

Index

Space Index

0-9 ... 0	A ... 0	B ... 1	C ... 1	D ... 2	E ... 1
F ... 1	G ... 1	H ... 4	I ... 1	J ... 1	K ... 0
L ... 0	M ... 0	N ... 2	O ... 0	P ... 0	Q ... 1
R ... 2	S ... 1	T ... 1	U ... 1	V ... 0	W ... 4
X ... 2	Y ... 0	Z ... 0	!@#\$... 0		

0-9	A
B Building ActiveSOAP uses Maven as its build tool. If you don't fancy using Maven you can use your IDE directly or Download a distribution or JAR. We have tested the build with Maven 1.0 and it works fine. Later versions of Maven may vary - if you hit problems on o	C Contributing There are many ways you can help make ActiveSOAP a better piece of software - please dive in and help! Try surf the documentation - if somethings confusing or not clear, let us know. Download the code & try it out and see what you think. Browse the source
D Download You can download a jar of ActiveSOAP here Or the full binary and source distributions are here Dynamic Client Instead of generating an interface for a WSDL endpoint as shown in the introductory examples, we can use the dynamic interface to pass in/out any arbitrary XML blob over REST or SOAP. So here is a dynamic web service invocation using XMLBeans as the body	E Example Here is an example of an interface to a service. Note that the interface is optional, you can expose a class as a service - we've just used an interface in this example as its good practice and makes the example easier to follow. {snippet:id=example lang=}
F FAQ	G General General questions about ActiveSOAP
H Home ActiveSOAP is a lightweight & easily embeddable REST and SOAP stack based on StAX with support for WS-Addressing and WSIF. ActiveSOAP uses StAX (the Standard API for pull parsing) to implement the SOAP protocols and then it delegates to plugin Handler obj How do I add SOAP headers on a client invocation You can register a Handler with the SoapClient to perform some processing and output any custom headers as part of a SOAP client invocation. e.g. SoapClient client = ...; Handler myHandler = ...; client.addHeaderHandler(myHandler); response = client.invoke How do I compile from the source See the Building page	I Index {index}{index}

<p>How it works</p> <p>ActiveSOAP is essentially a small, lightweight XML router using the StAX pull parser API. When an XML request is processed, the QName of the root element is extracted and then passed on to some Handler object for processing. In the case of the SOAP protoc</p>	
<p>J</p> <p>JAXB</p> <p>JAXB 2.0 now supports StAX natively and integrates easily into ActiveSOAP. Here is an example application which demonstrates writing a simple POJO based services using purely JAXB and letting ActiveSOAP be the REST or SOAP stack and handling the HTTP or J</p>	<p>K</p>
<p>L</p>	<p>M</p>
<p>N</p> <p>Navigation</p> <p>Overview Home News FAQ JavaDocs Download Using ActiveSOAP Building Example Dynamic Client WS-Addressing How it works Supported Tools JAXB XMLBeans XStream Related Projects ServiceMix ActiveMQ ActiveCluster ActiveSpace Support Issues Roadmap Change log Com</p> <p>News</p>	<p>O</p>
<p>P</p>	<p>Q</p> <p>QuickLinks</p> <p>Download JavaDocs CVS Wiki Mailing Lists IRC IRC Log Support</p>
<p>R</p> <p>REST</p> <p>REST stands for REpresentational State Transfer, and is an architectural style for large-scale software design. Within the scope of the ActiveSOAP project we tend to refer to REST services as being services which consume and emit XML documents using eithe</p> <p>RightHeader</p>	<p>S</p> <p>StAX</p> <p>Streaming API for XML is an event based pull parser API for Java. See JSR 173</p>
<p>T</p> <p>Terminology</p> <p>Questions on acronyms and terminology used</p>	<p>U</p> <p>Using ActiveSOAP</p> <p>Questions on using ActiveSOAP</p>
<p>V</p>	<p>W</p> <p>What is ActiveSOAP</p> <p>ActiveSOAP is a high performance, lightweight, StAX based SOAP framework for document centric Web Services and SOAP intermediaries. It allows a variety of XML marshalling and data binding tools to be dropped in to provide a flexible framework for document</p> <p>What is the licence</p> <p>This software is open source using the Apache 2.0</p>

	<p>licence (a liberal BSD style licence which is very commercial friendly)</p> <p>WS-Addressing</p> <p>You can see the WS-Addressing support in the javadoc. The most common use case is when implementing a server side handler wishing to process the current WS-Addressing context. Here's some example code to show you how to do this</p> <pre>{snippet:id=wsServer lang=j</pre> <p>WSIF</p> <p>The Web Services Invocation Framework provides a generic way to invoke web services along with a way of adorning a WSDL with extra metadata to describe the WSIF binding in more detail. We currently have experimental WSIF providers for XMLBeans and XStream</p>
<p>X</p> <p>XMLBeans</p> <p>XMLBeans is a tool for handling typically XSD typed XML with support for the autogenerating of beans to map to a WSDL or XSD file. We have an XMLBeanHandler in the XMLBeans package you can derive from if you wish to stay close to the metal. Alternatively w</p> <p>XStream</p> <p>We have support for the processing of XML headers or body elements through the use of the XStream library. XStream is idea when you have a bunch of POJOs and you want to stream them over XML. If you are given a WSDL and want to create beans for it, we rec</p>	<p>Y</p>
<p>Z</p>	<p>!@#\$</p>