

SOAP implementation technologies

Oxford University
Software Engineering
Programme
January 2018



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Why use a library?

- WSDL tooling
 - Make it quick and easy
- WS-* extensions
 - WS-Security and related
 - Much less: WS-RM, WS-AT, WS-Addressing



The two major toolkits

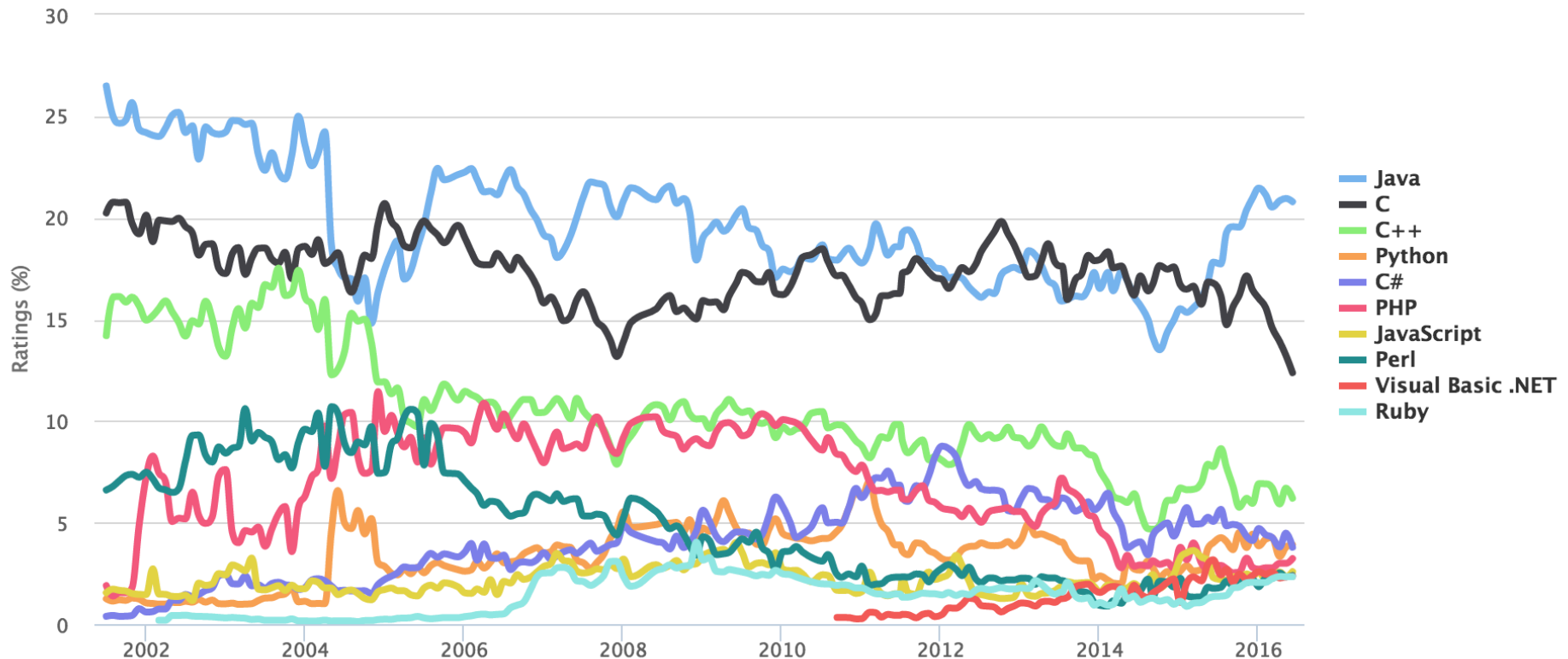
- **.NET WCF**
 - Two implementations
 - Windows
 - Mono
- **JAX-WS**
 - Multiple implementations
 - Sun, CXF, Axis2, etc



Poor support in other languages is one reason for SOAP's decline

TIOBE Programming Community Index

Source: www.tiobe.com



JAX-WS Motivation

- Java API for XML Web Services
 - Currently version 2.2
- Create a standard Java approach to creating and consuming SOAP/WSDL web services
- Based on annotations
- Work with WS-I Basic Profile
- Work with JAX-B (Java API for XML Binding)
- Replaced the (broken) JAX-RPC specification



Two approaches

- **Code first:**
 - Create Java code, annotate
 - Run Java2WS to create WSDL / XSD etc
- **Contract first:**
 - Create (or re-use) WSDL / XSD etc
 - run WSDL2Java to create the Java artefacts



Code first (annotated POJOs)

- Start with a **P**lain **O**ld **J**ava **O**bject
- Create annotations that document the service definition, binding approach, etc



