Java Spring Boot

Unidad 1. Fundamentos de servicio web

Índice

Introducción

La creación de aplicaciones distribuidas basadas en el uso de servicios Web, es algo más que una moda de programación, ya que desde que apareció el concepto de servicio Web a principios de los años 2000, no ha sido hasta los últimos años cuando ya se ha empezado a implantar definitivamente la arquitectura orientada a servicios.

Este despegue se ha producido gracias fundamentalmente a dos elementos tecnológicos:

- El paradigma REST, que permite crear servicios Web de manera simple y fáciles de utilizar.
- El framework Spring, y más concretamente Spring Boot, que simplifica enormemente la creación de este tipo de servicios, su configuración y despliegue en la nube.

A lo largo de este módulo, analizaremos los conceptos relativos a la arquitectura de los servicios Web y las diferentes aproximaciones para abordar su desarrollo.

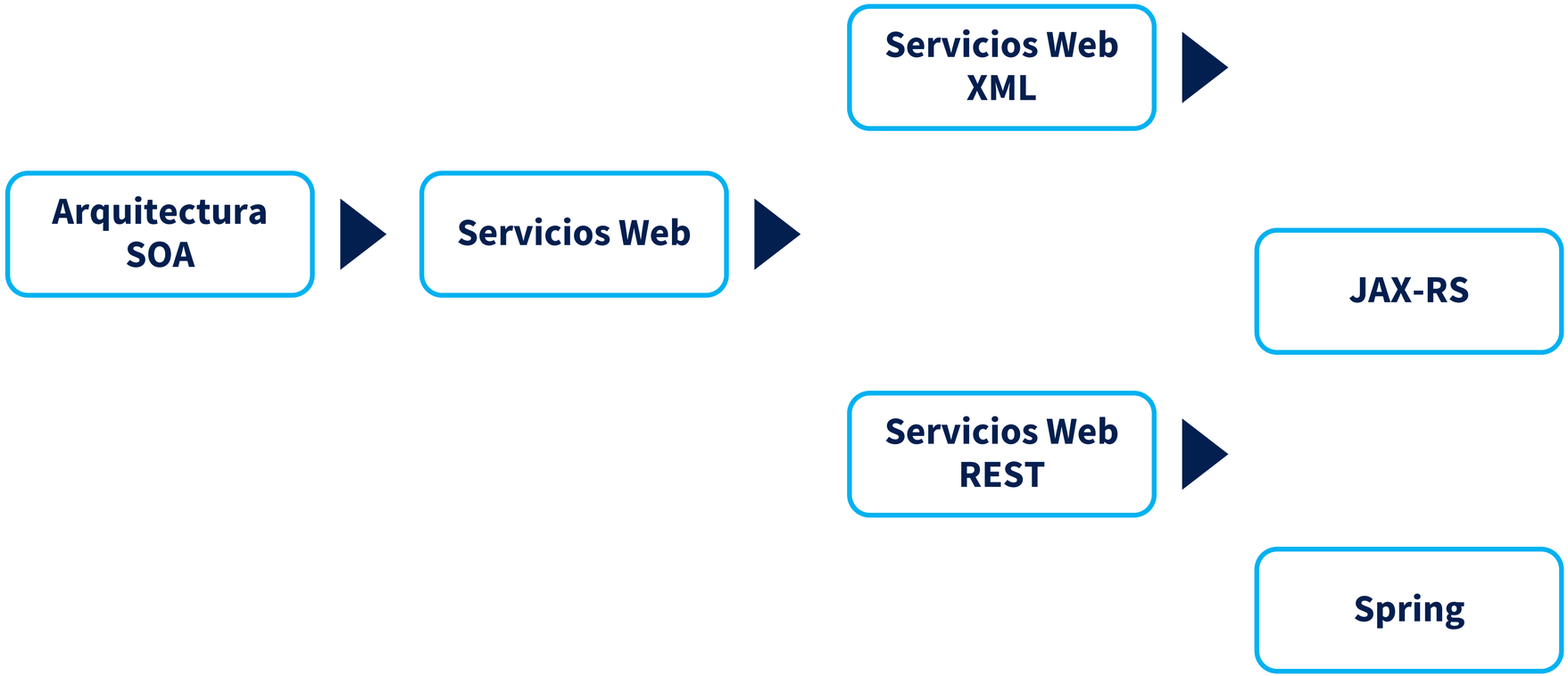
Objetivos

Comprender que es un servicio Web y su estructura.

