

Writing and Using Web Services

or

I don't care where my programs are

Rajarshi Guha
Indiana University
rguha@indiana.edu

Overview

- Two 45 minute sessions, with 15 minute break
- Some theory, but mainly focus on '*How do I do ...?*' type problems
- At the end you should be able to set up, write and provide your own web services
- I'll try to be (somewhat) language agnostic
 - I'll focus on Java but also talk about Python
- I'll focus on freely available software

Overview

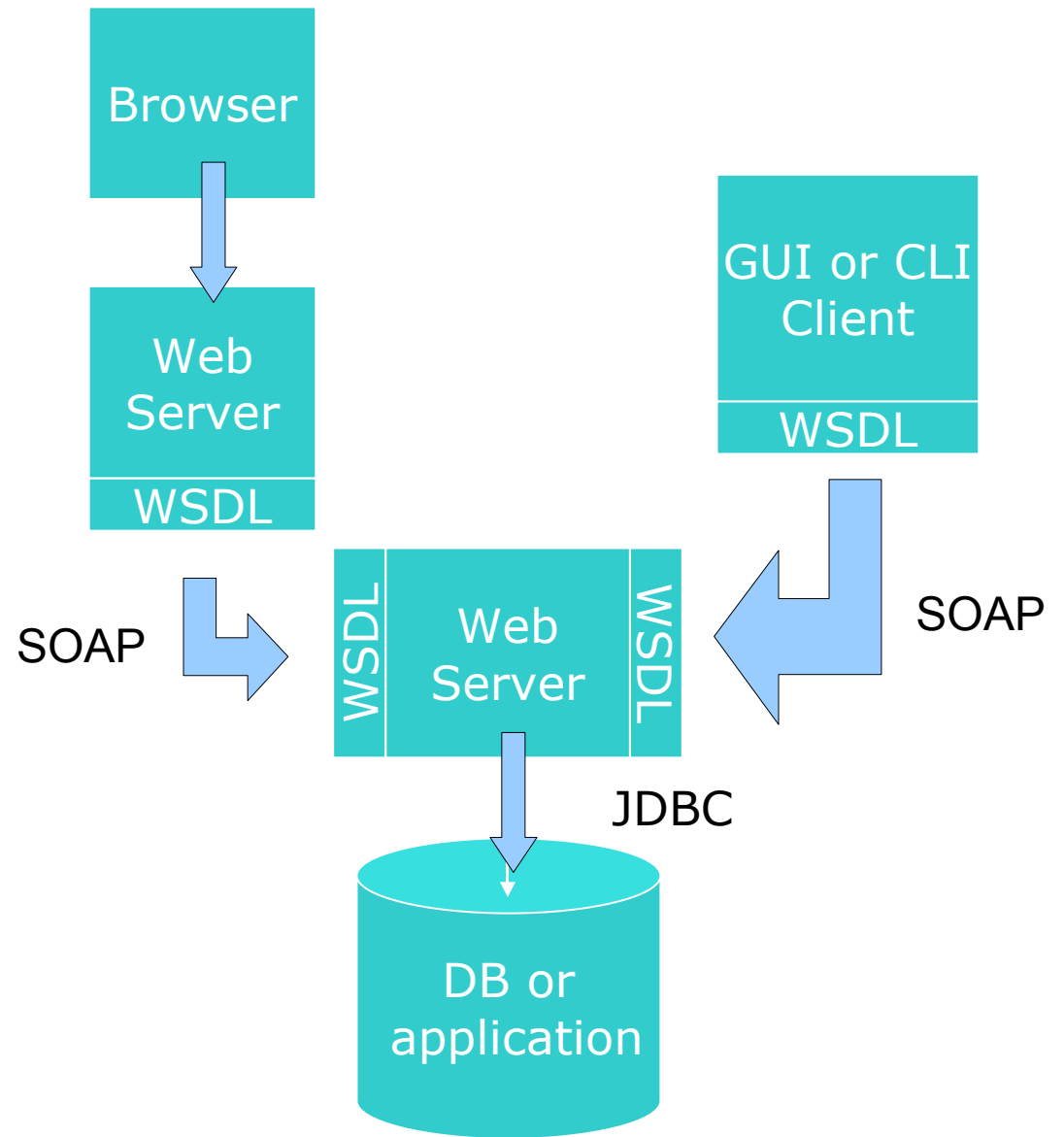
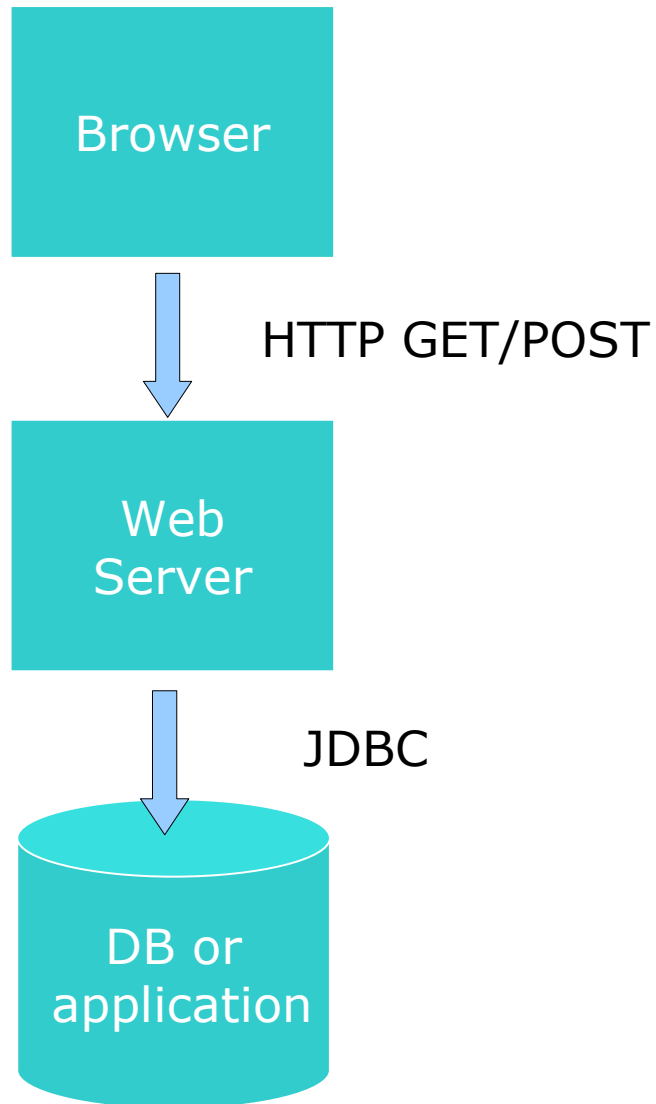
- Some of the topics I'll be covering
 - Software requirements
 - Protocols – SOAP & WSDL
 - Hosting web services
 - Writing a service
 - Writing clients
- Examples will be from cheminformatics – I'll be using the CDK to provide the functionality

Why Use Web Services?

What is a Web Service (WS) ?

- Code running on an arbitrary machine that can be accessed in a platform- and language-independent fashion
- More generally, a WS framework is an XML-based distributed services system.
 - Suitable for machine-to-machine interaction
 - Platform & language agnostic
 - Follows standards

Basic Architectures: CGI and Web Services



When is a WS Called For?

- Want to make algorithms available
 - avoid downloads
 - avoid support requests
 - if your algorithm takes milliseconds (or longer), it can be used in a WS
- Avoid direct database access

Why Not CGI?

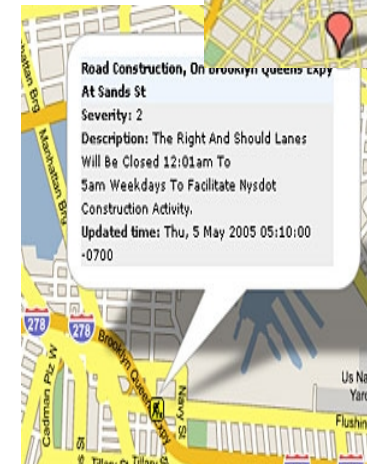
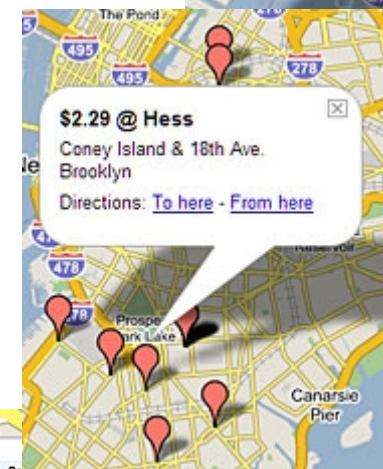
- CGI has been around for a long time
- Returns HTML
 - nice if your interface is a browser
 - if not, parse the HTML – not fun!
- Can be a security issue
- **But**, it is possible to use CGI & web services

What Can They Do For Us?

