://localhost:8080/axis/HelloWorld.jws"
  xmlns:impl="http://localhost:8080/axis/HelloWorld.jws"
  xmlns:intf="http://localhost:8080/axis/HelloWorld.jws"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
…
</wsdl:definitions>
```

WSDL Document Elements

- <wsdl:types>
  The datatypes used by the Web Service
- <wsdl:message>
  The abstract definition of the data being transmitted
- <wsdl:portType>
  The abstract operations that constitute the Web service
- <wsdl:binding>
  The concrete protocol and data format used by the Web service
- <wsdl:port>
  The address for a single communication endpoint
- <wsdl:service>
  An aggregation of related ports

WSDL Types

- Define the datatypes used as arguments to the Web service as well as the return values from a Web service
- Preferably XML Schema
  - XSD namespace
- Must Handle nillable (Java Wrapper Classes)
- SOAP

WSDL Types

- Map WSDL (XSD) to Language (e.g., Java)

xsd:boolean	boolean
xsd:byte	byte
xsd:double	double
xsd:float	float
xsd:int	int
xsd:long	long
xsd:short	short
xsd:dateTime	java.util.Calendar
xsd:decimal	java.math.BigDecimal
xsd:hexBinary	byte[]
xsd:base64Binary	byte[]
xsd:QName	javax.xml.namespace.QName
xsd:integer	java.math.BigInteger
xsd:string	java.lang.String

WSDL Types

- Recommended approach
  - Use Elements not Attributes
  - Only define types that refer to abstract content of messages (not protocols)
  - Array types should extend the SOAP Array type
    - Name scheme: ArrayOfXXX
  - Xsd:anyType used to represent any type.

<wsdl:types>

```
<wsdl:types>
  <schema xmlns="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://localhost:8080/axis/HelloWorld.jws">
    <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>
```

Web service Messages

- Interactions between Web service client and server are called messages
- Message element describes the messages that can be exchanged
- Logical definition of a type of message that may be used by operations listed in portType element
  - Input
  - Output
  - Fault Message
- Components
  - Message must have a local name

Web service Messages

- Components (wsdl:message element)
  - Message must have a local name
  - Use WSDL Namespace
  - Zero or more Part descriptions
    - part name
    - part type
    - Arguments or return parameters.
    - Should follow XML Schema
- Message element Future?

<wsdl:message>

```
<wsdl:message name="mainRequest">
  <wsdl:part name="args" type="impl:ArrayOf_xsd_string"/>
</wsdl:message>
<wsdl:message name="getMessageResponse">
  <wsdl:part name="getMessageReturn" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getMessageRequest">
</wsdl:message>
<wsdl:message name="mainResponse">
</wsdl:message>
```

WSDL Port Types

WSDL defines four transmission primitives (or operations) that an endpoint can support

- **One-way (input element)**
  - The endpoint receives a request, but does not send a response.
- **Request-response (input then output element)**
  - The endpoint receives a request, and sends a correlated response.
- **Solicit-response (output then input element)**
  - The endpoint sends a response, and receives a correlated response.
- **Notification** (output element)
  - The endpoint sends a response, but does not receive a request.

WSDL portType

- A portType element defines the interfaces that a Web service exposes.
- Similar to a
  - Class
  - Module
  - or Function Library
- The operations are the class/module/library methods.

<wsdl:portType>

```
<wsdl:portType name="HelloWorld">
  <wsdl:operation name="main" parameterOrder="args">
    <wsdl:input name="mainRequest"
      message="impl:mainRequest"/>
    <wsdl:output name="mainResponse"
      message="impl:mainResponse"/>
  </wsdl:operation>
  <wsdl:operation name="getMessage">
    <wsdl:input name="getMessageRequest"
      message="impl:getMessageRequest"/>
    <wsdl:output name="getMessageResponse"
      message="impl:getMessageResponse"/>
  </wsdl:operation>
</wsdl:portType>
```

WSDL Binding

- Defines message format
- For a given portType, defines protocol
  - for operations
  - for messages
- Requires unique name attribute
- Type attribute is portType Qname

<wsdl:binding>

```
<wsdl:binding name="HelloWorldSoapBinding" type="impl:HelloWorld">
  <wsdlsoap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
...
  <wsdl:operation name="getMessage">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="getMessageRequest">
      <wsdlsoap:body use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://localhost:8080/axis/HelloWorld.jws"/>
    </wsdl:input>
    <wsdl:output name="getMessageResponse">
      <wsdlsoap:body use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://localhost:8080/axis/HelloWorld.jws"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

WSDL Services

- A port defines a single endpoint
- The port can then be used for binding
- Multiple ports can reference the same address with different protocols
- A Service consists of one or more ports
- A service defines a single serviceType

