Desarrollo de Servicios Web 1

Tema Nº1:ARQUITECTURA Y ESTÁNDARES

Indicador de logro Nº1:Aplica los fundamentos de comunicación en sistemas distribuidos a través de los conceptos generales de web services.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº1:**

ARQUITECTURA Y ESTÁNDARES

**Subtema 1.1:**

¿Qué es Web Services?

**MARCO TEÓRICO**

Es un conjunto de estándar y protocolo para el desarrollo de aplicaciones que necesitan exponer o consumir interfaces de otras aplicaciones que pueden estar desarrolladas en lenguajes de programación diferentes o no, con capacidad para interoperar en forma remota.

Un sistema desarrollado por ejemplo Java, puede comunicarse con un sistema en PHP a través de un Web Services, esta es una de las aplicaciones principales, pero no necesariamente tienen que ser de diferentes plataformas, los sistemas pueden estar realizados en un mismo lenguaje.

**Historia**

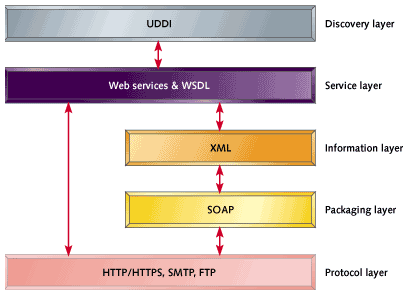
Los Servicios Web surgieron ante una necesidad de estandarizar la comunicación entre distintas plataformas (PC, Mainframe, Mac, etc.) y lenguajes de programación (PHP, C#, Java, etc.).

Los estándares anteriores a los actuales más conocidos fueron DCOM y CORBA, por ser dependientes de la implementación del vendedor DCOM - Microsoft, y CORBA - ORB no tuvieron demasiado éxito ya que la función principal de comunicar plataformas en diferentes lenguajes era muy difícil de llevar a cabo.

Otro gran problema es que para realizar la comunicación entre diferentes nodos se hacía uso de RPC (Remote Procedure Call), esto además de presentar ciertos problemas de seguridad, tiene la desventaja de que su implementación en un ambiente colmado de firewalls y dispositivos, imposibilitan a dos computadoras conectadas por Internet comunicarse.

Los Web Services surgieron para finalmente poder lograr la tan esperada comunicación entre diferentes plataformas. Es por esto que en 1999 se comenzó́ a plantear un nuevo estándar, el cual terminaría utilizando XML, SOAP, WSDL, y UDDI.

**Web Services Protocol Stack**



Es un conjunto de protocolos y estándares utilizados para definir cómo interactúa un Servicio Web con otro. El stack de protocolos que compone la implementación de Web Services es el siguiente:

* XML
* SOAP
* XML-RCP
* HTTP
* FTP
* SMTP
* WSDL
* UDDI
* WS-Security

**XML - eXtensible Markup Language**

Es un lenguaje de marcas (TAGS, Etiquetas) desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar información.

**SOAP - Simple Object Access Protocol**

Es un protocolo de comunicación que permite la comunicación entre aplicaciones a través de mensajes por medio de una red. Es independiente de la plataforma, y del lenguaje, está basado en XML y es la base principal de los Web Services. Los mensajes SOAP son documentos XML.

**WSDL - Web Services Description Language**

Es un protocolo basado en XML que describe las interfaces del Web Service. Indica cuales son las interfaces que provee el Servicio web y los tipos de datos necesarios para la utilización del mismo.

**UDDI - Universal Discovery Description and Integration**

Es un modelo de directorios para Web Services. Es una especificación para mantener directorios estandarizados de información acerca de los Web Services, sus capacidades, ubicación, y requerimientos en un formato reconocido universalmente.

UDDI utiliza WSDL para describir las interfaces de los Web Services. Es un lugar en el cual podemos buscar cuales son los Servicios web disponibles, una especie de directorio en el cual podemos encontrar los Web Services publicados y publicar los Web Services que desarrollemos.

**Subtema 1.2:**

¿Tipos de Web Services?

**MARCO TEÓRICO**

**Soap**

Soap son las siglas de Simple Object Access Protocol. Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

**Rest**

REST son las siglas de REpresentational State Transfer. Es un servicio Web al estilo arquitectónico en el que la atención se centra en la presencia de recursos en el sistema. Cada recurso debe ser identificado por el identificador global de un URI. Para acceder a estos recursos, los clientes se comunican con el servicio REST por HTTP, y el servidor responde con la representación de los recursos.

