Aplicaciones con Enfoque Orientado a Servicios

Tema Nº2:Introducción a servicios web y servicios web SOAP

Indicador de logro Nº2:Construye servicios web utilizando el protocolo SOAP

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº2:**

Introducción a servicios web y servicios web SOAP

**Subtema 2.1:**

Introducción a servicios web

**¿Qué es un servicio?**

Es un recurso abstracto que representa una capacidad para realizar tareas que representa una funcionalidad coherente desde el punto de vista de entidades proveedoras (productor) y solicitantes (consumidor). Para ser utilizado, un servicio debe ser implementado por un agente proveedor concreto.

Un servicio tiene las siguientes características:

* Es reutilizable
* Está débilmente acoplado
* Contiene lógica subyacente

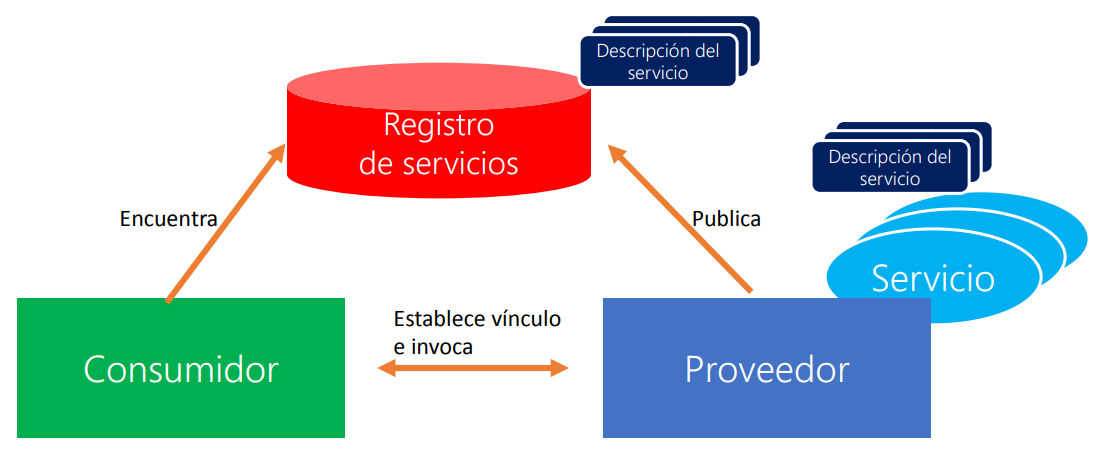
**¿Qué es un servicio web?**

Es un componente de software orientado a funcionar en un entorno distribuido. Su objetivo, como su nombre lo describe, es brindar funcionalidades como servicios que serán consumidos por otras aplicaciones.

Los servicios web se concentran netamente en la lógica que les concierne y la arquitectura con la que se han definido para ser eficientes, puesto que generalmente, son componentes concurrentes, es decir, una o más aplicaciones los usan para el intercambio de información o ejecución de transacciones. Para su funcionamiento, se utilizan distintos protocolos y estilos de arquitectura.

En la actualidad existen dos tecnologías orientadas al desarrollo de estos servicios:

* Servicios Web SOAP basados en el protocolo con el mismo nombre.
* Servicios Web REST, que no se implementan siguiendo un protocolo, sino un estilo de arquitectura.

****

*Imagen 1.* Modelo conceptual de un servicio

Por lo general, los componentes básicos de un servicio web son:

* Mecanismo para encontrar y registrar el interés en un servicio
* Definición de los parámetros de entrada y salida del servicio
* Mecanismo de transporte para acceder a un servicio

Sin embargo, también pueden incluir otras tecnologías para proporcionar características adicionales como seguridad, procesamiento de transacciones y otros.

Para trabajar con un servicio web, se sigue el siguiente proceso:

1. El proveedor de servicios publica un servicio en un registro externo
2. Un cliente (consumidor de servicios) busca un servicio en el registro
3. El registro devuelve información sobre el servicio
4. El cliente se vincula al servicio subyacente
5. El cliente invoca y accede al servicio

Algunas de las razones para implementar servicios web son:

* Son neutrales a la plataforma
* Son accesibles de forma estándar
* Son accesibles de forma interoperable
* Usan arquitectura simple y ubicua
* Son relativamente baratos
* Simplifican la integración empresarial

**Ejemplos:**

**Integración de aplicaciones**

Los sistemas heredados se pueden envolver como servicios web y están disponibles para la integración con otros sistemas.