Conocer las ventajas del uso de servicios Web.

Identificar las tecnologías y arquitecturas de servicios Web.

Mapa Conceptual



Arquitectura SOA

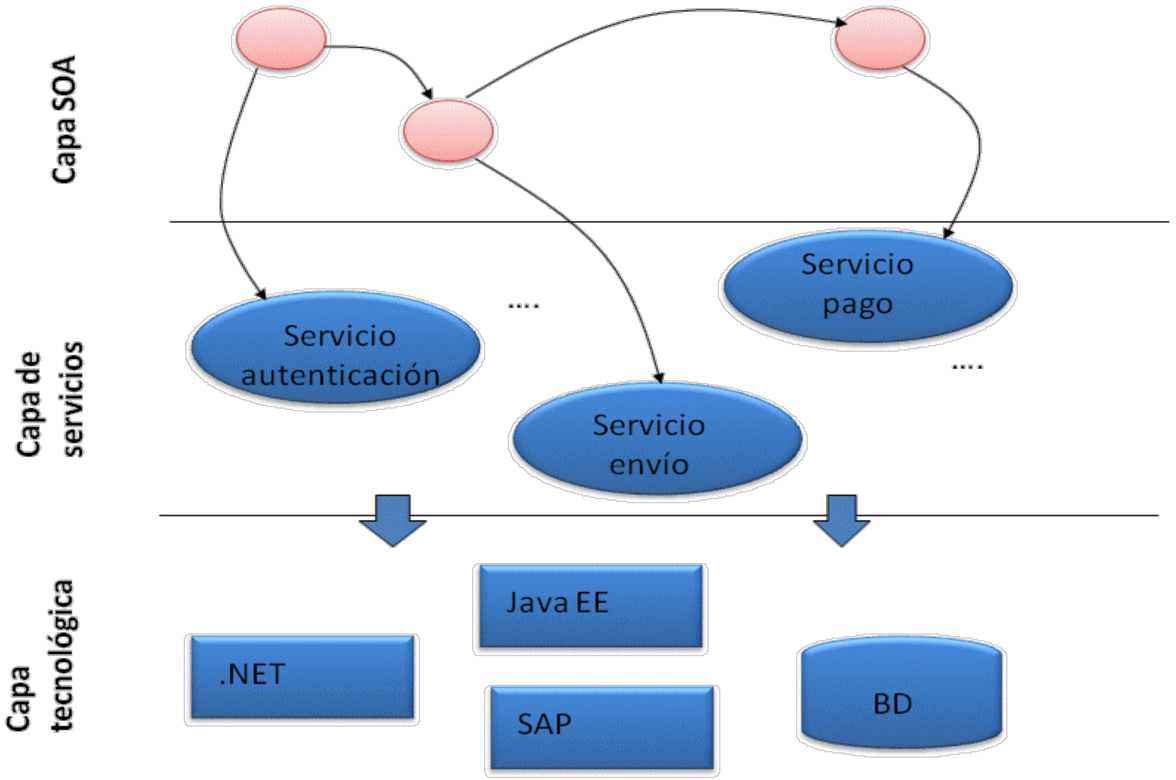
- La arquitectura orientada a servicios (SOA) es un paradigma de programación en el que las aplicaciones se crean a base de interconectar servicios, ya creados, que realizan determinadas funcionalidades.

- En SOA, un **servicio es un módulo de software reutilizable que realiza una determinada función**. Los servicios ofrecen operaciones (métodos) que pueden ser utilizados por otras aplicaciones para realizar tareas de más alto nivel.

- De esta manera, frente al desarrollo "monolítico" de las aplicaciones tradicionales, donde el código se escribe expresamente para cumplir con los requerimientos de una determinada aplicación, en el **desarrollo SOA las aplicaciones son distribuidas**, pues se construyen uniendo piezas de código independientes y autofuncionales, que se encuentran repartidas por la red.