Common Annotations

- @WebService
- @SOAPBinding
- @WebMethod
- @WebParam
- @OneWay
- @HandlerChain



WebService

Applies to class or interface

All parameters are optional

@WebService

```
(name = "OrderService",  
  serviceName = "OrderProcess",  
  portName = "OrderProcessPort",  
  targetNamespace = "http://freo.me/  
order",  
  wsdlLocation="path to existing wsdl")
```



WebService continued

@WebService(endpointInterface =
"me.freo.OrderProcess") applies to class
only

This allows you to create an interface
defining the service/WSDL and a
separate implementation. This is
especially important for WSDL first
operation



SOAPBinding

Applies to class or interface

```
@SOAPBinding(  
    style=SOAPBinding.Style.DOCUMENT,  
    use=SOAPBinding.Use.LITERAL,  
    parameterStyle=  
        SOAPBinding.ParameterStyle.WRAPPED)
```

My hint: ALWAYS use Doc/Lit/Wrapped

see <http://pzf.fremantle.org/2007/05/handlign.html>

Second hint: this is the default so don't use
@SOAPBinding!



WebMethod

Applies to Method

```
@WebMethod(  
    action="MySOAPAction", //optional  
    operationName="myWSDLop",  
    exclude=true) // do NOT expose this  
                // inherited method
```



OneWay

- Applies to a method that is marked `@WebMethod`
- Indicates that there is no response expected
- Assuming this is over HTTP, there should just be a HTTP 202 Accepted response
- Over JMS, no response message expected