Las aplicaciones expuestas como servicios web son accesibles por otras aplicaciones que se ejecutan en diferentes plataformas de hardware y que pueden estar escritas en diferentes lenguajes de programación.

**Integración business to business (B2B)**

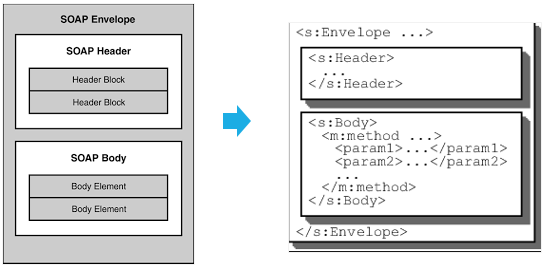
Los servicios web permiten la integración de socios de empresa a empresa (B2B) a través de Internet, de tal forma que se puedan respaldar procesos de negocio entre dos o más empresas, como las operaciones comerciales, bancarias, seguridad, entre otras.

**Subtema 2.2:**

Servicios web SOAP

*Simple Object Access Protocol*  o SOAP es un protocolo de comunicación basado en XML que permite establecer la forma de comunicación entre las aplicaciones que publican o consumen servicios web, especificando especifica el formato que deben tener los mensajes. Sin embargo, es independiente de la plataforma y lenguaje de programación.

Un mensaje SOAP está compuesto por un sobre (*envelope*), cuya estructura está formada por una cabecera (*header*) y un cuerpo (*body*).



*Imagen 2.* Estructura de un mensaje SOAP

**Componentes SOAP**

* XML

Son las siglas de eXtensible Markup Language y fue diseñado para almacenar y transportar datos. Su característica principal es el uso de etiquetas que permite ser legible tanto por humanos como por máquinas..

* XML Schema

Un esquema XML describe la estructura que tiene o debe tener un documento XML.

Si un documento XML tiene la sintaxis correcta se denomina que el documento está "bien formado". Si este mismo documento además está validado con un esquema XML se dice que está "bien formado" y es "válido".

* XSLT

Es un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros, incluso a formatos que no son XML. Las hojas de estilo XSLT realizan la transformación del documento utilizando una o varias reglas de plantilla unidas al documento fuente a transformar.

La unión de XML y XSLT permite separar contenido y presentación, aumentando así la productividad.

* XQuery

Es un lenguaje de consulta diseñado para escribir consultas sobre colecciones de datos expresadas en XML.

Algunas de las aplicaciones de XQuery son: extraer información para usar en un servicio web, generar informes resumidos, transformar datos XML a XHTML y buscar documentos web para obtener información relevante.

* XPath

Es una sintaxis para definir partes de un documento XML o navegar en su contenido a través de expresiones de ruta. Se utiliza en XSLT y en XQuery.

* WSDL

Son las siglas de Web Services Description Language y se refiere a un documento XML que describe un conjunto de mensajes SOAP y cómo se realiza el intercambio de mensajes. Debido a esto, se le conoce como el contrato de los servicios web SOAP.

**Ejemplos:**

* **XML**

<?xml version="1.0" encoding="UTF-8"?>  
<note>  
  <to>Yandira</to>  
  <from>Edinson</from>  
  <heading>Recordatorio</heading>  
  <body>No te olvides de la reunion este sábado</body>  
</note>

* **Esquema XML**

<xs:element name="note">  
<xs:complexType>  
  <xs:sequence>  
    <xs:element name="to" type="xs:string"/>  
    <xs:element name="from" type="xs:string"/>  
    <xs:element name="heading" type="xs:string"/>  
    <xs:element name="body" type="xs:string"/>  
  </xs:sequence>  
</xs:complexType>  
</xs:element>

* **WSDL**

<message name="getTermRequest">  
  <part name="term" type="xs:string"/>  
</message>  
  
<message name="getTermResponse">  
  <part name="value" type="xs:string"/>  
</message>  
  