- Generally we have to *get* software (programs, data)
 - Might involve compiling, debugging
 - Usually involves administration
 - Hassles!
- It would be nice to simply make calls to a library, that we don't have to manage
- A *WS* is essentially a set of function calls that a remote user can send input to and get a return value – *this is the RPC view*

Famous Web Services

- Google
 - Access their search engine
 - Access Google Maps
- Amazon
 - Access their search engine
 - Access historical data



Chemistry & Biology Web Services

- Indiana University, USA
- Cambridge University, UK
- NCBI, USA
- PathPort, Virginia Tech, USA
- EBI, UK
- KEGG Services, Kyoto University, Japan
- The [Taverna](#) project has a *registry* scientific web services

IU Web Services

- Variety of public cheminformatics services
 - Molecular descriptors
 - 2D structure diagrams
 - TPSA
 - Fingerprints
 - Similarity calculations
 - Toxicity class predictions

Caveats

- Web services are not a panacea
- Some issues may arise
 - Security
 - Provenance – what is the service *really* doing?
 - Needs a Net connection
 - Reliability

Software Requirements

Some Terminology

- **SOAP – Simple Object Access Protocol.**
Describes how data will be encoded when it's sent to a service and decoded when received from a service
- **WSDL – Web Service Description Language**
Describes how a client should interact with a web service
- **Application Container** – software that hosts services

The Components of a WS

- **Server side**
 - Web server
 - Application container
 - Service code
- Web server can be optional
- Container can be optional
- **Client side**
 - Some libraries
 - Client code
- Client code might be
 - Stand alone program
 - Web browser

Programming Languages

- Your favorite language!
- Support exists for
 - C/C++
 - Java
 - Perl
 - Python
 - PHP
 - Ruby
 - .Net (so you can access WS's from Excel)

Server Side Components (Java)

- Apache web server
 - Before v2.2 you will need to get **mod_jk**
 - From v2.2 onwards use **mod_proxy_ajp** – comes bundled
 - Latter is easier, but can involve security risks
- Application Container
 - Tomcat
 - JBoss
- SOAP libraries and any other dependencies

Server Side Components

- You don't have to have an explicit web server
- But makes management of your setup easier
 - Load balancing
 - URL rewriting
 - Efficient
- If you only have a single service you can even drop the container and run your service code directly - depends on what type of set up you have

Client Side

- Some SOAP libraries or packages, depending on your language
- Your client code
 - This can be standalone code
 - This could a web page (using PHP)

Personal Experience

- I tried using other peoples setups – failed
- Decided to see how long it takes to get my own
 - Less than one day – with no experience of application containers, SOAP etc
- The time consuming step is setting up your web server and application container
- Service and client code is easy!
- **Caveat:** This got me up and running, but not necessarily securely or optimally

Setting Up The Service Environment

What Environment Do We Use?

- For now we're going to consider Java services and clients
- So we'll be working with
 - Apache web server
 - Tomcat
 - AXIS libraries
- We'll look at other possibilities later on

Setting Up Apache (2.2)

- I'll assume that you will be using `mod_proxy_ajp` and running Linux
- We'll run the whole setup on the *local* machine
- In your *httpd.conf* file you need

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
...
ProxyRequests Off
ProxyPass /axis http://localhost:8080/axis
ProxyPassreverse /axis http://localhost:8080/
```


Setting Up Apache

- To restrict access to the proxied resource, add to *httpd.conf*

```
<Proxy http://your.domain.name/axis/*>  
  Order deny,allow  
  Allow from all  
  AuthType Basic  
  AuthName "Tomcat Axis Webservices"  
  AuthUserFile /path/to/apachepasswords  
  Require user user1 user2  
</Proxy>
```

- Restart the web server

Setting Up Tomcat

- We'll be using Tomcat 5.5 & JDK 1.5
- Need to define environment variables
 - CATALINA_HOME – points to the Tomcat installation
 - JAVA_HOME – points to the JDK
- Add a user and group, say, tomcat

```
groupadd tomcat
useradd -g tomcat -c "Tomcat User" \
  -d $CATALINA_HOME tomcat
passwd tomcat
```

Setting Up Tomcat

- We'll have to edit the config file – server.xml
- Make sure it contains:

```
<Connector  
  port="8009"  
  enableLookups="false"  
  redirectPort="8443"  
  protocol="AJP/1.3"  
>
```