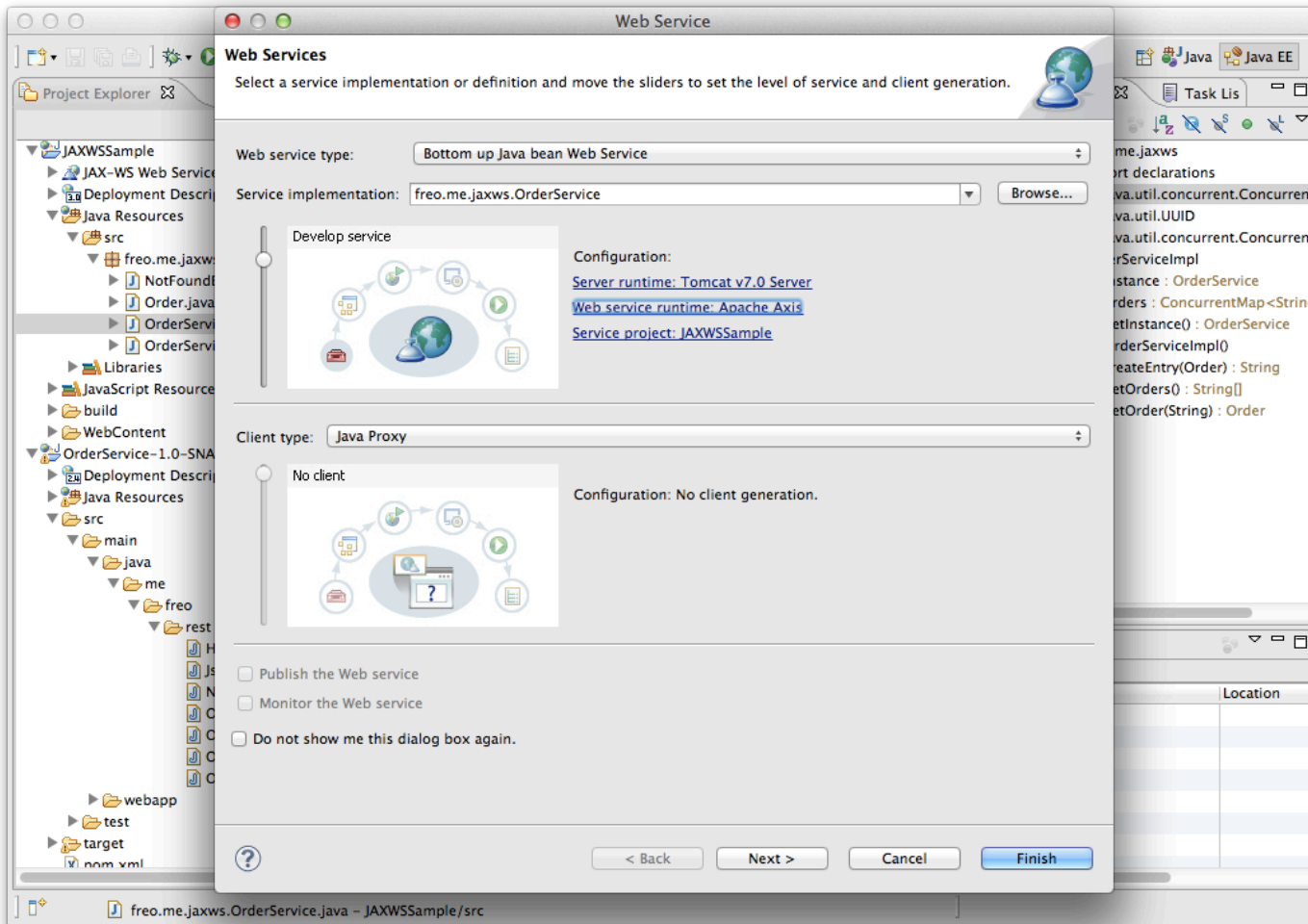


WebParam

- A way of defining the mapping between the XML/SOAP message and the Java Parameters

```
@WebParam(  
    name="nameOfXMLElement",  
    partName="nameOfWSDLPart",  
    targetNamespace="xmlNamespace",  
    mode="IN|OUT|INOUT",  
    header=true|false)
```

Eclipse Web Tools platform

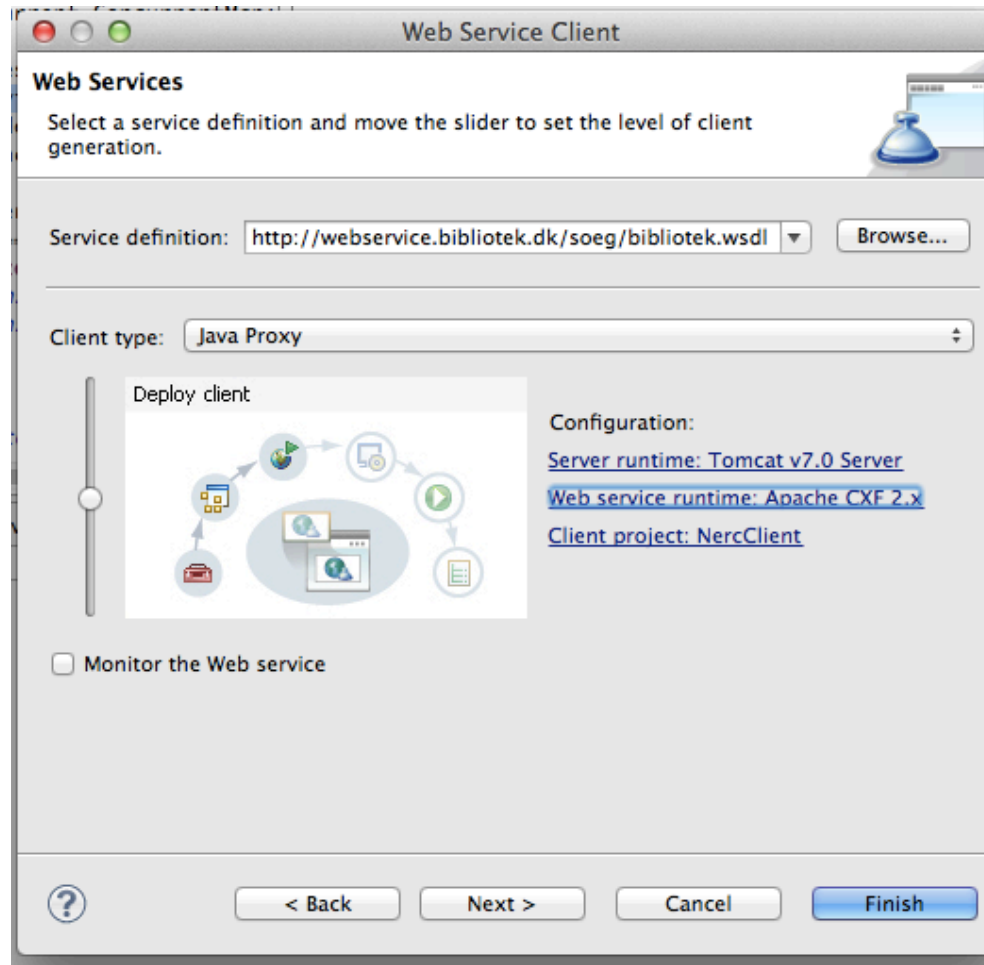


WSDL first

- Again there is a tool for this
- You might want to create a service
 - Contract-first (design the WSDL, then implement)
 - Implement a standard WSDL
 - Re-architect an existing service
 - Copy a competitor's service (though this is a thorny issue!)
- Very likely you need to call a service



Eclipse tooling



Resources

- The Labs
- The Spec
 - <http://jcp.org/aboutJava/communityprocess/mrel/jsr224/index3.html>
- The CXF documentation
 - <http://cxf.apache.org/docs/a-simple-jax-ws-service.html>
- The Reference Implementation
 - <http://jax-ws.java.net/>

