Tutorial VII. Web Service Deployment and Registry

Web services deployment

Web services registry

Web services discovery

1. Web services deployment

- If you are using TOMCAT/AXIS
  - Refactoring your package name to have a prefix such as "c408h0xx.OmniEditor"
  - cd /u/yijun/axis/c408h0xx
  - Copy your class files to /u/yijun/deployed/c408h0xx
  - Name your deployment file "deploy.wsdd", and "undeploy.wsdd"
  - ../seawolf-deploy.sh, ../seawolf-undeploy.sh (for publish purposes), ask instructor if you can't do it (permission error)
  - For test purposes: ../skywolf-deploy.sh, ../skywolf-undeploy.sh, or ../local.sh
  - Go to http://seawolf.cdf.toronto.edu:8081/axis to verify your web service is deployed

continued

- If you are using gSOAP
  - Modify the web server example to host your own web service
  - Let the server to listen to a port number 8xxx and deploy it to CDF

nohup /u/yijun/software/gsoap-2.7/soapcpp2/samples/webserver 8xxx &

2. Web Services Repository

- UDDI Server: web services registry
  - A UDDI registry contains three kinds of information
    - White pages: Information such as the name, address, telephone number, and other contact information of a given business.
    - Yellow pages: Information that categorizes businesses. This is based on existing (non-electronic) standards
    - Green pages: Technical information about the Web services provided by a given business
  - jUDDI: http://ws.apache.org/juddi
    - a java implementation of UDDI specification
    - can be deployed and accessed through a UDDI browser to browse/search web services
    - jUDDI on CDF: http://skywolf.cdf.toronto.edu:8081/juddi

continued

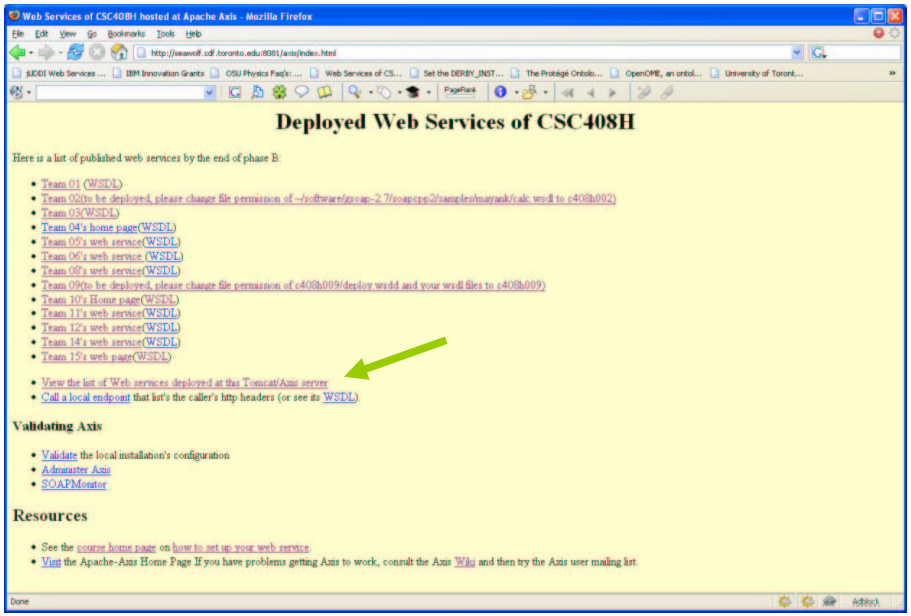
- Inquire web services on UDDI server
  - jUDDI: publisher table in database

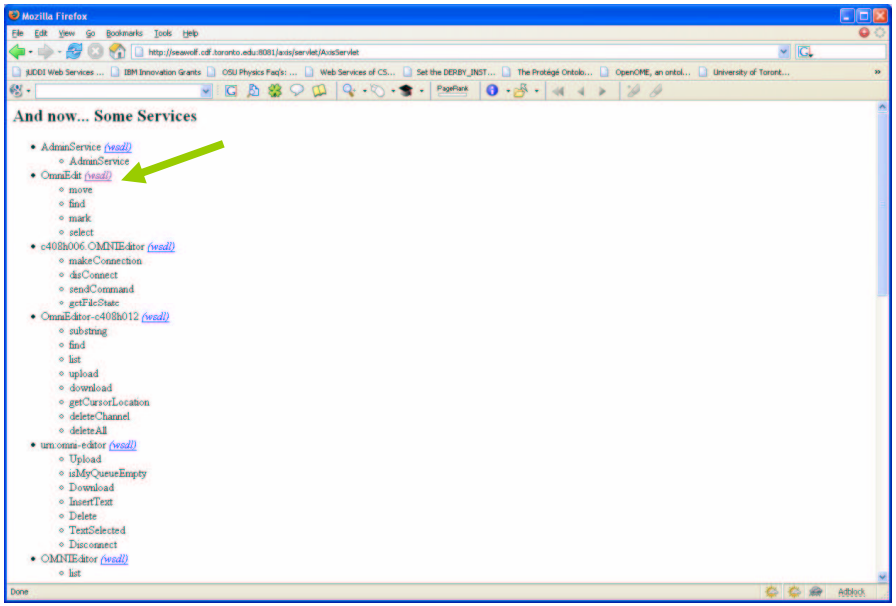
    mysql –h skywolf juddi -e 'select PUBLISHER_ID from PUBLISHER'