```
<Connector  
  port="8080"  
  enableLookups="false"  
  redirectPort="8443"  
>
```

Setting Up Tomcat

- If some of your web services are going to depend on some common jar files, place them in `$CATALINA_HOME/shared/lib`
- This keeps individual web services simple to manage

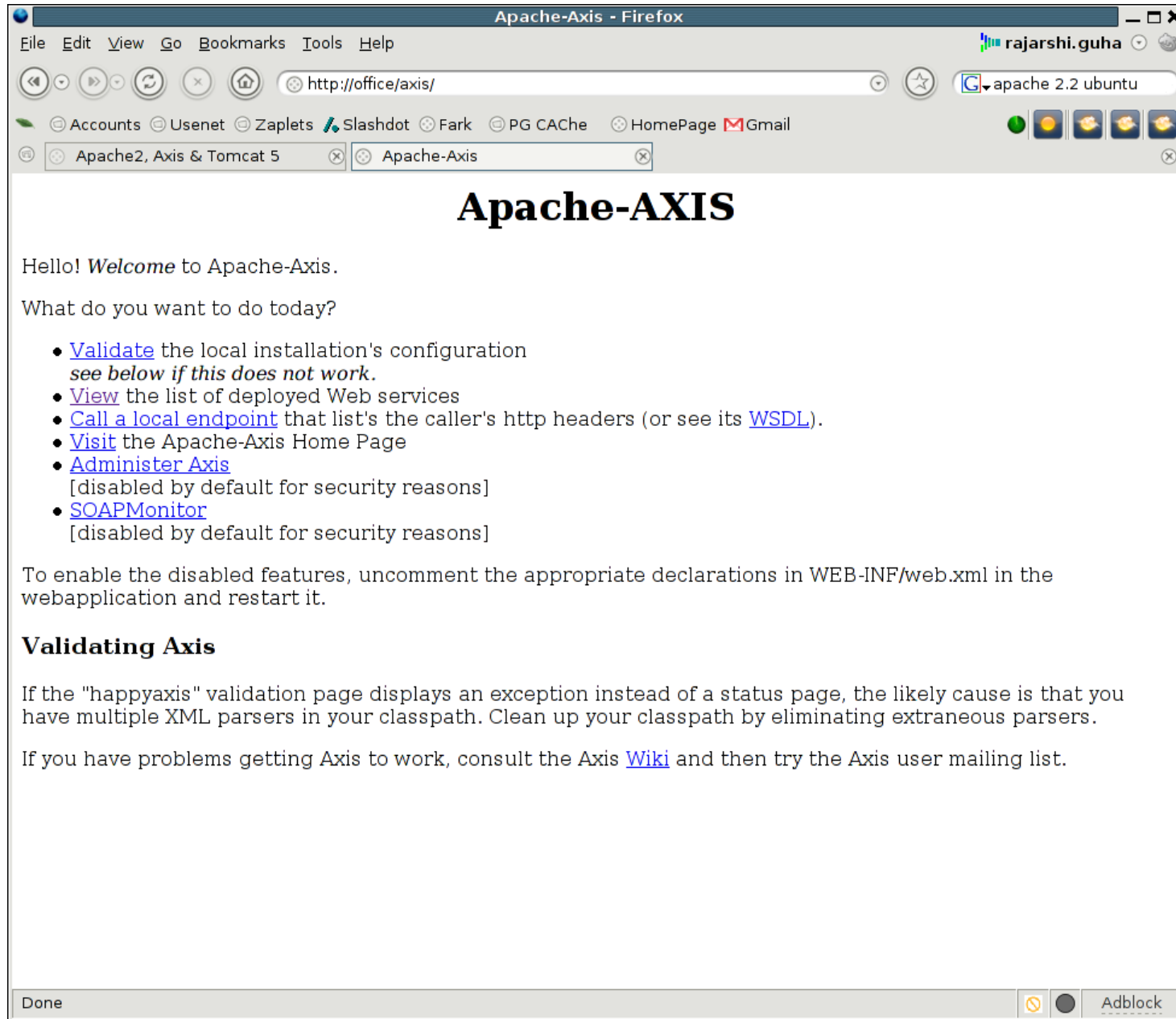
Setting Up AXIS

- AXIS is a set of libraries that allow the service code and the client code to understand SOAP
- Usually, all your WS's will need AXIS
- So we can place all the AXIS jars in `$CATALINA_HOME/shared/lib`
- For now we'll be a little crude:
 - From the AXIS tarball copy `webapps/axis` to `$CATALINA_HOME/webapps`

And it All Comes Together

- Start up Tomcat:
 - `sudo - tomcat -c $CATALINA_HOME/bin/startup.sh`
- Restart Apache
 - `/etc/init.d/httpd restart`
- Browse to <http://localhost/axis/>
 - If you choose not to run Apache then browse to <http://localhost:8080/axis/>
- All of this is described in
 - [Setting up Apache2, Axis & Tomcat 5](#) on my website

.... to Give



Apache-Axis - Firefox

File Edit View Go Bookmarks Tools Help

http://office/axis/

Accounts Usenet Zaplets Slashdot Fark PG CACHE HomePage Gmail

Apache2, Axis & Tomcat 5 Apache-Axis

Apache-AXIS

Hello! *Welcome* to Apache-Axis.

What do you want to do today?

- [Validate](#) the local installation's configuration
see below if this does not work.
- [View](#) the list of deployed Web services
- [Call a local endpoint](#) that list's the caller's http headers (or see its [WSDL](#)).
- [Visit](#) the Apache-Axis Home Page
- [Administer Axis](#)
[disabled by default for security reasons]
- [SOAPMonitor](#)
[disabled by default for security reasons]

To enable the disabled features, uncomment the appropriate declarations in WEB-INF/web.xml in the webapplication and restart it.

Validating Axis

If the "happyaxis" validation page displays an exception instead of a status page, the likely cause is that you have multiple XML parsers in your classpath. Clean up your classpath by eliminating extraneous parsers.

If you have problems getting Axis to work, consult the Axis [Wiki](#) and then try the Axis user mailing list.

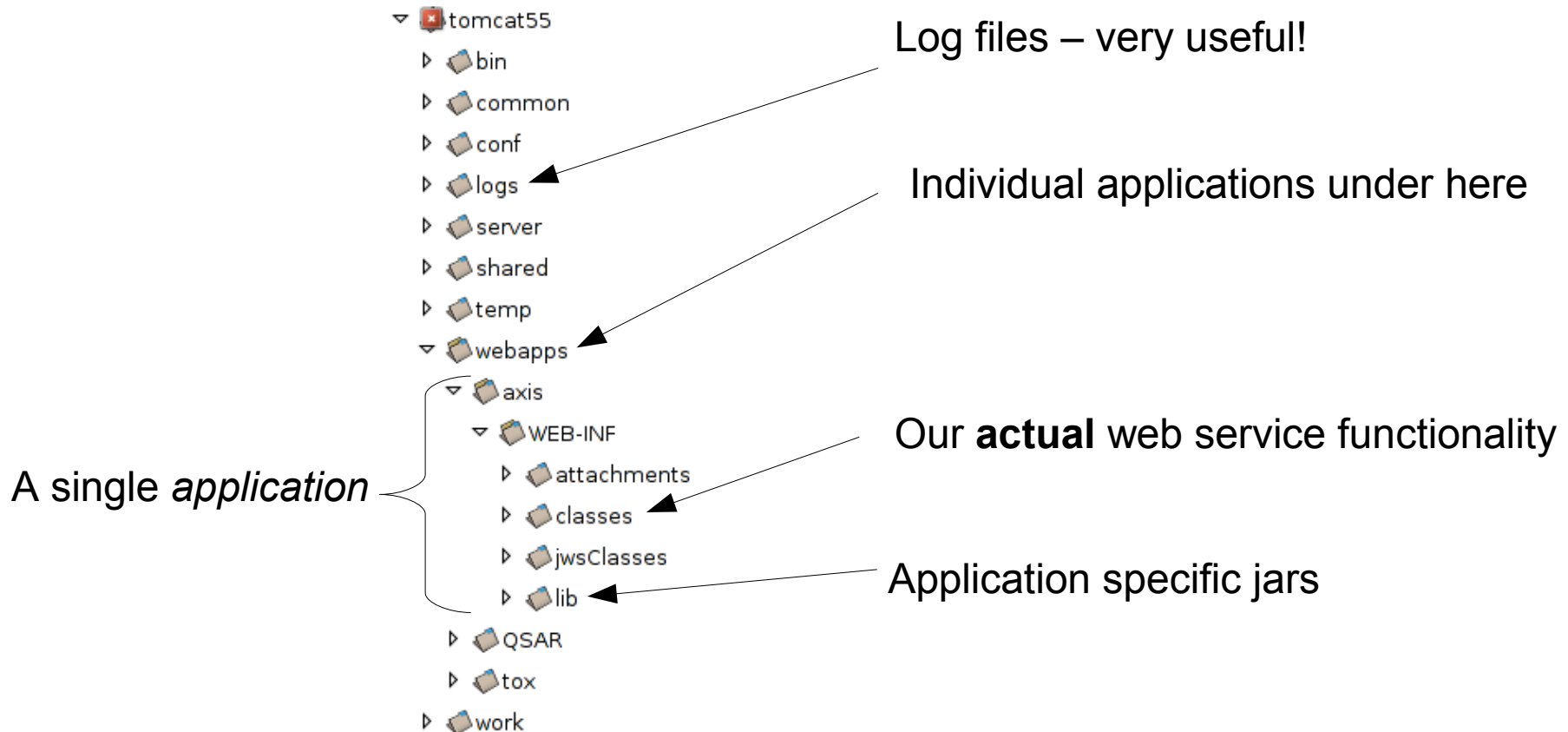
Done

Adblock

Writing a Web Service

Uptil Now

- We've installed a web server, app environment, some libraries
- We're ready to write and provide a web service



Getting Ready to Develop

- We're going to use the **CDK** to make some web services
- The CDK has methods for
 - fingerprints, 2D structure diagrams
 - descriptors, molecular weights & formulae
- Get the CDK jar and place it in your `CLASSPATH`
- Also, copy the jar to the proper Tomcat directory

Getting Ready to Develop

- We don't need to bother with SOAP libraries and Tomcat and so on, at this point

Writing a service is the same as writing an ordinary Java program

- It's not always as simple as this

Getting Fingerprints

- The CDK fingerprinter needs 3 things
 - An **IAtomContainer** object
 - how many bits
 - search depth
- And returns a **BitSet** object
- **Problems**
 - SOAP doesn't know about these objects
 - We are going to have a SMILES as input
 - We want bit positions as output

Getting Fingerprints

- Solutions

- The service will take a String object and output a String (for now)

```
public class CDKws {
    public String getFingerprintString(String s, int length, int depth) throws CDKException {
        String print = "";
        IMolecule mol = null;
        SmilesParser sp = new SmilesParser();
        mol = sp.parseSmiles(s);
        if (mol == null) {
            throw new CDKException("SmilesParser exception");
        }
        Fingerprinter fprinter = new Fingerprinter(length, depth);
        try {
            print = fprinter.getFingerprint(mol).toString();
        } catch (Exception e) {
            throw new CDKException("Fingerprinter error");
        }
        return(print);
    }
}
```

Getting Fingerprints

- If we called this web service with CC=CCOC, 1024, 6 and processed the SOAP payload, we'd get
 - “{75, 284, 323, 476, 500, 540, 544, 588, 633, 637, 741, 742, 831, 846}”
- So our client would have to parse the String object to get the actual bit positions
 - Not elegant!
- But, AXIS SOAP allows us to return **Vector** objects (as well as plain arrays)
- Let's add another method to the **CDKws** class

Getting Fingerprints

```
public Vector getFingerprintVector(String s, int length, int depth) throws CDKException {
    IMolecule mol = null;
    SmilesParser sp = new SmilesParser();
    mol = sp.parseSmiles(s);
    if (mol == null) {
        throw new CDKException("SmilesParser exception");
    }
    Fingerprinter fprinter = new Fingerprinter(length, depth);
    String fp = null;
    try {
        fp = fprinter.getFingerprint(mol).toString();
    } catch (Exception e) {
        throw new CDKException("Fingerprinter error");
    }
    String[] st = fp.substring(1,fp.length()-1).split("[\\s,]"); // I got lazy
    Vector v = new Vector();
    for (int i = 0; i < st.length; i++) {
        if (st[i].equals("")) continue;
        v.add( new Integer(st[i]) );
    }
    return(v);
}
```

- Now we get a **Vector** of the bit positions that are on

Some Things to Note

- The code does not consider the fact that it will be a web service
- This means you can do testing on the command line (or via your IDE)
- A single web service (e.g., fingerprints) can really be multiple web services via polymorphism
 - smiles
 - smiles, no. bits
 - smiles, no. bits, depth

Installing the Service

- Compile the service code
- Copy the class file to the web app `classes` directory
- Create a *web service deployment descriptor*

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
             xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="CDKws" provider="java:RPC">
    <parameter name="className" value="CDKws"/>
    <parameter name="allowedMethods" value="getFingerprintString getFingerprintVector />
  </service>
  <service name="CDKdesc" provider="java:RPC">
    <parameter name="className" value="CDKdesc"/>
    <parameter name="allowedMethods" value="getDescriptors"/>
  </service>
</deployment>
```