Arquitectura SOA

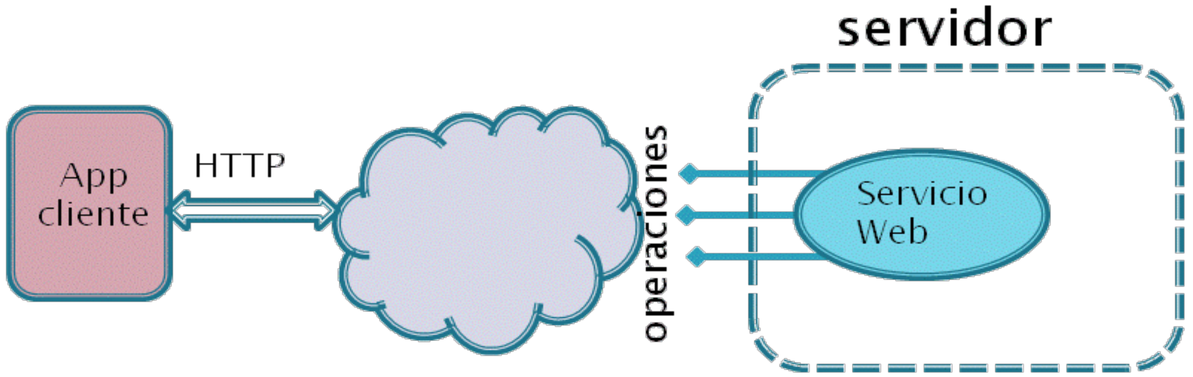
Las aplicaciones SOA se construyen en base a la interconexión de servicios. En la zona más interna, tenemos la **capa tecnológica**, que representa el lenguaje de programación o tecnología de desarrollo con la que se construye el servicio. Por encima de esta capa está la **capa de servicio**, que expone sus funcionalidades a la capa superior, permitiendo que esta se abstraiga de la tecnología de desarrollo empleada. Por último, **la capa SOA se construye según la orquestación de las llamadas a los servicios**.



Servicios Web

Concepto de servicio Web

Un servicio Web es un caso particular de servicio SOA. Se trata de un **componente de código reutilizable que expone una serie de operaciones a través de la Web**, a fin de que dichas operaciones puedan ser consumidas por otras aplicaciones.



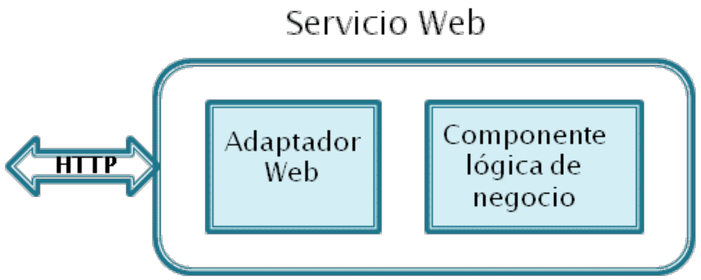
Como las aplicaciones Web tradicionales, un servicio Web es desplegado en un servidor de aplicaciones y es accesible a través del protocolo HTTP. Pero a diferencia de aquellas, los servicios Web no **ofrecen su funcionalidad** a un navegador, sino **a otros programas** que pueden manipular los datos de respuesta para cualquier fin.

Servicios Web

Estructura interna

Independientemente del tipo de aproximación utilizada en su implementación, un servicio Web se compone de dos partes fundamentales :

- **Lógica de negocio**. De lo que se trata con los servicios es de ofrecer ciertos métodos de negocio al exterior para que sean utilizados remotamente por otros programas, por tanto, una de las tareas será la de **implementar esta lógica de negocio en algún componente software**. En el caso de Java, estos componentes pueden ser **EJBs , clases estándares**, o **beans de Spring**.
- **Adaptador web**. Para **independizar al cliente de la tecnología utilizada en la creación de un servicio web**, es necesario desarrollar un bloque de código adaptador que, por un lado, se encargará de interpretar las llamadas externas y derivarlas a la lógica de negocio del componente y, por otro, de "adaptar" los resultados generados por dicha lógica a un formato interpretable por el cliente y, por consiguiente, que no dependa de la tecnología con la que se ha desarrollado el servicio. El intercambio de datos entre un servicio Web y las aplicaciones clientes se realiza utilizando algún formato de representación independiente, como **XML o JSON**.