<portType name="glossaryTerms">  
  <operation name="getTerm">  
    <input message="getTermRequest"/>  
    <output message="getTermResponse"/>  
  </operation>  
</portType>

**Subtema 2.3:**

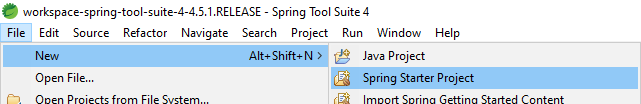
Creación y consumo de servicios web SOAP

Para crear un servicio web SOAP, utilizaremos la herramienta Spring Tool Suite, Maven y Java 8.

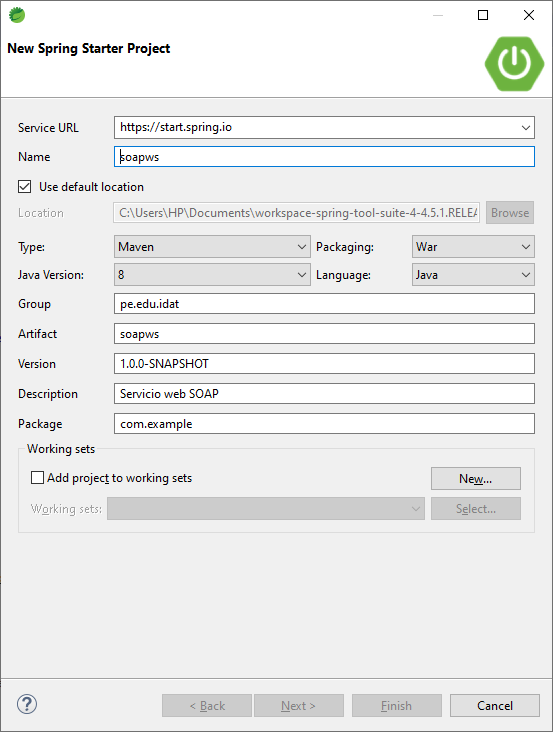
**Ejemplos:**

**Creación de servicio web SOAP**

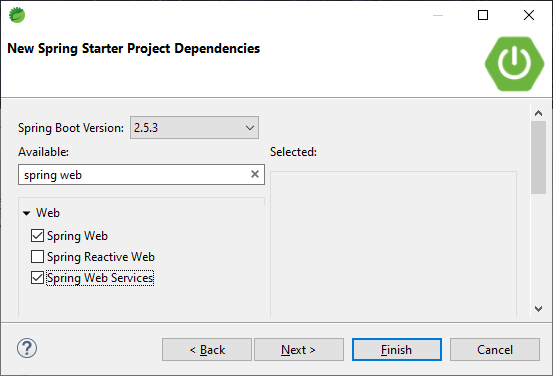
1. En Spring Tool Suite vaya al menú *File > New* y seleccione la opción *Spring Starter Project*.



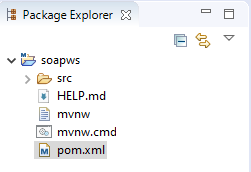
1. Cree el proyecto **soapws** con la siguiente configuración y haga clic en Siguiente:

****

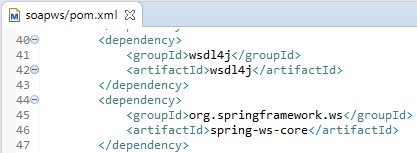
1. Agregue las dependencias **Spring Web** y **Spring Web Services** y haga clic en Siguiente.



1. En la opción de Site Info, haga clic en Finalizar y verifique su proyecto.



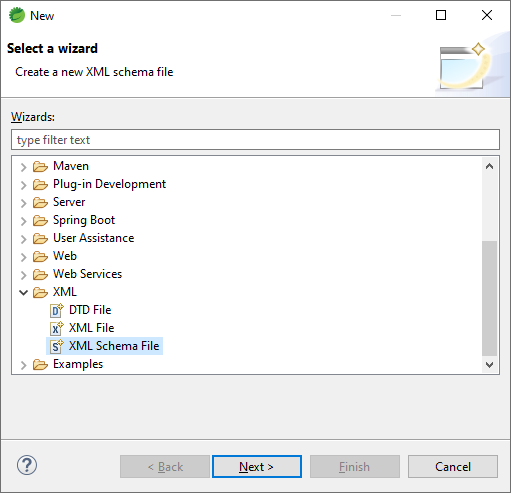
1. Añada las dependencias a **spring-ws-core** y **wsdl4j** en su archivo pom.xml



1. El dominio del servicio web se define en un archivo de esquema XML (XSD) que spring-ws exportará automáticamente como WSDL.