Installing the Service

- AXIS uses the WSDD to direct SOAP calls
- The part of interest is the `<service>` entry

What the service will be known as to the client

```
<service name="CDKws" provider="java:RPC">  
  <parameter name="className" value="CDKws"/>  
  <parameter name="allowedMethods" value="getFingerprintString  
  getFingerprintVector />  
</service>
```

The Java class that will provide the service

Which methods of the class can be called by the client

Installing the Service

- Save the WSDD to a file (mydeploy.wsdd)
- Make sure the AXIS jars are in the CLASSPATH
- Install the WSDD using AdminClient

```
- java -cp $CLASSPATH org.apache.axis.client.AdminClient \  
-lhttp://localhost:8080/axis/services/AdminService \  
mydeploy.wsdd
```
- All this can be automated if you're using an IDE (Eclipse, IDEA etc.)

Checking to See if it Worked

- Go to the AXIS page we saw before and click on ***View***

Firefox

File Edit View Go Bookmarks Tools Help

http://office/axis/servlet/AxisServlet

Accounts Usenet Zaplets Slashdot Fark PG CAChe HomePage Gmail

Apache2, Axis & Tomcat 5 http://office/ax...let/AxisServlet

And now... Some Services

- CDKstruct3D ([wsdl](#))
 - get3DCoordinates
- AdminService ([wsdl](#))
 - AdminService
- Version ([wsdl](#))
 - getVersion
- CDKdesc ([wsdl](#))
 - getDescriptors
- CDKsim ([wsdl](#))
 - getSimMatrix
 - getSim
- CDKsdg ([wsdl](#))
 - getSDG
- CDKws ([wsdl](#))
 - getTPSA
 - getMolecularWeight
 - getMolecularFormula
 - getHTMLMolecularFormula
 - getFingerprintString
 - getFingerprintString
 - getFingerprintString
 - getFingerprintVector
 - getFingerprintVector
 - getFingerprintVector
- CDKsima ([wsdl](#))
 - getSim

Done

Adblock

• Each bullet is a service

• Individual services provide one or more methods

• Note that there are multiple methods with the same name

• These are polymorphic versions of the same method

Writing a WS Client

Requirements

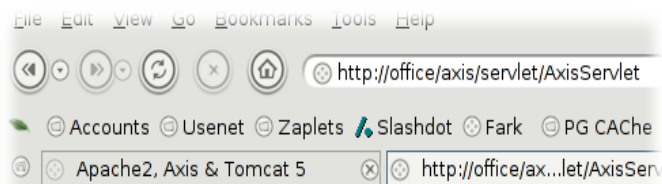
- If all we are going to do is call the service, we only need the **AXIS** libraries in the `CLASSPATH`
- So no need to fiddle with the **CDK** `CLASSPATH`
- However,
 - the client needs to know where the service is
 - what methods can be called
 - what is the input to and output from the methods
- It's nice if the author of the service lets us know
- Otherwise we can use **WSDL**

Web Service Description Language

- WSDL is an XML-based description of a web service
- It describes
 - what type of objects it accepts and returns and what type of exceptions can occur amongst other things
- This is what makes a WS independent of language
- What do we do with WSDL?
 - Look at it and write a client
 - Generate a *proxy* and use that in the client

Where Does the WSDL Come From?

- We don't have to generate WSDL
- The server will generate it for us
- But we still need to know where to find it
- *Service discovery* is not fully solved
- One approach is UDDI
- The other is to ask around



And now... Some Services

- CDKstruct3D ([wsdl](#))
 - get3DCoordinates
- AdminService ([wsdl](#))
 - AdminService
- Version ([wsdl](#))
 - getVersion
- CDKdesc ([wsdl](#))
 - getDescriptors
- CDKsim ([wsdl](#))
 - getSimMatrix
 - getSim
- CDKsdg ([wsdl](#))
 - getSDG
- CDKws ([wsdl](#))
 - getTPSA
 - getMolecularWeight
 - getMolecularFormula
 - getHTMLMolecularFormula
 - getFingerprintString
 - getFingerprintString
 - getFingerprintString
 - getFingerprintString
 - getFingerprintVector
 - getFingerprintVector
 - getFingerprintVector
- CDKsima ([wsdl](#))
 - getSim

What Does WSDL Look Like?

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://office:8080/axis/services/CDKstruct3D" xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://office:8080/axis/services/CDKstruct3D" xmlns:intf="http://office:8080/axis/services/CDKstruct3D"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns1="http://exception.cdk.openscience.org"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.2RC2
Built on Nov 16, 2004 (12:19:44 EST)-->
<wsdl:types>
<schema targetNamespace="http://exception.cdk.openscience.org" xmlns="http://www.w3.org/2001/XMLSchema">
<import namespace="http://schemas.xmlsoap.org/soap/encoding"/>
<complexType name="CDKException">
<sequence/>
</complexType>
</schema>
</wsdl:types>
<wsdl:message name="CDKException">
<wsdl:part name="fault" type="tns1:CDKException"/>
</wsdl:message>
<wsdl:message name="get3DCoordinatesResponse">
<wsdl:part name="get3DCoordinatesReturn" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="get3DCoordinatesRequest">
<wsdl:part name="in0" type="soapenc:string"/>
<wsdl:part name="in1" type="soapenc:string"/>
</wsdl:message>
<wsdl:portType name="CDKstruct3D">
<wsdl:operation name="get3DCoordinates" parameterOrder="in0 in1">
<wsdl:input message="impl:get3DCoordinatesRequest" name="get3DCoordinatesRequest"/>
<wsdl:output message="impl:get3DCoordinatesResponse" name="get3DCoordinatesResponse"/>
<wsdl:fault message="impl:CDKException" name="CDKException"/>
</wsdl:operation>
</wsdl:portType>
```

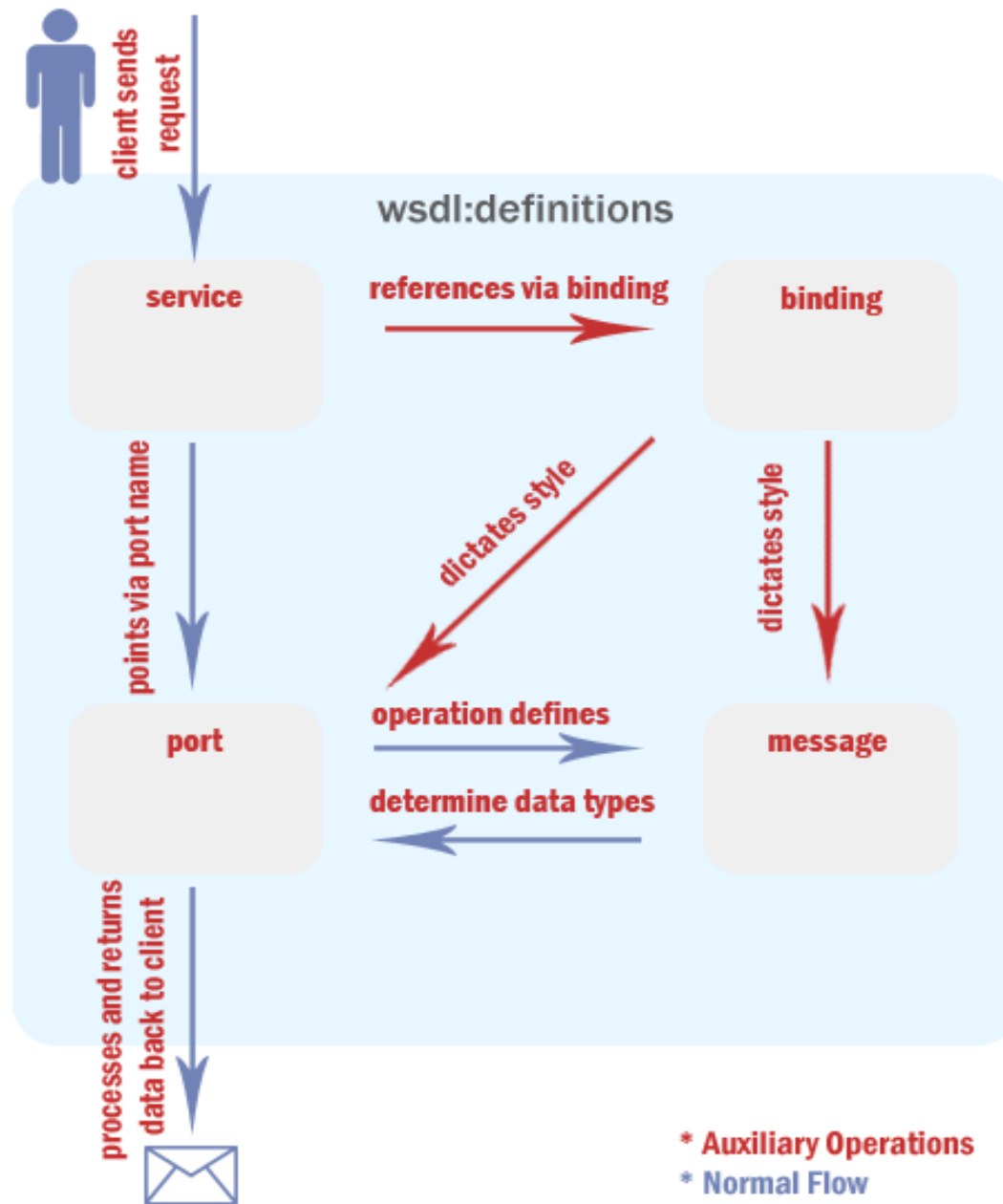
My eyes hurt!

....

The Components of WSDL

- **Types**
 - Used to define custom message types
- **Messages**
 - Abstraction of request and response messages that my client and service need to communicate.
- **PortTypes**
 - Contains a set of operations.
 - Operations organize WSDL messages.
 - Operation->method name, PortType->Java interface
- **Bindings**
 - Binds the PortType to a specific protocol (typically SOAP over http).
- **Services**
 - Gives you one or more URLs for the service.
 - Let's us know where to go to execute 'CDKstruct3D'

The Components of WSDL



Getting Something Out of WSDL

Service

