**POC 2**

1. **Steps for creating a SOAP Web Service by using Apache CXF, Eclipse and Apache Tomcat.**
   1. Create a Dynamic Web Project in the eclipse with Apache CXF by using spring and add Apache Tomcat Server as well.
   2. Use bottom-up approach (creates java class first and then generates the wsdl file) to develop the web service.
   3. Deploy the web service in the Apache Tomcat Server.
   4. We can use Soap UI to test the web service.
2. **Steps for adding WS-Security to the Web Service by using spring and WSS4j.**
   1. Add below configuration in the spring Configuration file (for e.g. “beans.xml”)

<bean id=*"serverPasswordCallback"* class=*"com.crimsonlogic.webservices.poc.callbackHandler.ServicePWCallback"* />

<bean id=*"wss4jInInterceptor"* class=*"org.apache.cxf.ws.security.wss4j.WSS4JInInterceptor"* >

<constructor-arg>

<map>

<entry key=*"action"* value=*"UsernameToken"* />

<entry key=*"passwordType"* value=*"PasswordDigest"* />

<entry key=*"passwordCallbackRef"*>

<ref bean=*"serverPasswordCallback"*/>

</entry>

</map>

</constructor-arg>

</bean>

<jaxws:endpoint xmlns:tns=*"http://services.poc.webservices.crimsonlogic.com/"*

id=*"userservice"* implementor=*"com.crimsonlogic.webservices.poc.services.UserService"*

wsdlLocation=*"wsdl/userservice.wsdl"* endpointName=*"tns:UserServicePort"*

serviceName=*"tns:UserServiceService"* address=*"/UserServicePort"*>

<jaxws:features>

<bean class=*"org.apache.cxf.feature.LoggingFeature"* />

</jaxws:features>

<jaxws:inInterceptors>

<bean class=*"org.apache.cxf.binding.soap.saaj.SAAJInInterceptor"* />

<ref bean=*"wss4jInInterceptor"* />

</jaxws:inInterceptors>

</jaxws:endpoint>

* 1. Write one custom CallbackHandler class by implementing “CallbackHandler” interface.

If the UsernameToken contains a digest password, the CallbackHandler class must return the corresponding password for the given user name (given by “WSPasswordCallback.getIdentifier()”). Verification of the password (by comparing with the digest password) is done by the WSS4J at runtime.

We can override the unimplemented “handle(Callback[] callbacks)” method as given below.

**public** **void** handle(Callback[] callbacks) **throws** IOException,

UnsupportedCallbackException {

WSPasswordCallback pc = (WSPasswordCallback)

callbacks[0];

**if** (pc.getIdentifier() != **null**) {

UserTO uto = dao.getUser(pc.getIdentifier());

pc.setPassword(uto.getPassword());

}

}

1. **Steps for Integrating the Web Service with SymasOpenLdap.**

User below steps to integrate the web service with SymasOpenLdap server.

* 1. Add below jars in the classpath.

spring-ldap-core-1.3.2.CI-20101130.200034-1.jar

spring-orm-3.0.3.RELEASE-sources.jar

spring-tx-3.0.7.RELEASE.jar

spring-jdbc-3.0.3.RELEASE-sources.jar

commons-pool-1.3.jar

* 1. Add the Ldap Spring Configruation file in the classpath.

In case of Spring 3 framework use below contents in the configuration file.

<bean id=*"userDAO"* class=*"com.crimsonlogic.webservices.poc.ldap.dao.UserDAOImpl"*>

<property name=*"ldapTemplate"* ref=*"ldapTemplate"* />

</bean>

<bean id=*"ldapTemplate"* class=*"org.springframework.ldap.core.LdapTemplate"*>

<constructor-arg ref=*"contextSource"* />

</bean>

<bean id=*"contextSource"* class=*"org.springframework.ldap.core.support.LdapContextSource"*>

<property name=*"url"* value=*"ldap://192.168.24.131:389"* />

<property name=*"base"* value=*"cn=user,dc=example,dc=com"*/>

<property name=*"userDn"* value=*"cn=user,dc=example,dc=com"* />

<property name=*"password"* value=*"secret"* />

</bean>

* 1. In custom callbackHandler class we can call the SymasOpenLdap API for getting the corresponding password for the given username.

1. **Sample of SOAP Request and Response**
   1. SOAP Request

<soapenv:Envelope xmlns:ser="http://services.poc.webservices.crimsonlogic.com/" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

<soapenv:Header>

<wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<wsse:UsernameToken wsu:Id="UsernameToken-41">

<wsse:Username>Test4</wsse:Username>

<wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordDigest">auYA4wfizkHp78WfyrGyuJcIa/w=</wsse:Password>

</wsse:UsernameToken></wsse:Security></soapenv:Header>

<soapenv:Body>

<ser:getUser>

<arg0>Suraj</arg0>

<arg1>Kumar</arg1>

</ser:getUser>

</soapenv:Body>

</soapenv:Envelope>

* 1. SOAP Response

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Body>

<ns2:getUserResponse xmlns:ns2="http://services.poc.webservices.crimsonlogic.com/">

<return>SurajKumar</return>

</ns2:getUserResponse>

</soap:Body>

</soap:Envelope>

1. **References**

|  |  |
| --- | --- |
| Installation of “apache-cxf-2.7.2.zip” | <http://cxf.apache.org/download.html> |