Por ello, cree el archivo XSD **países.xsd en** *src / main / resources*. Este archivo contendrá una operación para devolver el nombre, la población, capital y moneda de un país.

Para crear el archivo vaya a *src / main / resources*, haga clic derecho, seleccione la opción *XML / XML Schema File* y luego haga clic en Siguiente.



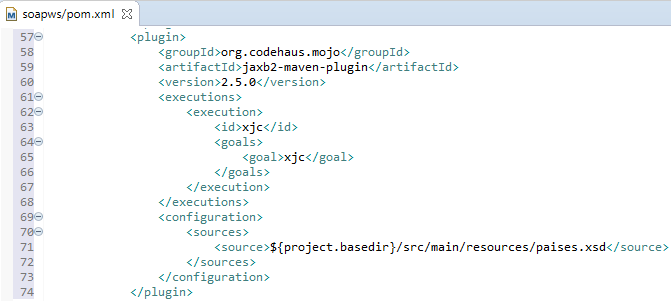
En la siguiente pantalla, ingrese el nombre **países.xsd** y haga clic en Terminar.

1. Modifique el archivo países.xsd de la siguiente forma:

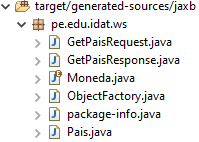




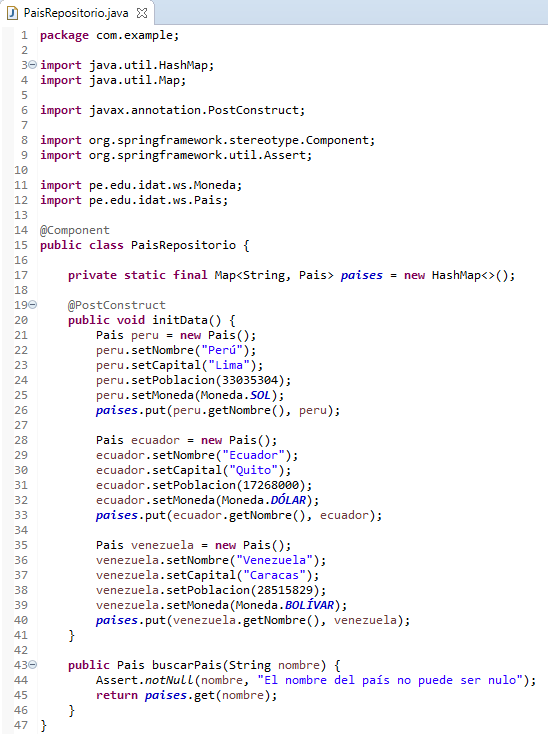
1. Ahora, generaremos clases de dominio Java a partir del archivo de esquema (XSD). Para ello, configuraremos el siguiente *plugin* en el archivo pom.xml.



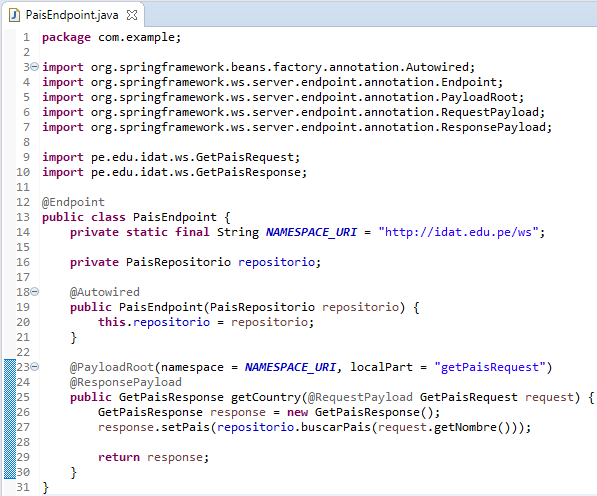
1. Verifique que se hayan generado las clases en el directorio *target/generated-sources/jaxb.*