```
<wsdl:service name="CDKstruct3DService">  
  <wsdl:port binding="impl:CDKstruct3DSoapBinding" name="CDKstruct3D">  
    <wsdlsoap:address location="http://office:8080/axis/services/CDKstruct3D"/>  
  </wsdl:port>  
</wsdl:service>
```

Indicates what protocol we'll use

The location of the service

Binding

```
<wsdl:binding name="CDKstruct3DSoapBinding" type="impl:CDKstruct3D">  
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>  
  <wsdl:operation name="get3DCoordinates">  
    <wsdlsoap:operation soapAction=""/>  
    <wsdl:input name="get3DCoordinatesRequest">  
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"  
        namespace="http://DefaultNamespace" use="encoded"/>  
    </wsdl:input>  
    <wsdl:output name="get3DCoordinatesResponse">  
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"  
        namespace="http://office:8080/axis/services/CDKstruct3D" use="encoded"/>  
    </wsdl:output>  
    <wsdl:fault name="CDKException">  
      <wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"  
        name="CDKException"  
          namespace="http://office:8080/axis/services/CDKstruct3D"  
          use="encoded"/>  
    </wsdl:fault>  
  </wsdl:operation>
```

The protocol is SOAP over HTTP

Description of the input

Description of the output

Getting Something Out of WSDL

PortType

A message which describes how a client will communicate

Two parameters in specified order

```
<wsdl:portType name="CDKstruct3D">
  <wsdl:operation name="get3DCoordinates" parameterOrder="in0 in1">
    <wsdl:input message="impl:get3DCoordinatesRequest" name="get3DCoordinatesRequest"/>
    <wsdl:output message="impl:get3DCoordinatesResponse"
      name="get3DCoordinatesResponse"/>
    <wsdl:fault message="impl:CDKException" name="CDKException"/>
  </wsdl:operation>
```

Messages

```
<wsdl:message name="CDKException">
  <wsdl:part name="fault" type="tns1:CDKException"/>
</wsdl:message>
<wsdl:message name="get3DCoordinatesResponse">
  <wsdl:part name="get3DCoordinatesReturn" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="get3DCoordinatesRequest">
  <wsdl:part name="in0" type="soapenc:string"/>
  <wsdl:part name="in1" type="soapenc:string"/>
</wsdl:message>
```

A description of the exception that can be thrown

The type of the input and their order

What is the Result of This?

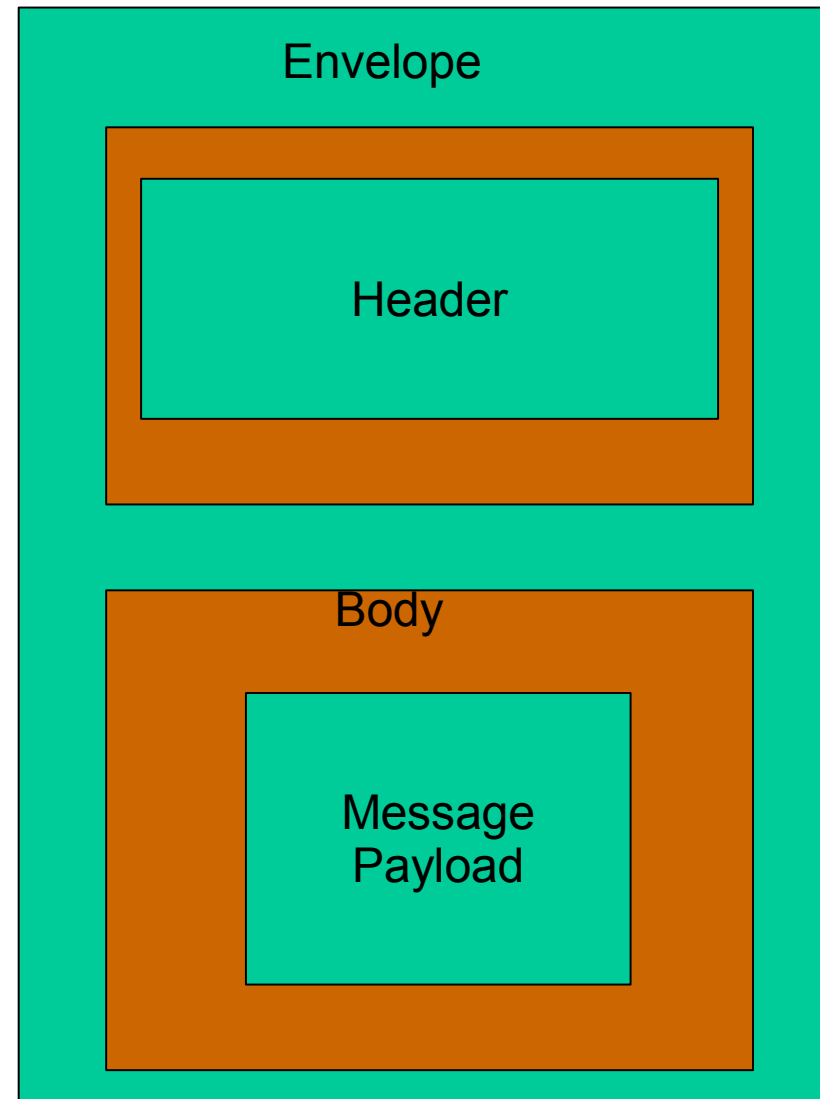
- Examining the WSDL gives us the following information about the CDKstruct3D service
 - Location of the service
 - It takes 2 inputs, which will be SOAP encoded strings
 - It returns a SOAP encoded string
 - It may return a fault of type **CDKException**
- But we need to know what the arguments mean
 - Got to ask the author!

Simple Object Access Protocol

- SOAP is an XML message format (encoding)
 - Describes how an object is represented when sent to or received from a service
- SOAP messages have to be **sent** to the service
 - Usually over HTTP
- SOAP is usually linked to WSDL
- What does a SOAP message contain?
 - Headers
 - Payload – arbitrary XML

A SOAP Message

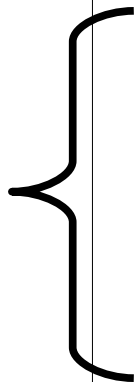
- SOAP structure is very simple.
 - 0 or 1 header elements
 - 1 body element
 - Envelop that wraps it all.
- Body contains XML payload.
- Headers are structured the same way.
 - Can contain additional payloads of **meta-data**
 - Security information, quality of service, etc.



What Does SOAP Look Like?

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  SOAP-
  ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
  >
  <SOAP-ENV:Body>
    <ns1:getFingerprintString
      xmlns:ns1="http://localhost:8080/axis/services/CDKws"
      SOAP-ENC:root="1">
      <v1 xsi:type="xsd:string">CC=COC</v1>
      <v2 xsi:type="xsd:int">1024</v2>
      <v3 xsi:type="xsd:int">6</v3>
    </ns1:getFingerprintString>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Payload



What Types Does SOAP Allow?

- SOAP specifies a number of primitive built in types
 - int, double, float, boolean, String, Date, arrays etc.
- Some compound types also allowed
 - arrays, hashes etc
- So when making a request using SOAP we need to convert objects to a proper SOAP type
 - The required type is described in the WSDL
- Luckily we don't need to deal with raw SOAP messages
 - The SOAP toolkit (AXIS for us) will handle it

The Client Code

- Initialization

Set up the service location

