Dynamic Web Services in Java

- Technology overview: web services in Java
- Dynamic web services with Axis2
- Packaging and deployment options; demo
- Conclusions

Technology overview: web services in Java

- Must have
- Popular frameworks
- Deployment options
- <u>Code 1st or contract 1st</u>
- Annotations or XML
- Static or dynamic

Must have

- POJO endpoints - became a must; integration with Spring -advantage
- WS-I, WS-Addressing - must; WS-Security, WS-Policy, WS-AtomicTransactions - advantage
- MTOM, SOAP attachments, StAX - big advantage
- SOAP 1.1, SOAP 1.2 and REST binding
- Synchronous/asynchronous messages
- HTTP - must; JMS, SMTP, etc. - advantage

Axis and Axis 2

- Popular and well supported
- Exists for Java and C
- Progressing in good direction
- Customizable and extendable
- Worked from a first try
- “No one gets fired for choosing ~~IBM~~ Axis”

XFire and CXF

- New, cool, easy to use
- Good coverage WS-* standards
- Solves problems when Axis fails
- ServiceMix ESB support
- non-XML type bindings, such as JSON and CORBA

JAX-WS

- Official standard
- Annotation spec
- Will work with any modern WS framework
- Understood by JavaEE 5 container

Spring WS

WS-I, contract first, loose coupling

- Makes the Best Practice an Easy Practice
- Powerful mappings

  Depending on payload, SOAP header, XPath
- XML API support

  DOM, SAX, StAX, JDOM, dom4j, XOM, etc.
- Flexible XML Marshalling

  JAXB 1 and 2, Castor, XMLBeans, JiBX, XStream
- Reuses your Spring expertise

JBossWS

- JBoss AS family member
- JSR-109 (Web Services for J2EE)
- JAX-WS frontend
- JBossWS-Native, JBossWS-CXF or JBossWS-Metro backends

Other WS frameworks?

Code 1st or contract 1st

- Code 1st
  - Developer works with familiar concepts, WSDL is created automatically
  - Control on WSDL via annotations and XML configuration files
  - Java interfaces don't change
  - WSDL is unstable

- Contract 1st
  - Easy control what WSDL will look like
  - WSDL does not change - good for remote teams
  - Java interfaces are unstable

Axis2 dynamic web services

- Axis2: packaging and deployment options
- Axis2: dynamic services creation/removal
- Authentication – different options
- Custom handling of SOAP messages

Services creation/removal with Axis

- Dynamic nature of AxisConfiguration
- Creating services - few options
  - services.xml
  - endpoint (implementing class)
  - WSDL and message receivers
- Removing services