**REST vs. SOAP**

* REST utiliza HTTP, en consecuencia, es mucho más sencillo desarrollar APIs, crear clientes y la documentación es más fácil de entender.
* REST permite muchos formatos de datos, dando al desarrollador la posibilidad de utilizar JSON en lugar de XML que es más rápido, permite también un mejor soporte a los clientes del explorador. SOAP solamente permite XML.
* REST tiene mejor escalabilidad y rendimiento.
* Las lecturas del REST se pueden cachear, las lecturas basadas en SOAP no se pueden.
* SOAP resuelve más aspectos de seguridad, si bien tanto REST como SOAP soportan SSL, SOAP también soporta WS-Security lo que añade características de seguridad Enterprise.
* SOAP proporciona una implementación estándar de integridad de datos y privacidad de datos.
* REST no soporta comunicación de fallos.

**1. MATERIALES**

Para la experiencia a realizar se requiere lo siguiente:

1. EQUIPO

Se requiere tener un equipo de cómputo (PC o laptop).

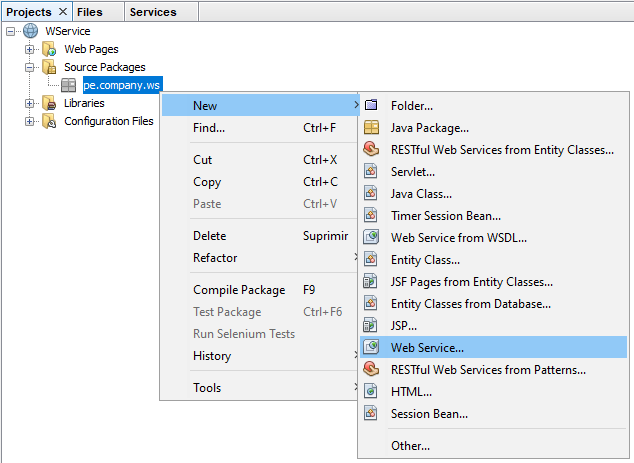
1. HERRAMIENTAS

* Java Developer Kit (JDK) version 8.x
* NetBeans 8.2
* Apache Tomcat 9.0.12
* Jax-ws/metro
* SoapUI 5.5

**2. PROCEDIMIENTO**

Se va a crear un Web Service, con un proyecto java web, que reciba como parámetro un String y devuelva el mismo String concatenado a “Hola Mundo con Web Service”.

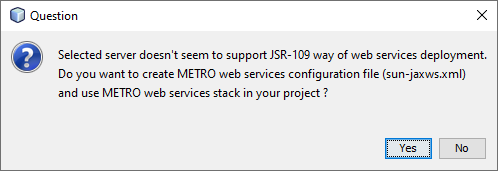
Creamos un Web Service:



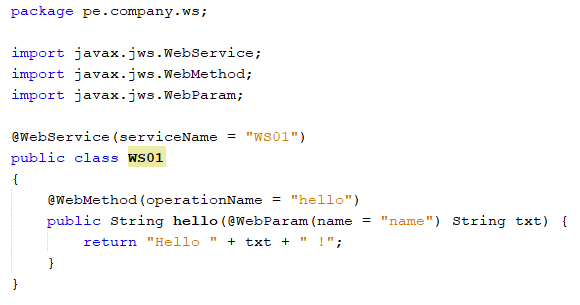
Ponemos un nombre para el Web Service



Nos pregunta si debemos agregar la librería METRO, le damos Yes:



Por defecto nos muestra:



* @WebService(serviceName = "WS01")

Se observa el nombre del Servicio Web

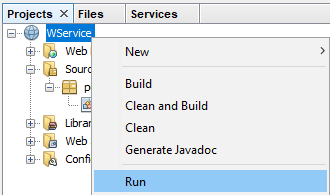
* @WebMethod(operationName = "hello")

Se observa el nombre de la operación del Servicio Web

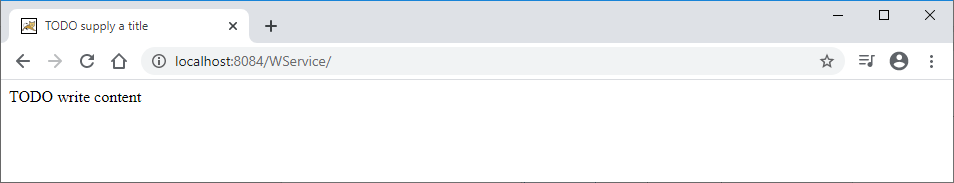
Adecuamos para un mejor mensaje:



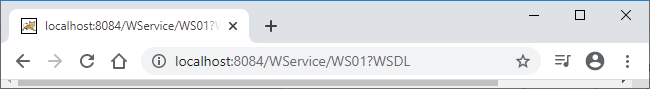
Una vez creado el Servicio Web, se ejecuta el proyecto:



Se observa en el explorador:



Se pasa a obtener el WSDL agregando:



<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is

JAX-WS RI 2.2-hudson-740-. -->

<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is

JAX-WS RI 2.2-hudson-740-. -->

<definitions

xmlns:wsu=*"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"*

xmlns:wsp=*"http://www.w3.org/ns/ws-policy"*

xmlns:wsp1\_2=*"http://schemas.xmlsoap.org/ws/2004/09/policy"*

xmlns:wsam=*"http://www.w3.org/2007/05/addressing/metadata"*

xmlns:soap=*"http://schemas.xmlsoap.org/wsdl/soap/"*

xmlns:tns=*"http://ws.company.pe/"*

xmlns:xsd=*"http://www.w3.org/2001/XMLSchema"*

xmlns=*"http://schemas.xmlsoap.org/wsdl/"*

targetNamespace=*"http://ws.company.pe/"* name=*"WS01"*>

<types>

<xsd:schema>

<xsd:import namespace=*"http://ws.company.pe/"*

schemaLocation=*"http://localhost:8084/WService/WS01?xsd=1"* />

</xsd:schema>

</types>

<message name=*"hello"*>

<part name=*"parameters"* element=*"tns:hello"* />

</message>

<message name=*"helloResponse"*>

<part name=*"parameters"* element=*"tns:helloResponse"* />

</message>

<portType name=*"WS01"*>

<operation name=*"hello"*>

<input wsam:Action=*"http://ws.company.pe/WS01/helloRequest"*

message=*"tns:hello"* />

<output wsam:Action=*"http://ws.company.pe/WS01/helloResponse"*

message=*"tns:helloResponse"* />

</operation>

</portType>

<binding name=*"WS01PortBinding"* type=*"tns:WS01"*>

<soap:binding

transport=*"http://schemas.xmlsoap.org/soap/http"* style=*"document"* />

<operation name=*"hello"*>

<soap:operation soapAction=*""* />

<input>

<soap:body use=*"literal"* />

</input>

<output>

<soap:body use=*"literal"* />

</output>

</operation>

</binding>

<service name=*"WS01"*>

<port name=*"WS01Port"* binding=*"tns:WS01PortBinding"*>

<soap:address

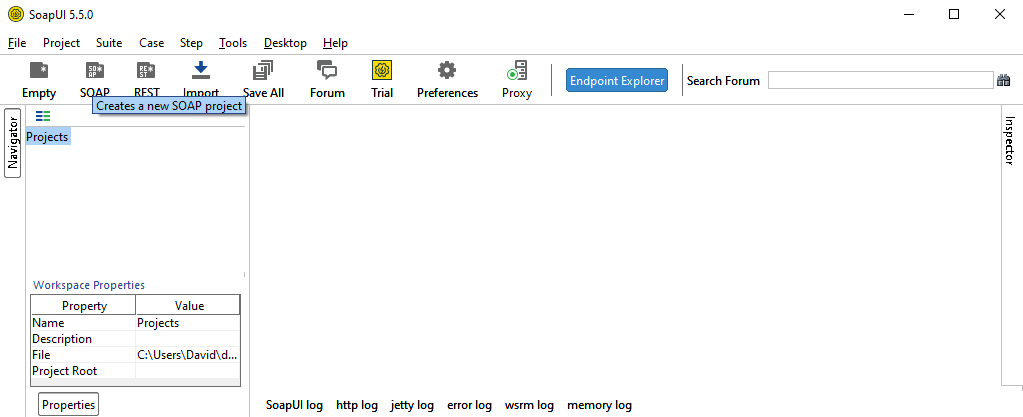
location=*"http://localhost:8084/WService/WS01"* />