1. Cree un repositorio de países ficticio con datos en el código. Para ello agregue una nueva clase **PaisRepositorio**.

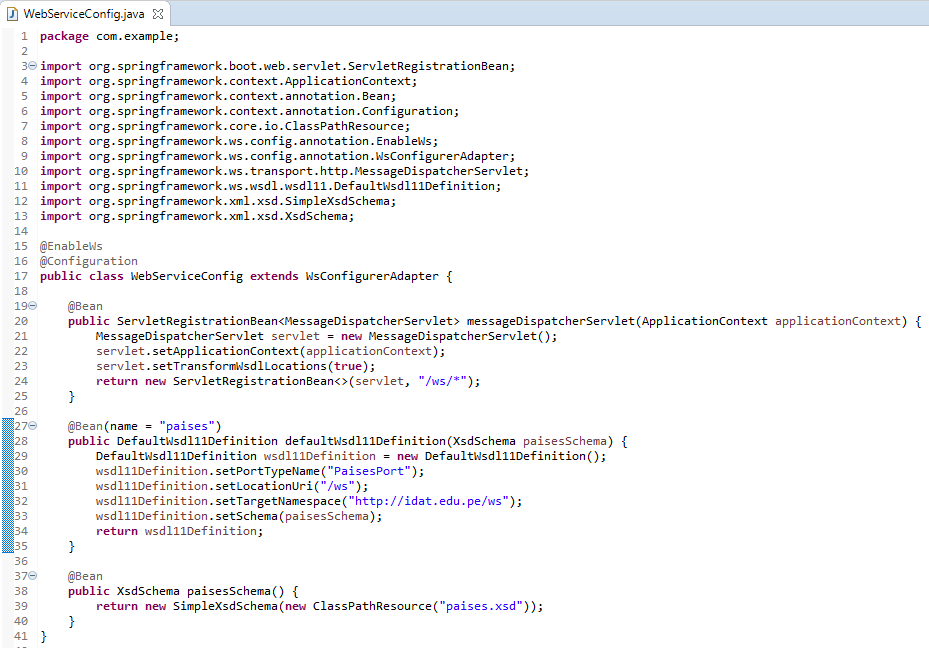


1. Para crear un punto final (*endpoint*) de servicio, cree una clase POJO con algunas anotaciones Spring WS para manejar las solicitudes SOAP entrantes. Para ello agregue una nueva clase **PaisEndpoint**.



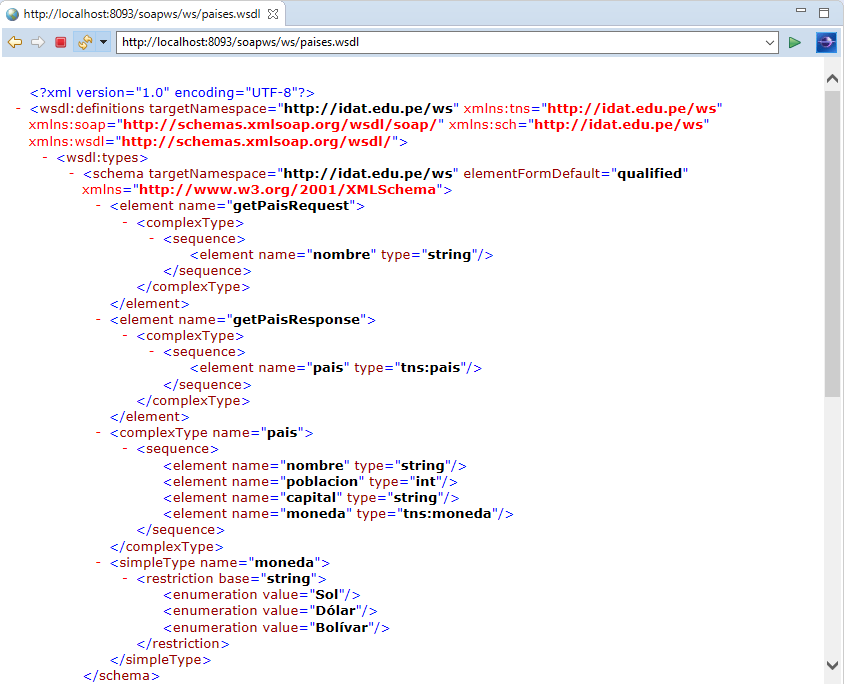
* La anotación @Endpoint registra la clase con Spring WS como un candidato potencial para procesar mensajes SOAP entrantes.
* Luego, Spring WS usa la anotación @PayloadRoot para elegir el método del controlador, según el espacio de nombres del mensaje y localPart.
* La anotación @RequestPayload indica que el mensaje entrante se asignará al parámetro de solicitud del método.
* La anotación @ResponsePayload hace que Spring WS asigne el valor devuelto a la carga útil de la respuesta.