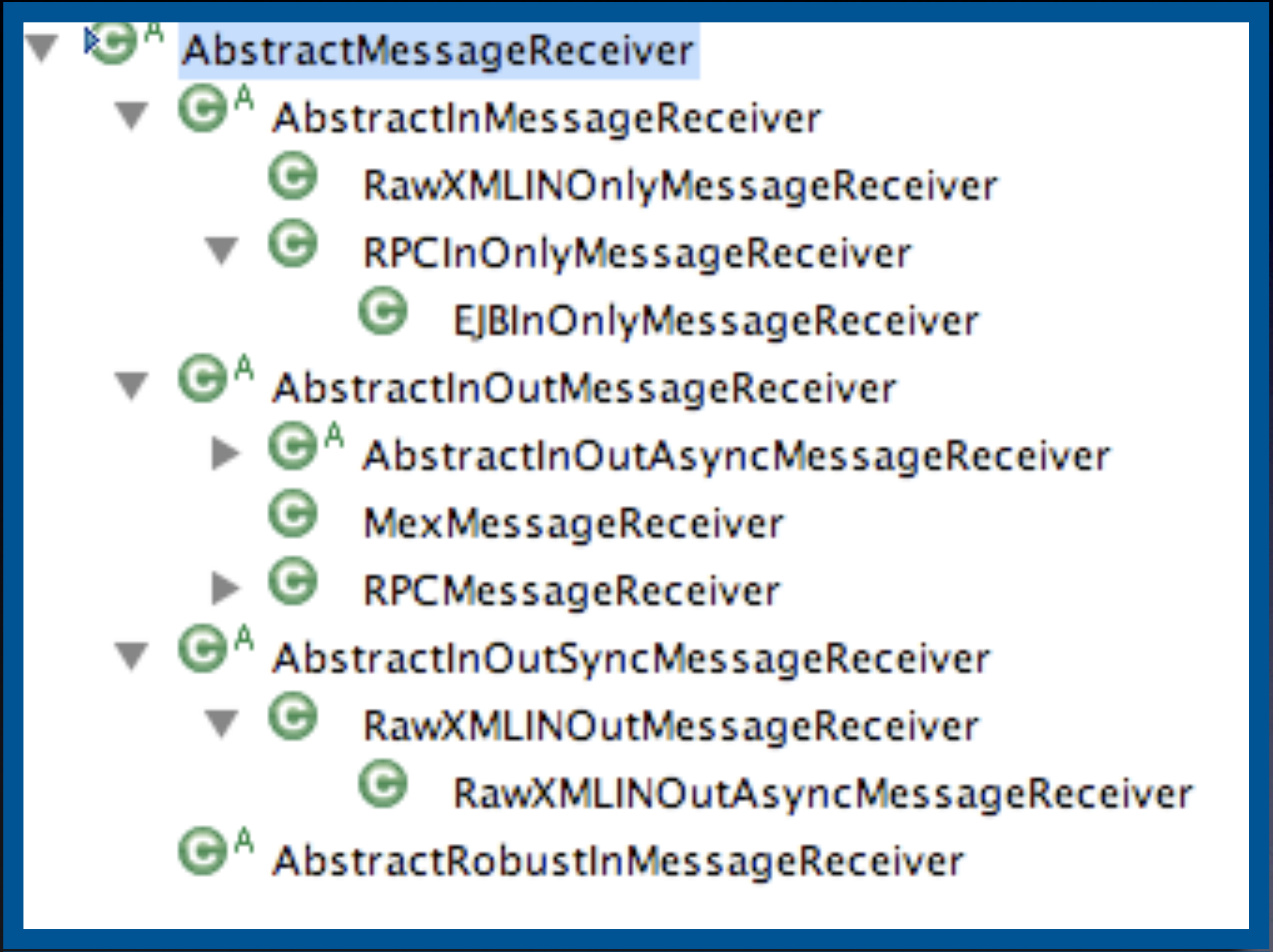
Authentication – different options

- HTTP headers
  - Basic
  - Digest
  - Certificate
- WS-Security
- Application-level

Custom handling of Soap messages

- Axis2 dynamic handler chains
  - First, last, before given handler, after
  - Transport in, pre-dispatch, dispatch, post-dispatch
- Message receiver