<wsdl:service> & <wsdl:port>

```
<wsdl:service name="HelloWorldService">
  <wsdl:port
    name="HelloWorld"
    binding="impl:HelloWorldSoapBinding">
    <wsdlsoap:address
      location="http://localhost:8080/axis/HelloWorld.jws"/>
  </wsdl:port>
</wsdl:service>
```

Invoking a Service

- Use SOAP to communicate messages
  - SOAP Sender to SOAP Receiver
  - Potential SOAP Intermediaries
- Essentially a one-way communication between SOAP nodes.
  - RPC style
  - Document style

SOAP Basics

- Message is wrapped in the Envelope
- Envelope consists of
  - Header (Optional) used by intermediaries
  - Body contains the actual message
    - Document
    - Service Call
- Fault Handling
  - Child element of body
  - Contains Reason and Code elements

SOAP Basics

- Fault Handling (V1.2)
  - Fault Element is a child element of body
    - No other elements in the body
  - Contains
    - Reason element (Mandatory)
    - Code element (Mandatory)
      - Standard List
    - Detail element (Optional)
    - Node element (Optional)
    - Role element (Optional)

SOAP Request (HelloWorld)

```
POST /axis/HelloWorld.jws HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.0
Host: localhost
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 407

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:getMessage
   soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:ns1="http://localhost:8080/axis/HelloWorld.jws"/>
 </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response (HelloWorld)

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Connection: close
Date: Wed, 09 Oct 2002 21:34:47 GMT
Server: Apache Tomcat/4.0.6 (HTTP/1.1 Connector)
Set-Cookie: JSESSIONID=8A6802F3136B882A53BC0E8E1E30F8CC;Path=/axis

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:getMessageResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:ns1="http://localhost:8080/axis/HelloWorld.jws">