```
public CDKwsClient(String host) {  
    String endpoint = "http://localhost:8080/axis/services/CDKws";  
    try {  
        Service service = new Service();  
        call = (Call) service.createCall();  
        call.setTargetEndpointAddress( new java.net.URL(endpoint) );  
    } catch (Throwable t) {  
        // handle the exception  
    }  
}
```

Initialize the service

The Client Code

- Getting a fingerprint as a **String** object

Setup the input parameter, a single SMILES string

The method we will call

```
public String getFPString(String[] args) {
    try {
        call.removeAllParameters();
        call.setOperationName( "getFingerprintString" );
        call.addParameter( "s", XMLType.XSD_STRING, ParameterMode.IN);
        call.setReturnType( XMLType.XSD_STRING );
        String ret = (String)call.invoke( new Object[] { args[0] } );
        System.out.println(ret);
        return ret;
    } catch (Throwable t) {
        // handle the error
    }
}
```

Setup the return type

Perform the call

Print out what we got

The Client Code

- Getting a fingerprint as a **Vector** object

The method we will call

```
public Vector getFPVector(String[] args) {
    try {
        call.removeAllParameters();
        call.setOperationName( "getFingerprintVector" );
        call.addParameter( "s", XMLType.XSD_STRING, ParameterMode.IN);
        call.setReturnType( new QName("Vector"), Vector.class );
        Vector ret = (Vector)call.invoke( new Object[] { args[0] } );
        Iterator it = ret.iterator();
        while (it.hasNext()) {
            System.out.println( it.next() );
        }
        return ret;
    } catch (Throwable t) {
        // handle the error
    }
}
```

We setup
the return
type as a
Vector

Print the
elements
of the Vector

Isn't This Too Much Work?

- Yes!
- Let AXIS generate Java classes from the WSDL

```
java -cp $CLASSPATH org.apache.axis.wsdl.WSDL2Java \  
-p javaworld.axis.proxy \  
http://localhost:8080/axis/services/CDKws?wsdl
```

- This will create a set of classes that correspond to the WSDL components
- So now the client code will be ...

Using the Proxy Class

```
public Vector getFPVector(String[] args) throws Exception {  
    CDKwsServiceLocator service = new CDKwsServiceLocator();  
    CDKws port = service.getPort();  
    Vector ret = port.getFingerprintVector(args[0]);  
    return ret  
}
```

- The *AXIS* generated code creates a *proxy*
- Rather than dealing with *SOAP* we now deal with the *WS* in terms of *WSDL*
- Essentially we treat the service as if it were a local object

Other Clients – The Browser

- You can directly access a service using your browser

<http://localhost:8080/axis/services/CDKws?method=getFingerprintString&s=CC=CCOC>

- And you get
- Which is useful for checking but not much else
- Can't pass anything but strings

```
-<soapenv:Envelope>  
  -<soapenv:Body>  
    -<getFingerprintStringResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">  
      -<getFingerprintStringReturn xsi:type="soapenc:string">  
        {75, 284, 323, 476, 500, 540, 544, 588, 633, 637, 741, 742, 831, 846}  
      </getFingerprintStringReturn>  
    </getFingerprintStringResponse>  
  </soapenv:Body>  
</soapenv:Envelope>
```


Other Clients – The Web Page

- It's handy to have a web page that provides a form interface
- Depends on whether the WS is useful as a web page
- Very easy with the PHP

Bookmarks Tools Help

http://cheminfo.informatics.indiana.edu/~rguha/code/java/cdkws.html

Openet Zaplets Slashdot Fark PG CACHe HomePage Gmail

es/CDKsdg?wsdl CDK Web Services

You can also get the tox class for a SMILES string by doing

```
http://blue.chem.psu.edu/tox/services/toxTree?method=getCramerClass&smiles=c1ccccc1
```

[Service code]

Topological Polar Surface Area

The CDK includes a molecular descriptor to calculate the Topological Polar Surface Area [1]. Since it's generally useful I made it available as a service. The code will add hydrogens to satisfy valencies and will also perform aromaticity detection.

SMILES Get TPSA

You can also get a value of the TPSA for a SMILES string by doing

```
http://blue.chem.psu.edu/axis/services/CDKws?method=getTPSA&sm=C=O
```

[Service code & Client code]

Structure Diagrams

In many cases it is useful to obtain a 2D representation of a molecular structure. This web service takes a SMILES string and returns a JPEG image of the structure using the StructureDiagramGenerator of the CDK. The call to the service can specify the width, height and scaling factor for the image though by setting all of them to zero default values are used.

A command line client to access this service can be used as

```
java -cp $CLASSPATH:./ CDKsdgClient CC=OC http://blue.chem.psu.edu/axis/services/CDKsdg
```

Note that you can't specify the image width and height, but it's a trivial change to the source code. The program will generate a file called img.jpg in the current working directory. You can also access the service from the form provided below.

SMILES Get 2D Structure Diagram

Width

Height

Scale (Between 0 and 1)

[Service code & Client code]

Molecular Weight & Formula

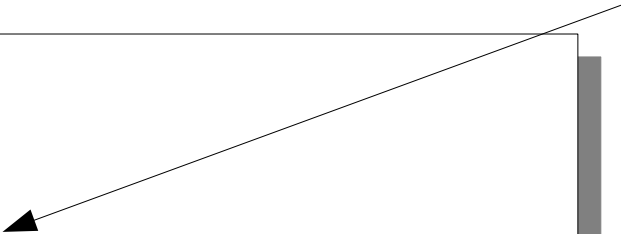
Since the CDK can generate molecular formulae in plain text format as well as HTML-sized text format I've put up 3 services that calculate the m

Other Clients – The Web Page


```
<?php
require_once('SOAP/WSDL.php');
require_once('SOAP/Client.php');
function CallTPSAService($t) {
    $client = new
        SOAP_Client('http://localhost:8080/axis/services/CDKws');
    $params = array('smiles'=>trim($t));
    $soapoptions = array('timeout' => 360);
    $tpsa = $client->call('getTPSA',$params,$soapoptions);

    if (PEAR::isError($ret)) {
        // handle error
    } else {
        // make a nice web page
    }
}
$smiles = $_POST['smiles']
CallTPSAService($smiles)
?>
```

The PHP code
called by the
form



The HTML
form



```
<form action="phptpsa.php" method="post" enctype=
    "multipart/form-data">
<table><tr>
    <td><b>SMILES</b></td>
    <td><input name="smiles" type="text" /></td>
    <td>
        <input type="submit" value="Get TPSA" />
    </td>
</tr></table>
</form>
```

Python Web Services

- Python has a number of ways to deal with WS's
 - SOAPpy – simple to use
 - ZSI – more comprehensive, learning curve
- Python handling of SOAP is not as well developed as for Java
- But much easier to write and maintain

A Python Client

- Client code to use the fingerprint service

```
import SOAPpy

if __name__ == '__main__':

    service = SOAPpy.WSDL.Proxy( \
        'http://localhost:8080/axis/services/CDKws?wsdl')
    ret = service.getFingerprintString('CC=COC')
    print ret
```

- One would expect that calling the **Vector** version of the web service would return a *list*
- SOAPpy 0.12.0 & ZSI 1.7.0 throw an exception

More Python Client Fun

- Very easy to get available methods and their parameter names and datatypes

```
>>> from SOAPpy import WSDL
>>> service = WSDL.Proxy('http://localhost:8080/axis/services/CDKws?wsdl')
>>> service.methods.keys()
[u'getMolecularWeight', u'getMolecularFormula', u'getFingerprintVector', u'getFingerprintString',
u'getHTMLMolecularFormula', u'getTPSA']
>>> fpMethod = service.methods['getFingerprintVector']
>>> fpMethod.inparams[0].name
u'in0'
>>> fpMethod.inparams[0].type
(u'http://schemas.xmlsoap.org/soap/encoding/', u'string')
>>>
>>> fpMethod.outparams[0].name
u'getFingerprintVectorReturn'
>>> fpMethod.outparams[0].type
(u'http://xml.apache.org/xml-soap', u'Vector')
```

A Python Server

- A simple fingerprint service in Python takes 7 lines
- We use the **Frowns** toolkit to provide fingerprint functionality
- Start this in a terminal