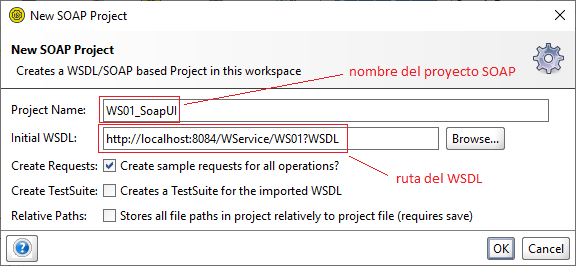
</port>

</service>

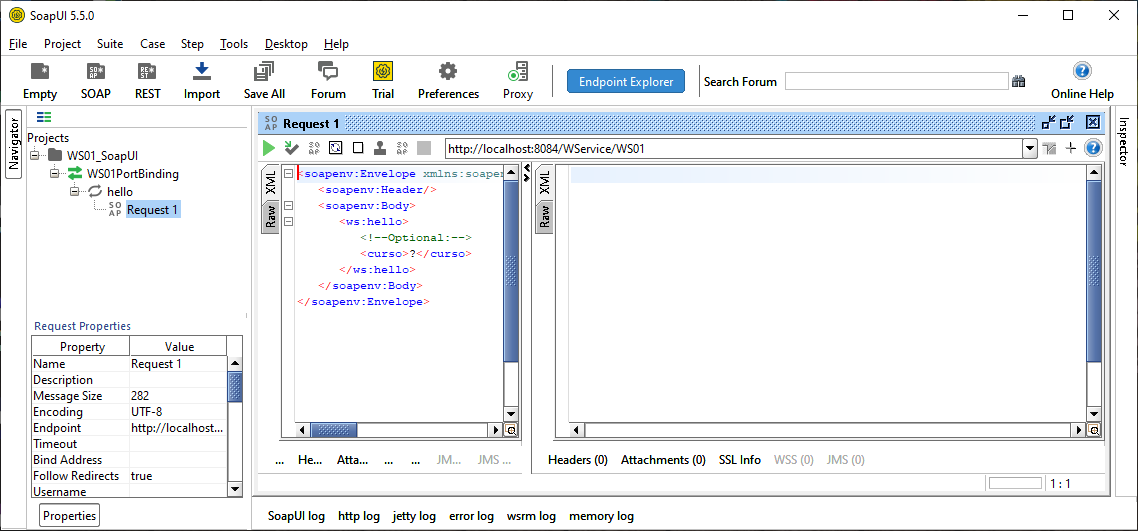
</definitions>

Ahora se pasará a probar este Servicio Web usando SoapUI, Creamos un nuevo proyecto SOAP:





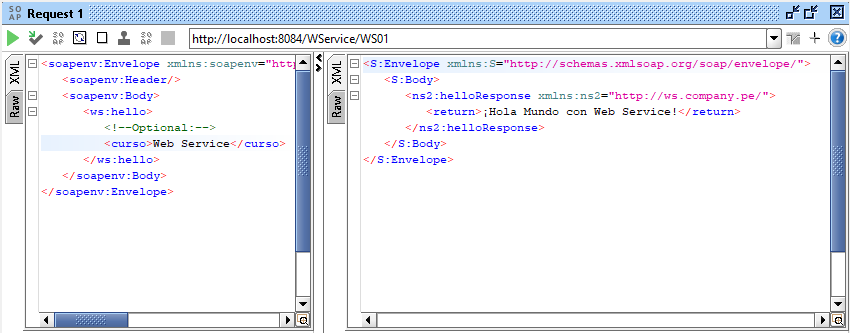
Pasa a crear la trama REQUEST:



Dentro del Envelope se tiene el Body y es en donde se va a enviar la trama:



Presionamos el botón ejecutar:



Se observa la trama RESPONSE, el mensaje de respuesta.

**ACTIVIDAD VIRTUAL:**

1. **CUESTIONARIO TÉCNICO**

Revisa y analiza el tema desarrollado en la presente sesión, luego responde las siguientes preguntas propuestas:

* ¿Qué es un Web Service y cuál es su importancia en la informática?
* ¿Cuáles son los componentes de los Web Service?
* ¿Cuáles son las ventajas y desventajas de los Web Service?
* ¿SOAP y REST son lo mismo, explique el funcionamiento de cada una de ellos?
* Realice un Web Service mediante un código y mostrar sus datos personales.

1. **ENLACES DE REFERENCIA**

* http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html
* https://es.wikipedia.org/wiki/Servicio\_web
* https://www.chiyanasimoes.com/blog/rest-vs-soap-al-servicio-de-la-web

1. **CONCLUSIONES DE LA EXPERIENCIA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_