1. Para configurar un bean de servicio web relacionado a Spring, añada la clase **WebServiceConfig** que herede de **WsConfigurerAdapter**.



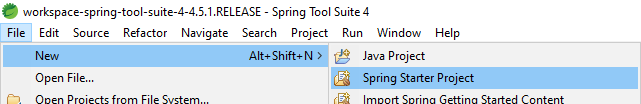
* Spring WS usa un tipo de servlet diferente para manejar mensajes SOAP: MessageDispatcherServlet. Es importante inyectar y configurar ApplicationContext en MessageDispatcherServlet. Sin eso, Spring WS no detectará automáticamente los frijoles Spring.
* Nombrar este bean messageDispatcherServlet no reemplaza el bean DispatcherServlet predeterminado de Spring Boot.
* DefaultMethodEndpointAdapter configura el modelo de programación Spring WS controlado por anotaciones. Esto hace posible utilizar las distintas anotaciones, como @Endpoint.
* DefaultWsdl11Definition expone un WSDL 1.1 estándar utilizando XsdSchema.

1. Ejecute la aplicación y verifique la ruta: <servidor>:<puerto>/<nombre de proyecto>/ws/paises.wsdl

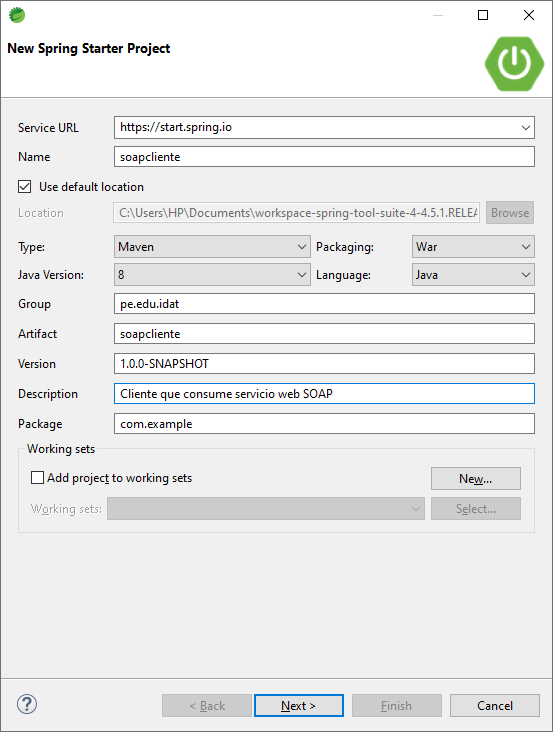


**Consumo de servicio web SOAP**

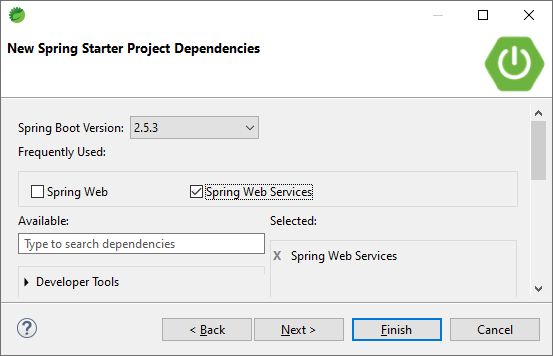
1. En Spring Tool Suite vaya al menú *File > New* y seleccione la opción *Spring Starter Project*.