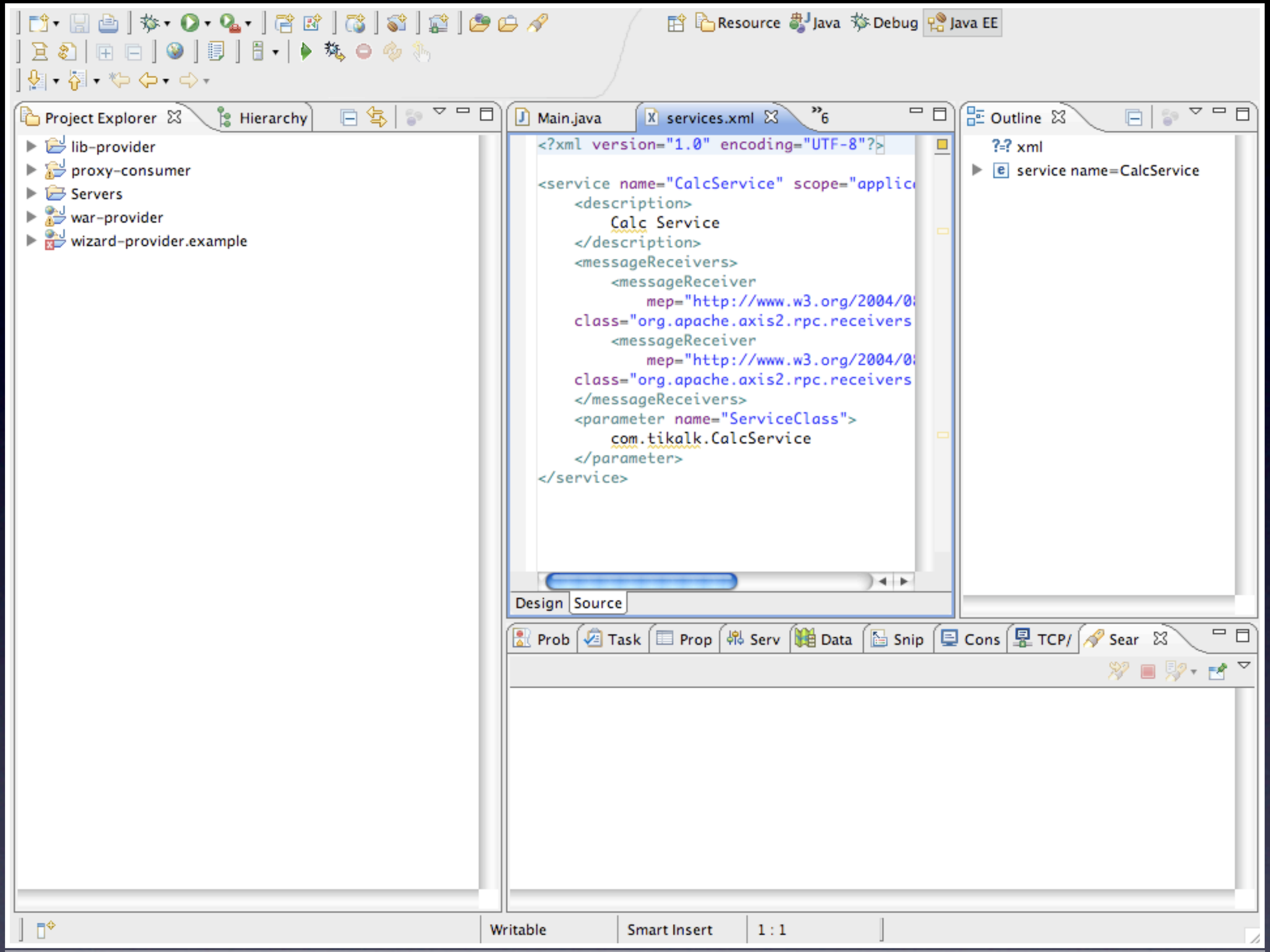
Axis2 message receivers



- AbstractMessageReceiver
  - AbstractInMessageReceiver
    - RawXMLINOnlyMessageReceiver
    - RPCInOnlyMessageReceiver
      - EJBInOnlyMessageReceiver
  - AbstractInOutMessageReceiver
    - AbstractInOutAsyncMessageReceiver
    - MexMessageReceiver
    - RPCMessageReceiver
  - AbstractInOutSyncMessageReceiver
    - RawXMLINOutMessageReceiver
      - RawXMLINOutAsyncMessageReceiver
  - AbstractRobustInMessageReceiver

Axis2: packaging and deployment options

- WS provider is a library, Axis2 is a WAR
- WS provider is a WAR, Axis2 is Eclipse magic
- WS provider is a WAR, Axis2 is a servlet from library - undocumented
- WS consumer - Axis2 is a proxy
- WS consumer - Axis2 is a library



```xml
<?xml version="1.0" encoding="UTF-8"?>

<service name="CalcService" scope="applic
    <description>
        Calc Service
    </description>
    <messageReceivers>
        <messageReceiver
            mep="http://www.w3.org/2004/0
    class="org.apache.axis2.rpc.receivers
        <messageReceiver
            mep="http://www.w3.org/2004/0
    class="org.apache.axis2.rpc.receivers
    </messageReceivers>
    <parameter name="ServiceClass">
        com.tikalk.CalcService
    </parameter>
</service>
```

Demo

Conclusions

- Java WS frameworks are very flexible
- Open source frameworks must be learned by looking into the source code in addition to the documentation
- Dynamic abilities of WS are not used wide enough
- “Every line of code is design” - decisions took by a coder often influence the whole system

Q&A