      <getMessageReturn xsi:type="xsd:string">Hello World!</getMessageReturn>
    </ns1:getMessageResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

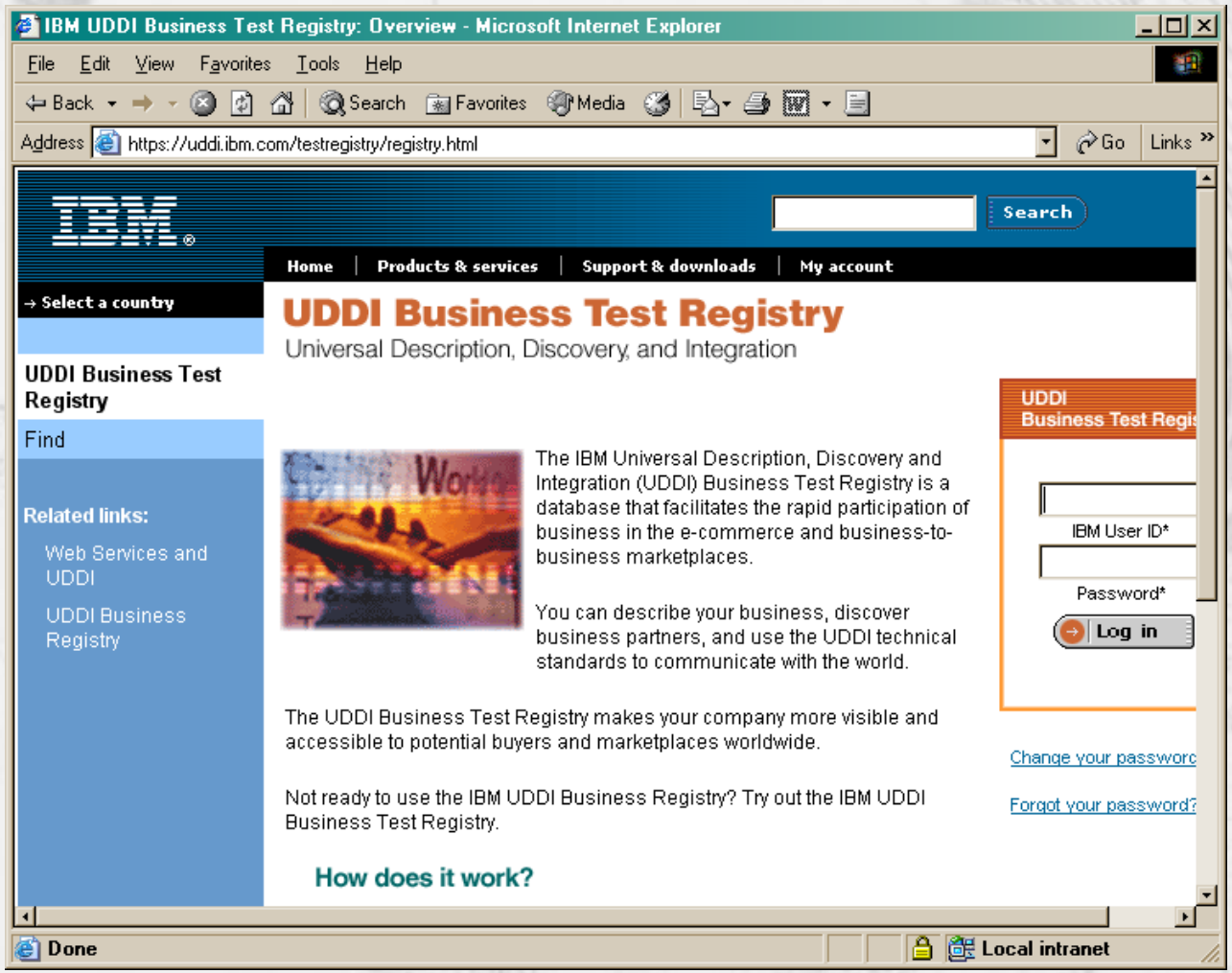
Web Service Registries

- UDDI Currently Dominant
  - Public Registries
    - IBM, MS, SAP, etc.
  - Private Registries
  - UDDI Functions
    - Describe services
    - Discover businesses
    - Integrate business services
- The MetaData Problem

UDDI Registry

- Business Entity
  - Can have multiple Services
- Business Service
  - Has an associated specification
- Specification Pointers
  - Detailed information on service
- Service Types
  - Defined by a tModel
  - tModel and WSDL

UDDI Registry



UDDI Private Registry

- Some development tools or products provide private UDDI registry server
  - Java WS Developer pack.
  - Oracle JDeveloper
  - IBM WS toolkit
  - MS VS .NET
- Greater control, no registration!

