Web Services   
--------------------

SOAP is a protocol for exchanging XML based data between server and client over the web and different protocols such as HTTP and SMNP and TCP/IP 🡪 this is called webServcies

is a simple XML-based protocol to let applications exchange information over HTTP.

**Component of Web Services**

**WSDL,UDDI(XML based standard for desrying) and SOAP**

**WSDL** (web service description language)

It is an XML based language to describe web services and how to access them

**SOAP**

SOAP is XML based protocol being used in Web services to exchange information over HTTP

**JAXB**

Mapps Java class to XML.   
Ability to marshal Java object to XML or unMarshal from XML to Java is done by JAXB

**UDDI**

**UDDI** is an XML-based standard for describing, publishing, and finding web services

**XSLT**

It is XML style and transformation - it converts XML to HTML

**JAXP**:

XMP parsing and processing.

**SAX**: 🡪 simple API for large document

It is event based and parse the document one element by one , slower but very good for

large size file

**DOM**:

document object model , it reads the entire XML in to memory and pars it as tree of

object.   
 DOM is very fast for smaller files but might goes slow for larger file.

**JAX-WS**

It is an API to cerate web services

**SOAP handler**

It is a soap message intercepter to manipulate both request and response object

It is an interface

**Authentication**

There are different type and level of security in WS

* Transport level
  + HTTP basic , SSL or Digest
* Message level
  + WS-security

Encryption of WS messages.

Spring has SecurityIntercepter to trap SOAP messages to authenticate

<bean id="wsSecurityInterceptor" class="org.springframework.ws.soap.security.xwss.XwsSecurityInterceptor">

<property name="secureResponse" value="false"/>

<property name="policyConfiguration"

value="classpath:securityPolicy.xml"/>

<property name="callbackHandler">

<bean class="org.springframework.ws.soap.security.xwss.callback.

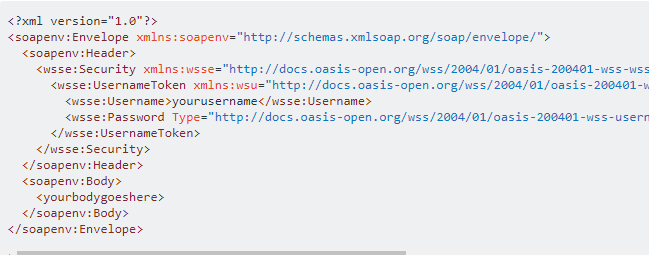
SpringPlainTextPasswordValidationCallbackHandler">

<property name="authenticationManager" ref="securityService"/>

</bean>

</property>

</bean>



public class SecurityService implements AuthenticationManager {

private final Log log = LogFactory.getLog(this.getClass());  
 @Autowired(required=true)  
 WebkinzUserService webkinzUserService;  
 @Override  
 public Authentication authenticate(Authentication auth) throws AuthenticationException {  
 log.info("Start Authentication : ");   
 if (auth != null)

String userName = auth.getName();

if("dummyWebkinzUser".equalsIgnoreCase(userName)) { return auth;

}

String securityToken = auth.getCredentials().toString();

if (webkinzUserService.isWebkinzeStoreUserValid(userName, securityToken)){

log.info("END Authentication : " + userName + "validated");

return auth }else{

throw new BadCredentialsException("Username/password doesn't not exist");

}

return null;}

From Client Side we use ClientIntercepter to add Token to SOAP header

public class SecurityInterceptor implements ClientInterceptor {

@Override

public boolean handleRequest(MessageContext messageContext) throws WebServiceClientException {

TransportContext context = TransportContextHolder.getTransportContext();

HttpUrlConnection connection = (HttpUrlConnection) context.getConnection();

try {

connection.addRequestHeader("Authorization","Basic VVNFUk5BTUU6cGFzc3dvcmQ=");

} catch (IOException e) {

log.error(e.getMessage());

}

return true;

}

//TODO:: other methods and constructor..

}