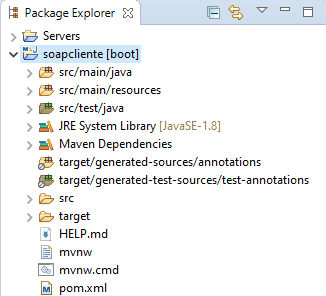
1. Cree el proyecto **soapcliente** con la siguiente configuración y haga clic en Siguiente:



1. Agregue la dependencia Spring Web Services y haga clic en Siguiente.



1. En la opción de Site Info, haga clic en Finalizar y verifique su proyecto.



1. Modifique el archivo **pom.xml** para excluir el servidor tomcat en la dependencia de web services, de tal forma que la sección **dependencies** quede de la siguiente manera:

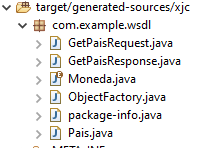


1. Para generar las clases Java a partir del archivo WSDL (propiamente del XSD ubicado en la sección Types) del servicio a consumir, modifique el archivo pom.xml para añadir el siguiente *plugin*.

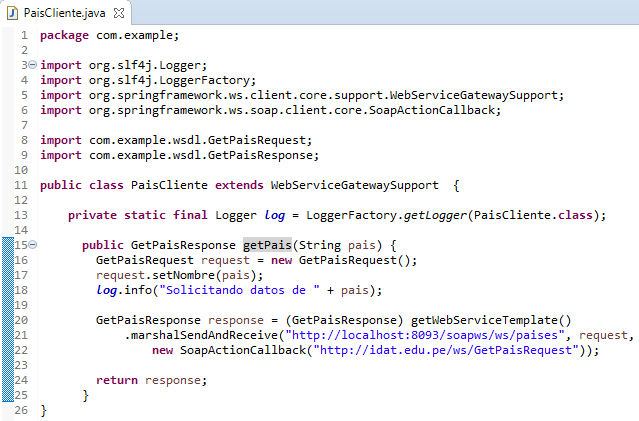
Considere la url (incluido el puerto) del servicio soap que desarrolló inicialmente.

****

1. Esta configuración generará clases para el WSDL que se encuentra en la URL especificada, colocando esas clases en el paquete **com.example.wsdl**.



1. Ahora, añada la clase java **PaisCliente**, que debe extender de la clase **WebServiceGatewaySupport**, para crear un cliente web service.

****

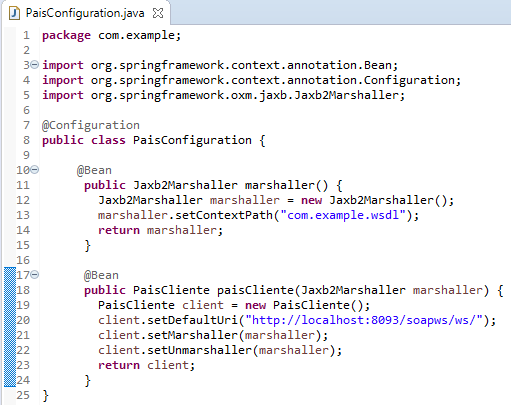
**¿Cómo funciona?**

El cliente contiene el método getPais que realiza la comunicación con el servicio SOAP.

En este método, las clases GetPaisRequest y GetPaisResponse se derivan del WSDL y se generaron en el proceso de generación JAXB (pasos 6 y 7). Luego, crea el objeto de solicitud GetPaisRequest y lo configura con el parámetro *pais* (el nombre del país). Después de imprimir el nombre del país, utiliza WebServiceTemplate proporcionada por la clase base WebServiceGatewaySupport para realizar el intercambio SOAP. Pasa el objeto de solicitud GetPaisRequest (así como un SoapActionCallback para pasar un encabezado SOAPAction con la solicitud) ya que el WSDL describió que necesitaba este encabezado en los elementos. Finalmente, transmite la respuesta a un objeto GetPaisResponse, el cual será devuelto.