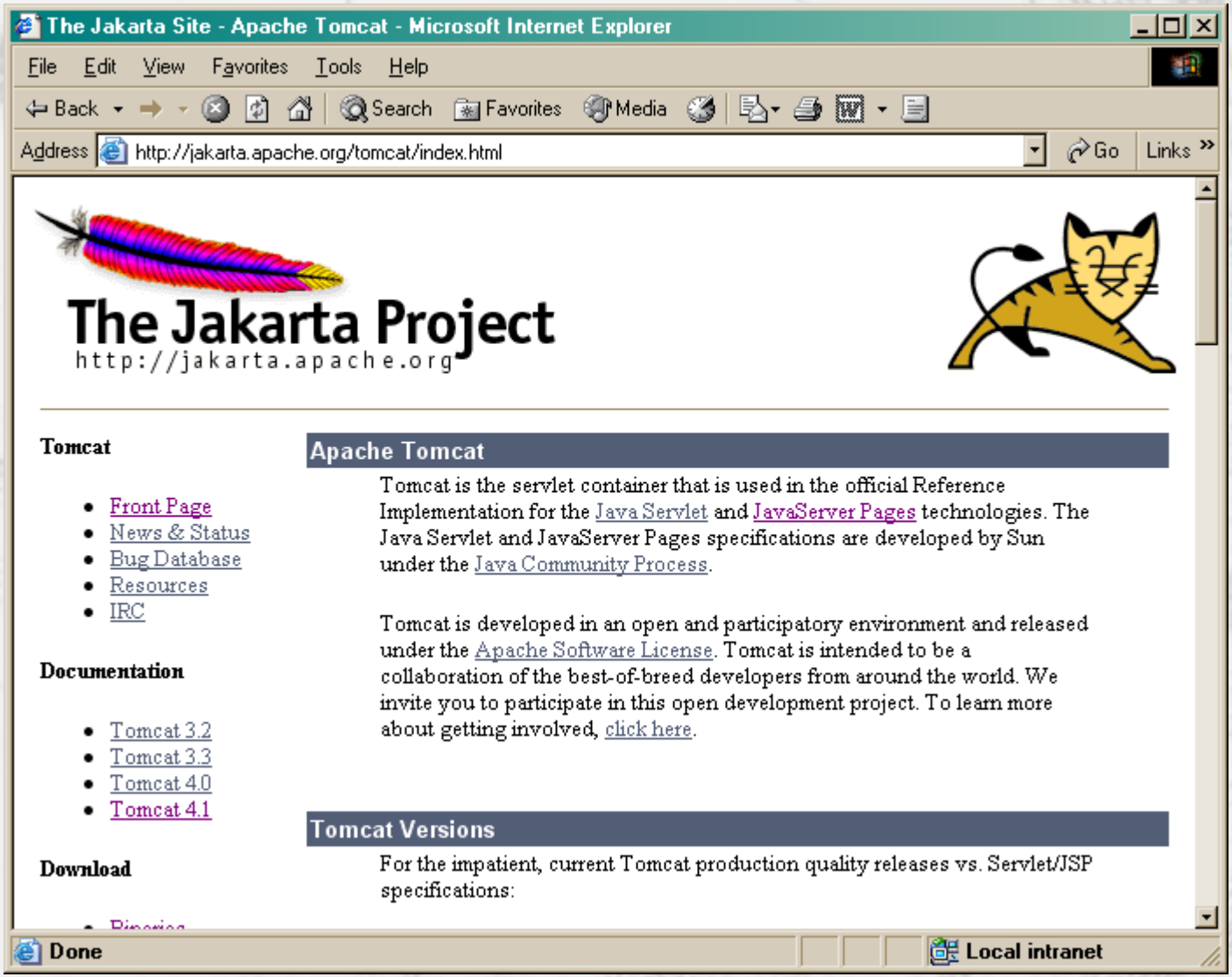
Web Services for Astronomers

- What are Web Services
- Web Service Architecture
- Building Web Services
- The Future of Web Services

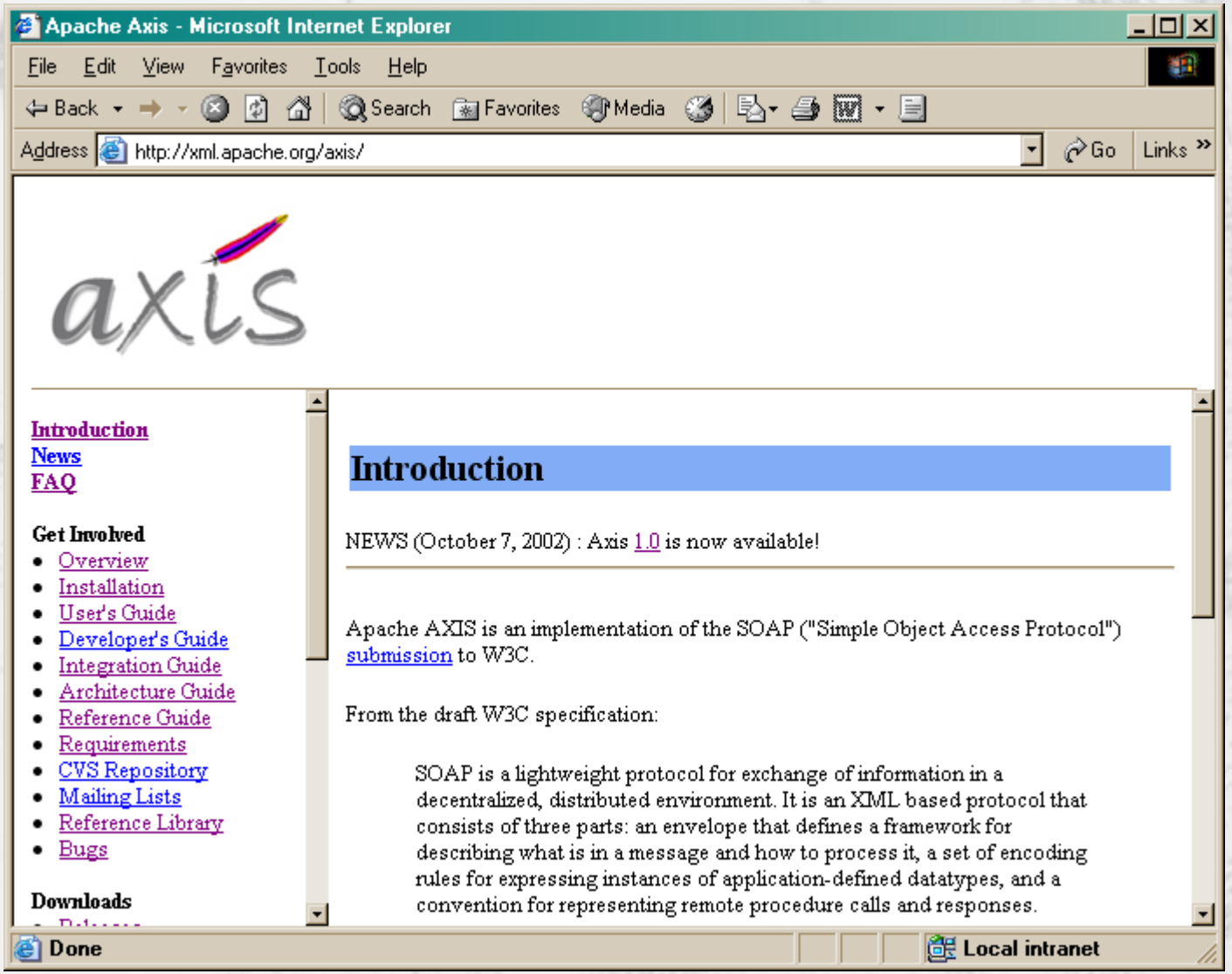
Building Web Services

- Simple Demonstration of Deploying a Web service
  - Use Java (but other options exist: .NET, Perl, python, etc.)

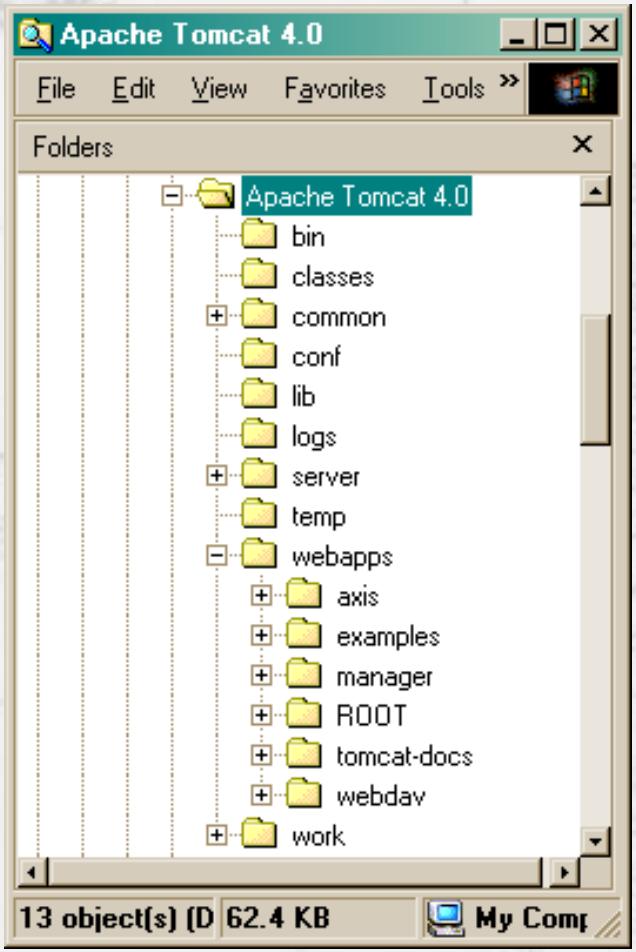
Tomcat Server



AXIS SOAP Server



Installation & Setup



- Install Tomcat
- Deploy Axis web apps into Tomcat webapps directory.
- Start Tomcat Server
- Validate AXIS Installation

Web Service Deployment

- Simple Technique (JWS)
  - Copy Java Source file containing the method(s) to be exposed to axis directory
    - HelloWorld.java -> HelloWorld.jws
- Complex Technique (WSDD)
  - AXIS solution
  - Web Service Deployment Descriptor
- Annotations (.NET approach)
  - [WebMethod]

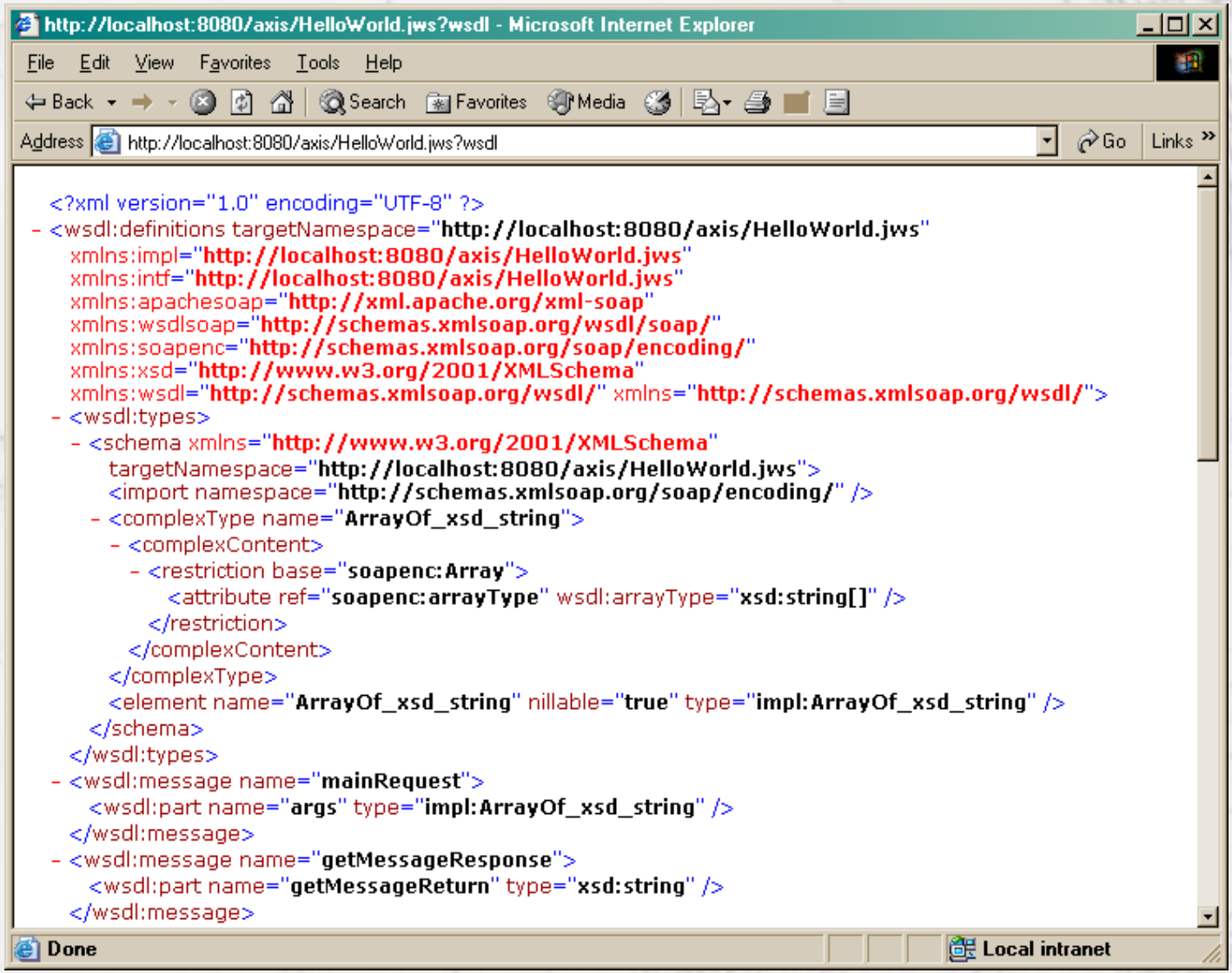
Hello World (Java)