```
import frowns.Fingerprint
from frowns import Smiles

import SOAPpy

def getFP(smiles):
    mol = Smiles.smilin(smiles)
    fp = frowns.Fingerprint.generateFingerprint(mol)
    return fp

if __name__ == '__main__':
    server = SOAPpy.SOAPServer(('localhost', 8888))
    server.registerFunction(getFP)
    server.serve_forever()
```

A Client for the Server

- As before the client is pretty trivial

- Run this in another terminal

```
import SOAPpy

if __name__ == '__main__':
    remote = SOAPpy.SOAPProxy("http://localhost:8888")
    fp = remote.getFP('CC(CCOCC)C=C(CC)OC')
    print fp
```

- Python analogs to Tomcat include Plone & Zope

Advanced Issues

Can We Use Complex Types?

- Just passing **String**, **int**, **float** objects to WS's can be limiting
- SOAP allows us to pass some complex objects such as arrays & structs
- We can pass more complex objects if we specify them in a **schema**
 - AXIS provides implementations for some collection classes specified by the SOAP schema :
Vector, Enumeration, Hashtable, Map

Can We Use Complex Types?

- The type definition describes how an object will be **serialized** and **deserialized**



- A general Java class does not serialize to XML
- But JavaBeans do serialize.
 - A bean is a class with accessor (get/set) methods associated with each of its data types.
- XML Beans and Castor are two Java-to-XML converters.
- AXIS supports both

Complex Types - Serialization

- Java class

```
class DescriptorBean {  
    String name="TPSA";  
    float value = 51.453;  
    public String getName() {return name;}  
    public void setName(String n) {name=n;}  
    public String getValue() {return value;}  
    public void setValue(float v) {value=v;}  
}
```

- Possible SOAP encoding

```
<DescriptorBean>  
    <name xsi:type="xsd:string">TPSA</name>  
    <value xsi:type="xsd:float">51.453</value>  
</DescriptorBean>
```

- You can also write your own serializer - deserializer

Asynchronous Services

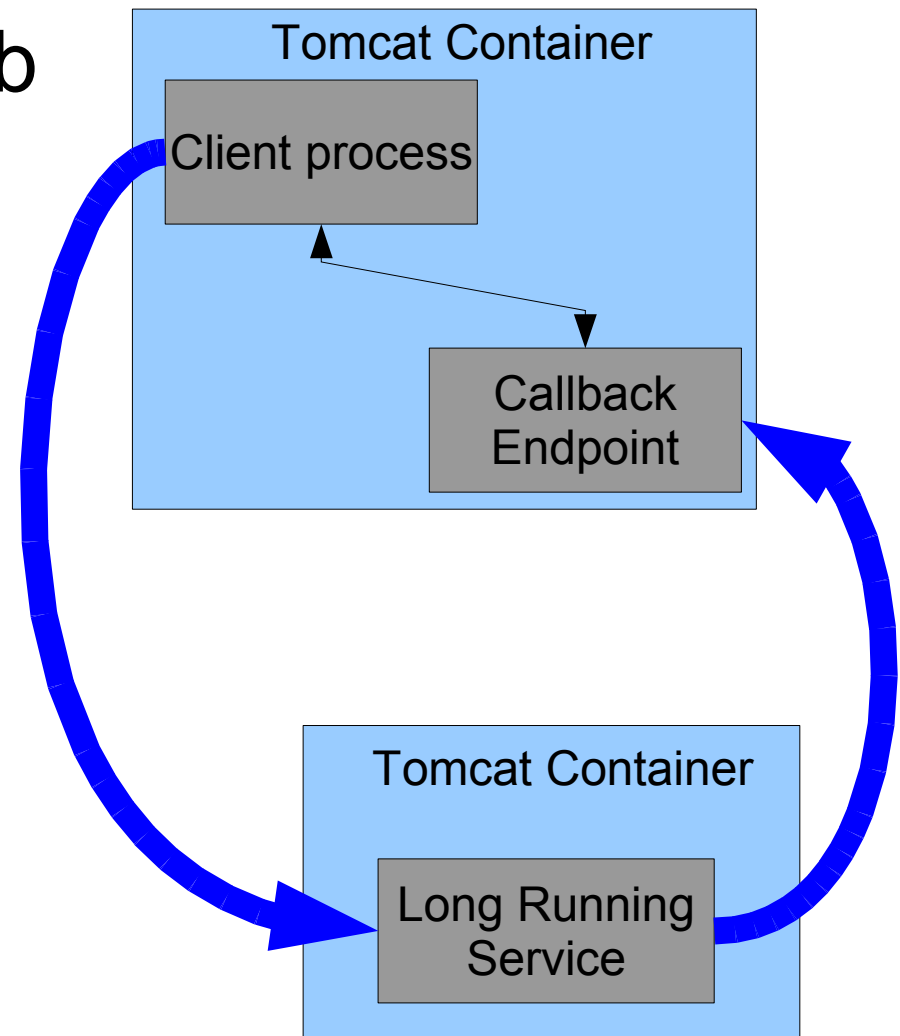
- What if the service takes a long time to run?
- Ideally, the service will be *non-blocking*
 - Client sends the request and does not wait for an answer.
- How does it know when the job is done?
- Two ways to achieve this behavior
 - **Polling** – client periodically checks with the service to see if it's done
 - **Callbacks** – a method provided by the client, called by the service to let it know that the job is done

Asynchronous Services

- These services lead to an important distinction
 - Uptil now we have considered WS's as function calls. This is the **RPC** view of web services
 - Polling and callbacks represent a **Message** based view of web services
- Since requests & responses (i.e., messages) are no longer in-order we need a way to *correlate* them
 - Include unique ID's in the SOAP header
 - Java Messaging Services (JMS) provides a way to do this via JMS headers

Asynchronous Services

- How does the callback approach work?
- The callback can be a web service located in
 - the server container
 - the client container
 - within the client
- The service just needs to know where it is

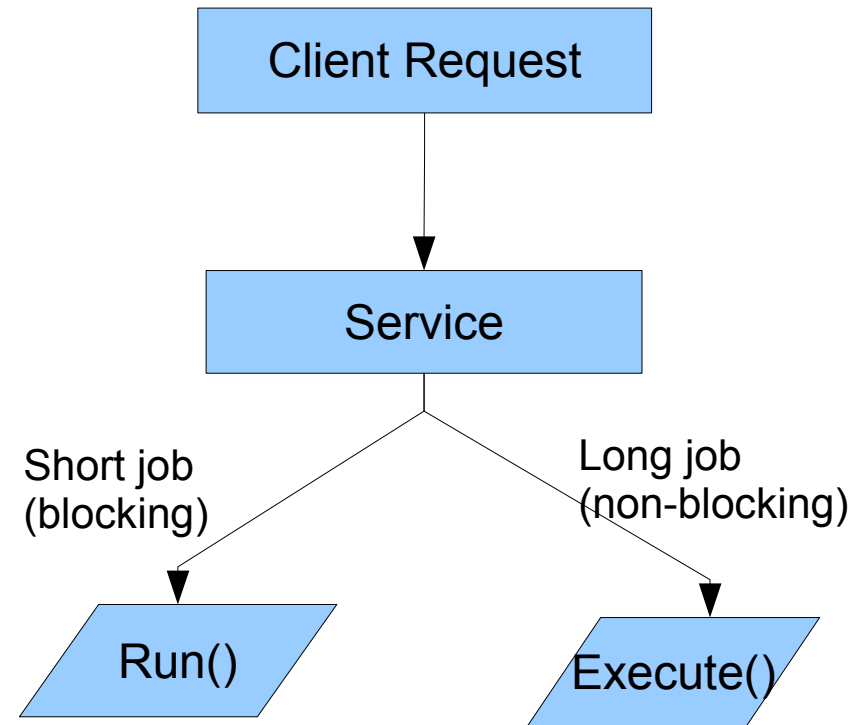


Using Code You Didn't Write?

- So far we have been writing the services
- What if you have pre-existing code?
 - Hack it into a service – too much work, might not have the sources
 - Run it directly
- Running external code
 - Tomcat security can make running arbitrary executables, via **exec()**, tricky
 - Ant provides a way around this

Ant Based Services

- Ant is a build system for Java programs
- It can execute arbitrary programs via the **<execute>** task
- We have two versions
 - one for short jobs (sec's)
 - one for long jobs (hours, days)
- The status of an **Execute()** can be queried via callbacks using a *Context* web service



Ant Based Web Services

- We currently have some services that are based on Ant
 - BCI clustering & fingerprints
 - ToxTree toxicity predictions
- BCI is compiled C code, ToxTree is a Java jar file
- Clearly, any executable can be made into a web service
- **Caveat** – make sure the GUI is separate from the core code

Handling Binary Data

- Sometimes you will return binary data
 - The CDKsdg service returns a 2D structure diagram as a stream of bytes representing the JPEG image
- Can be handled in two ways
 - Use attachments (more elegant)
 - SOAP with Attachments – implemented in SAAJ
 - WS-Attachments
 - Base64 encoding (crude)
 - Expands data size by 1.33x
 - Overhead in processing (encoding & decoding)
 - Easy to do!

Handling Binary Data

- Service code to return an image