1. Ahora configure los componentes de servicios web. Para ello, agregue la clase java **PaisConfiguration**.

Spring web services usa el módulo OXM de Spring Framework, que tiene Jaxb2Marshaller para serializar y deserializar solicitudes XML.



El marshaller apunta a la colección de objetos de dominio generados y los usará para serializar y deserializar entre XML y POJO.

El paisCliente se crea y configura con el URI del servicio del país que se mostró anteriormente. También está configurado para utilizar el contador de referencias JAXB.

1. Modifique la clase **SoapclienteApplication** para empaquetar la aplicación de tal forma que se ejecute desde la consola y recupere los datos de un nombre de país determinado. Luego ejecute la aplicación.



**Subtema 2.4:**

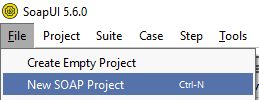
Pruebas de servicios web

Para probar servicios web que hemos desarrollado o de terceros sin necesidad de desarrollar una aplicación cliente, podemos utilizar otros programas. Uno de ellos es SoapUI.

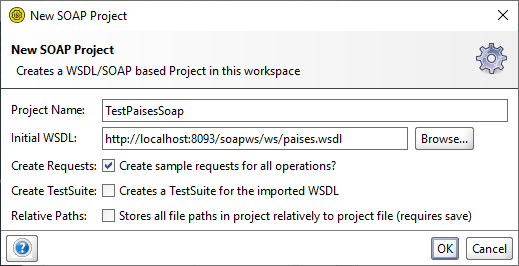
**Ejemplos:**

**Pruebas en SoapUI**

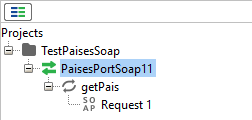
1. En SoapUI, haga clic en el botón SOAP o en el menú *File / New SOAP Project*.



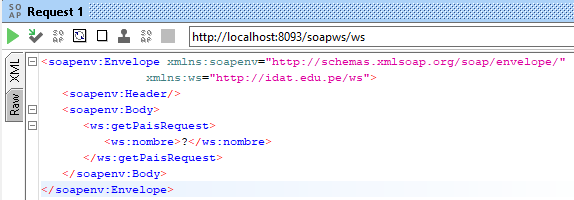
1. En la ventana *New SOAP project,* coloque el nombre TestPaisesSoap, ingrese la URL del archivo WSDL y presione OK.



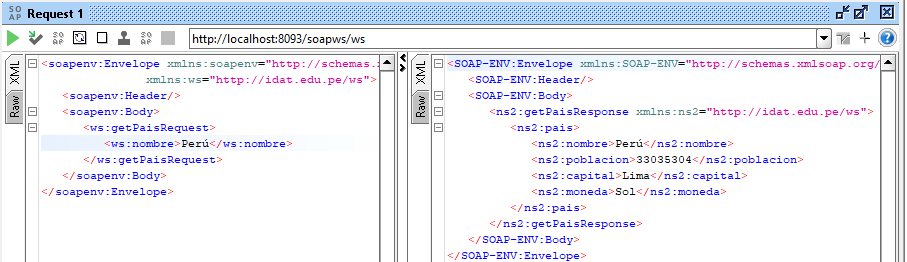
1. Verifique que se haya creado el Request de prueba.



1. Haga doble clic en Request1.



1. En la etiqueta *nombre* ingrese uno de los siguientes nombres: Perú, Ecuador o Venezuela y haga clic en el botón .



1. Realice pruebas con alguno de los otros países.

**Actividad:**

1. Cree una aplicación web que contenga un servicio SOAP que realice un CRUD básico a una entidad (registrar, actualizar, eliminar, listar y obtener por ID). La entidad debe tener como mínimo 6 campos y uno de ellos debe ser un identificador.
2. Cree una aplicación que a través de la consola ingrese datos y consuma los métodos SOAP creados en la actividad N° 1.
3. En SoapUI, realice pruebas de 3 servicios SOAP públicos de la Universidad de Alicante.

**Documentación**: <https://si.ua.es/es/servicios-web/serviciosweb/publicos/publicos.html>

**WSDL**: <https://cvnet.cpd.ua.es/servicioweb/publicos/pub_gral.asmx?WSDL>