```
public class HelloWorld {
    public java.lang.String getMessage() {
        return "Hello World!" ;
    }
}
```

Hello World (CSharp)

```
using System.Web.Services ;
public class HelloWorld : WebService {
    [WebMethod]
    public string getMessage() {
        return "Hello World!" ;
    }
}
```

View WSDL


http://localhost:8080/axis/HelloWorld.jws?wsdl - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/axis/HelloWorld.jws?wsdl
Go Links
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://localhost:8080/axis/HelloWorld.jws"
xmlns:impl="http://localhost:8080/axis/HelloWorld.jws"
xmlns:intf="http://localhost:8080/axis/HelloWorld.jws"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/">
<wsdl:types>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://localhost:8080/axis/HelloWorld.jws">
<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="mainRequest">
<wsdl:part name="args" type="impl:ArrayOf_xsd_string" />
</wsdl:message>
<wsdl:message name="getMessageResponse">
<wsdl:part name="getMessageReturn" type="xsd:string" />
</wsdl:message>
Done
Local intranet

Web Service Client

- Generate Client Stub from WSDL
  - wsdl2java tool included with AXIS

```
>java org.apache.axis.wsdl.WSDL2Java
    http://localhost:8080/axis/HelloWorld.jws?wsdl
```

  - Generates
    - localhost\HelloWorld.java
    - localhost\HelloWorldService.java
    - localhost\HelloWorldServiceLocator.java
    - localhost\HelloWorldSoapBindingStub.java

Utilizing the Stub Classes

- HelloWorldClient.java

```
package localhost ;

public class HelloWorldClient
{
    public static void main(String[] args) throws Exception {
        // Make a service
        HelloWorldService service = new HelloWorldServiceLocator();

        // Now use the service to get a stub
        HelloWorld port = service.getHelloWorld();