```
Graphics snapGraphics = img.getGraphics();
paint(snapGraphics);
RenderedOp image = JAI.create("AWTImage", img);

JPEGEncodeParam params = new JPEGEncodeParam();
baos = new ByteArrayOutputStream();
ImageEncoder encoder = ImageCodec.createImageEncoder(
    "JPEG", baos, params);
encoder.encode(image);
baos.close();

...
Base64 converter = new Base64();
return converter.encode(baos.toByteArray());
```

- The client can (after decoding)
 - Dump it to a file and link to it
 - Stream it to the browser which will show the image

Large Amounts of Data

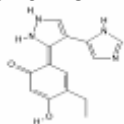
- What if we want to process 10K SD files?
- Sending it over the network can be slow
- Better solution is to send a URI which could be
 - http://
 - ftp://
 - [file://](#)
- Similarly, if we need to return large amounts of data, return the URI rather than the data
- This approach requires infrastructure to be present

Using WS in a Workflow

- Workflows provide a plug-n-play approach to performing scientific tasks
- Join atomic operations in a GUI to get some result
 - Read SMILES
 - Filter molecule
 - Dock molecule
 - Output energy
 - Rank ligands

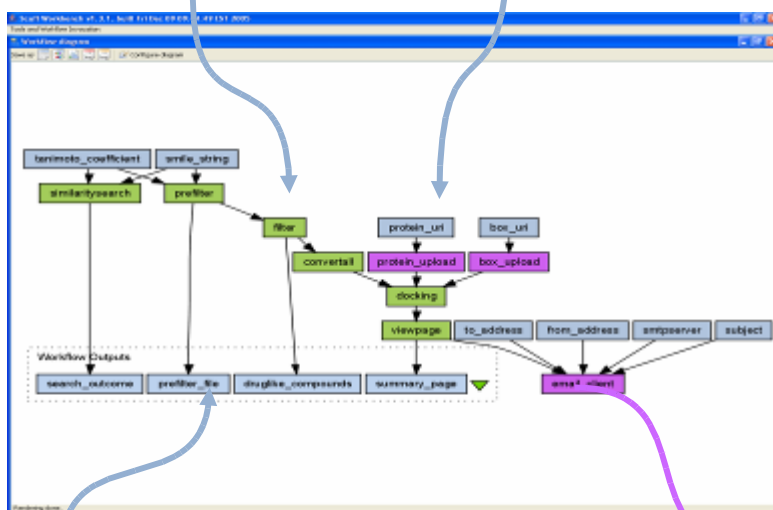
A Taverna Workflow

A 2D structure is supplied for input into the similarity search (in this case, the extracted bound ligand from the PDB 1Y4 complex)



A protein implicated in tumor growth is supplied to the docking program (in this case HSP90 taken from the PDB 1Y4 complex)

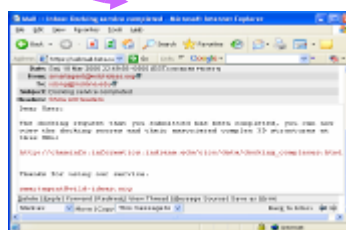
Correlation of docking results and “biological fingerprints” across the human tumor cell lines can help identify potential mechanisms of action of DTP compounds



Once docking is complete, the user visualizes the high-scoring docked structures in a portlet using the Jmol applet.

SMILES	IC50	IC50
	0.0000	0.0000
	0.0000	0.0000
	0.0000	0.0000
	0.0000	0.0000

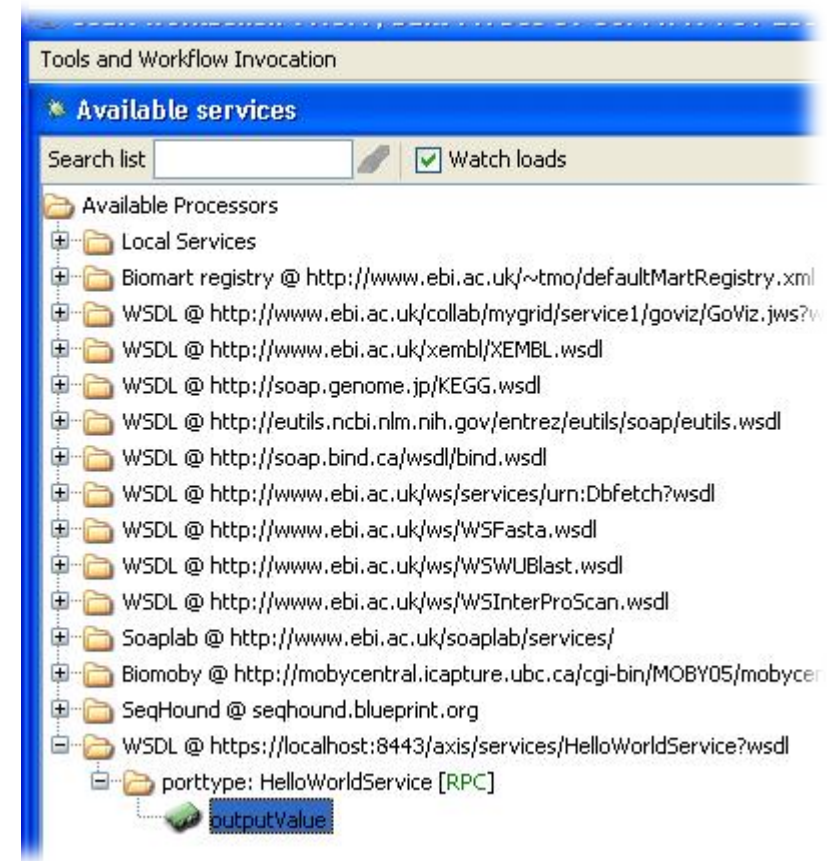
The workflow employs our local NIH DTP database service to search 200,000 compounds tested in human tumor cellular assays for similar structures to the ligand.



Similar structures are filtered for drugability, and are automatically passed to the OpenEye FRED docking program for docking into the target protein.

What If You Don't Have a Component?

- Web services to the rescue!
- Taverna can use WS's
 - You can specify
 - Scavenge them
- Provides an easy interface to WS invocation
- Expands your tool kit



Things I Didn't Cover

- Namespaces
- Security
 - Apache authentication + SSL is a simple way
 - *Secure SOAP messages* – allows for digital signing
- Addressing
- Reliability
- Other SOAP toolkits
 - C/C++ can use [gSOAP](#)
 - Perl can use [soaplite](#)

Summary

- Writing web services is not difficult
 - Setting up your server environment is harder!
- You can write new services or wrap a pre-existing program as a service
- It can be tricky to use non-primitive objects in IO
- Given web services, we can call them using
 - a browser
 - CLI or GUI programs
- We can also aggregate them using workflow tools like Taverna

Links & Downloads

- Apache
 - Web server: <http://httpd.apache.org/>
 - Tomcat : <http://tomcat.apache.org/>
 - AXIS : <http://ws.apache.org/axis> - Java and C++ bindings
- Asynchronous web services
 - [Asynchronous patterns](#)
- SOAP
 - [Specification, Serialization](#)
- CDK
 - [Homepage, Nightly builds](#)

Links & Downloads

- Python
 - [SOAPpy](#)
 - [ZSI](#)
 - [Webservices with Zope & Plone](#)
- C/C++
 - [gSOAP](#)
- Perl
 - [soaplite](#)

Links & Downloads

- IU Web Services
 - http://www.chembiogrid.org/projects/proj_cdk.html
 - http://www.chembiogrid.org/projects/proj_toxtree.html
 - http://www.chembiogrid.org/projects/proj_cambridge.html
- Examples & Howto's
 - CDK Web Services
 - Setting up Apache2, Axis & Tomcat 5
 - Hollow World blog