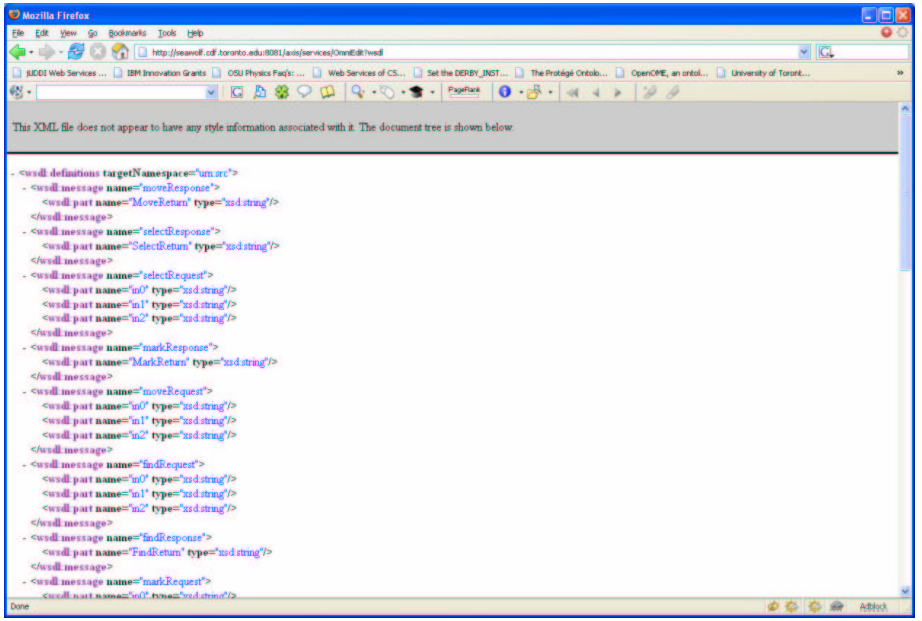
PUBLISHER_ID	PUBLISHER_NAME	LAST_NAME	FIRST_NAME	MIDDLE_INIT	WORK_PHONE	MOBILE_PHONE	PAGER	EMAIL_ADDRESS	ADMIN	ENABLED
jdoe	John Doe	NULL	NULL	NULL	NULL	NULL	NULL	john.doe@apache.org	true	true
c408h001	Team One	NULL	NULL	NULL	NULL	NULL	NULL	c408h001@cdf.toronto.edu	true	true
c408h002	Team Two	NULL	NULL	NULL	NULL	NULL	NULL	c408h002@cdf.toronto.edu	true	true
c408h003	Team Three	NULL	NULL	NULL	NULL	NULL	NULL	c408h003@cdf.toronto.edu	true	true
c408h004	Team Four	NULL	NULL	NULL	NULL	NULL	NULL	c408h004@cdf.toronto.edu	true	true
c408h005	Team Five	NULL	NULL	NULL	NULL	NULL	NULL	c408h005@cdf.toronto.edu	true	true
c408h006	Team Six	NULL	NULL	NULL	NULL	NULL	NULL	c408h006@cdf.toronto.edu	true	true
c408h008	Team Eight	NULL	NULL	NULL	NULL	NULL	NULL	c408h008@cdf.toronto.edu	true	true
c408h009	Team Nine	NULL	NULL	NULL	NULL	NULL	NULL	c408h009@cdf.toronto.edu	true	true
c408h010	Team Ten	NULL	NULL	NULL	NULL	NULL	NULL	c408h010@cdf.toronto.edu	true	true
c408h011	Team 11	NULL	NULL	NULL	NULL	NULL	NULL	c408h011@cdf.toronto.edu	true	true
c408h012	Team 12	NULL	NULL	NULL	NULL	NULL	NULL	c408h012@cdf.toronto.edu	true	true
c408h014	Team 14	NULL	NULL	NULL	NULL	NULL	NULL	c408h014@cdf.toronto.edu	true	true
c408h015	Team 15	NULL	NULL	NULL	NULL	NULL	NULL	c408h015@cdf.toronto.edu	true	true

  - HTML-based deployed web services

    http://seawolf.cdf.toronto.edu:8081/axis







3. Publish and Discover Web Services

- UDDI4J: java implementation of UDDI client side
  - Provide API to interact with UDDI registry to:
    - Publish: How the provider of a Web service registers itself.
    - Find: How an application finds a particular Web service.
    - Bind: How an application connects to, and interacts with, a Web service after it's been found.
  - http://www-124.ibm.com/developerworks/oss/uddi4j/
- Public and private UDDI registry nodes
  - IBM UDDI Test Registry (providing “publish” and “discover”) http://uddi.ibm.com/testregistry
  - Microsoft UDDI Test Registry (providing “publish” and “discover”) http://test.uddi.microsoft.com
- UDDI browser: browse/search web services
  - An example of UDDI browser: http://www.soapclient.com/uddisearch.html

4. Use Web Services

- Find/Discover the WSDLs listed in a specific web registry
- Generate the corresponding client
  - AXIS: wsdl2java
  - gSOAP: wsdl2h
- OmniEditor project:
  - Analyze the variability of hosted web services, write a generic client to use several variants of the OmniEditor, find a way to adapt to the changes

Continued (OmniEditor)

- Supporting variability
  - Compile time variability
    - Macros (compile time)
      - » #define WSDLURL “c408h001.OmniEditor.wsdl”
      - » make –DWSDLURL=c408h001.OmniEditor.wsdl
    - Java annotations
  - Run-time variability: generative programming

```
<config>
    <service name=“c408h001.OmniEditor.wsdl”/>
</config>
wsdlURL = getXPath(“config.xml”, “/config/service/@name”);
```

    - Java properties
      - » wsdlURL = System.getProperty(“wsdlurl”);
      - » java –Dwsdlurl=… client
      - » client.properties

Continued(OmniEditor)

- Pick one of the editors
  - VIM
  - Eclipse
  - Emacs
  - jEdit
- Each web services is integrated by at least one client. A system test may involve two instances of the clients