        System.out.println(port.getMessage());
    }
}
```

AXIS Extras

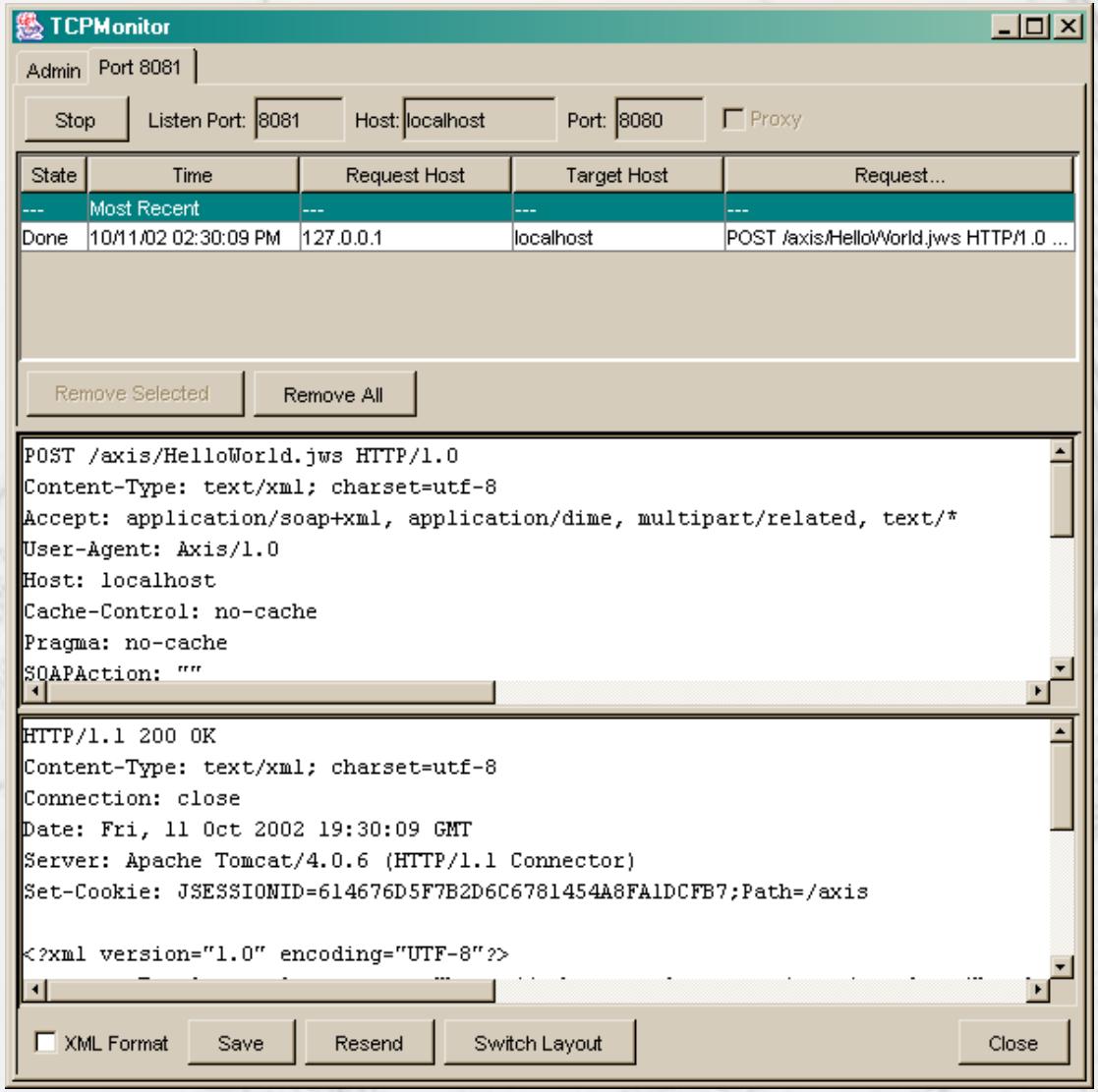
- Generate Server Skeleton Stub from WSDL

>java org.apache.axis.wsdl.WSDL2Java –s http://localhost:8080/axis/HelloWorld.jws?wsdl

  - Generates
    - localhost\HelloWorldSoapBindingImpl.java

- More arguments for additional functionality

AXIS TCP Monitor



Web Services for Astronomers

- What are Web Services
- Web Service Architecture
- Building Web Services
- The Future of Web Services

Roadblocks or Speedbumps?

- Reliable Protocol Needed (HTTPR, BEEP)
- Lack of State
- Implementation Inconsistencies
  - unsigned
  - multipart/structures
- Security!

Reliable Protocols

- HTTP – Reliable HTTP
  - IBM Initiative

    http://www-106.ibm.com/developerworks/library/ws-phtt/
  - Adds Persistence to HTTP
- BEEP (Blocks Extensible Exchange Protocol

  http://www.ietf.org/rfc/rfc3080.txt
  - Connection-oriented
  - Asynchronous interactions

DIME (Direct Internet Message Encapsulation)

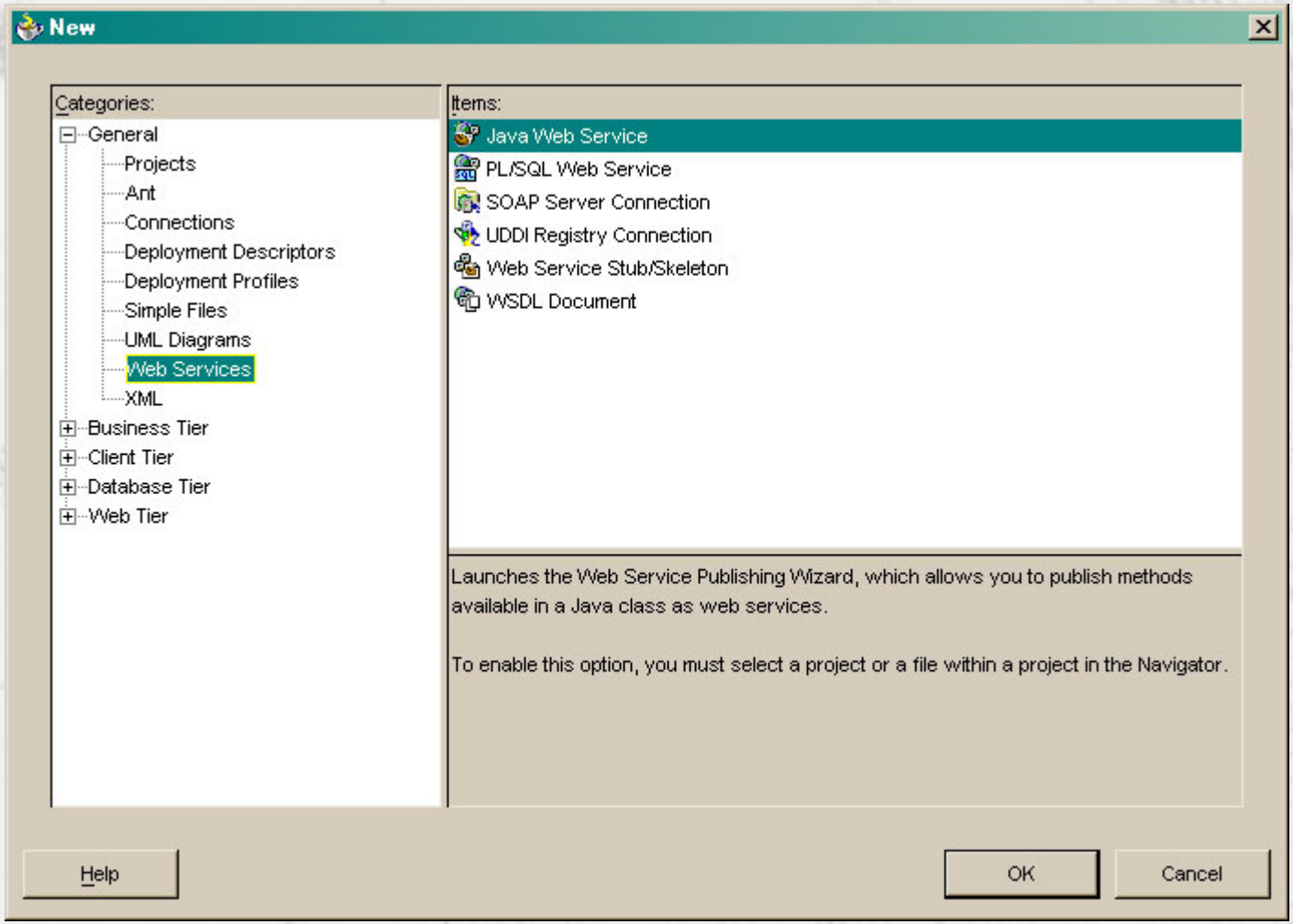
- General purpose binary message format
- Enable Web services to efficiently handle multiple attachments
  - Encrypted messages
  - Graphics
  - Multimedia content
  - General Documents
- DIME Message (application/dime)
  - 1+ records to deliver payload
  - Chunking

http://www.ietf.org/internet-drafts/draft-nielsen-dime-02.txt

BPEL4WS

- Business Process Execution Language for Web Services
  - Implementing executable business processes.
  - Describing non-executable abstract processes.
- Merging of WSFL and Xlang
  - Ugliest WS Acronym award
- Define new Web service as a composition of existing Web services

Simplify Development/Deployment



J2EE Web Services

- Java APIs for XML
  - JAX-RPC
  - JAXM (SAAJ)
  - JAXR
- JSR 109 - Implementing Enterprise Web Services
- JSR 110 - Java APIs for WSDL

Security

- Issues include
  - Message Integrity
  - Message Confidentiality
  - Authentication
- Technologies include
  - Secure Sockets Layer (SSL)
  - Transport Layer Security (TLS)
  - Message Encryption
  - Digital Signatures
- But Standards !!!!!

Summary

- Web services provide a powerful programming paradigm
- Mucho Hype
- Looking for Real Applications (NVO)
- Open Grid Services Architecture (OGSA)