Servicios Web

Aproximaciones de implementación

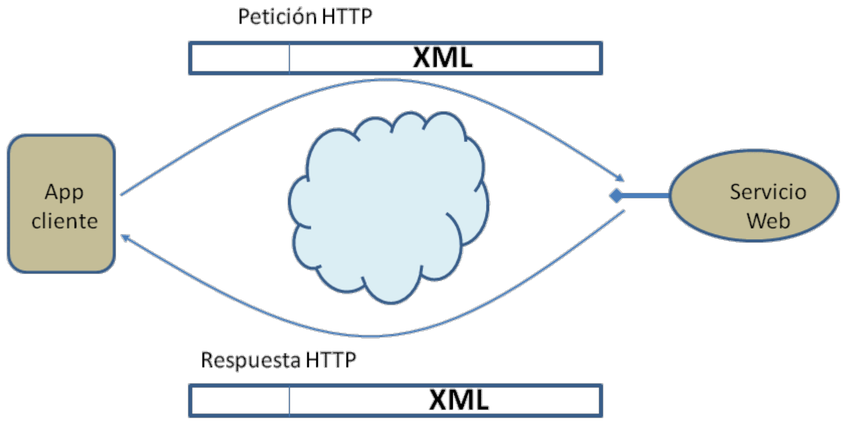
A la hora de implementar una arquitectura basada en servicios Web, existen dos aproximaciones o estándares para llevar a cabo esta labor:

- **Servicios Web XML**. Los servicios Web XML exponen operaciones en la Web a las que se accede mediante peticiones HTTP que incluyen un **documento XML en el cuerpo de la petición**, que contiene la información necesaria para realizar la llamada a la operación. Los datos de respuesta son también envueltos en un documento XML que se envía en el cuerpo de dicha respuesta. Se trata de la primera aproximación a la creación de servicios Web y está bastante en desuso.
- **Servicios REST**. Los servicios REST exponen una serie de recursos a los que se accede con **una simple petición HTTP**, sin necesidad de enviar ningún documento con información adicional. La propia **información contenida en la petición (tipo de método, url, parámetros, etc.), determina la operación a realizar por parte del servicio**. Si el servicio REST y el cliente necesitan intercambiar algún dato complejo, este puede estar formateado con cualquier mecanismo soportado por ambos, normalmente JSON que es mucho más ligero que XML.

Servicios Web XML

Características y funcionamiento

Como ya hemos indicado, un servicio Web XML es aquel que expone una serie de operaciones en la Web a las que se accede mediante el intercambio de documentos XML entre el servicio y la aplicación cliente. Estos **documentos deben tener un formato específico**, definido por uno de los estándares de la arquitectura de servicios Web XML.



Debido a su rigidez y peor eficiencia respecto a REST, esta aproximación es cada vez menos usada y, aunque no la vamos a tratar en profundidad en este curso, conviene conocer los diferentes estándares y tecnologías en las que se basa.

Servicios Web XML

Estándares

En los servicios web XML, el uso de HTTP como protocolo de comunicación y de XML como formato de intercambio de datos constituyen la base de la arquitectura. Pero, además de estos estándares básicos, hay ciertos problemas que resolver cuando se van a implementar aplicaciones basadas en servicios web, como la **estructura que tienen que tener los documentos XML** que se van a intercambiar el servicio web con sus clientes, o la manera en la que se deben especificar las **operaciones que un servicio web ofrece.**

Para la resolución de estas cuestiones, el Consorcio Internacional de la Web (**W3C) ha definido unos estándares**, que establecen la forma en la que un servicio web debe definir su interfaz y cómo debe interaccionar con las aplicaciones clientes. Estos estándares son:

- **SOAP**. *Simple Object Access Protocol*, establece el **formato de los documentos XML** que deben intercambiarse el servicio web y las aplicaciones clientes, tanto durante la llamada a los métodos del servicio como al generar la respuesta.
- **WSDL**. *Web Service Description Language* también es un lenguaje basado en XML y se emplea para **describir la interfaz del servicio**, es decir, qué operaciones expone, parámetros de entrada y salida, así como la localización del propio servicio.

Servicios Web XML

El estándar SOAP

Comentemos brevemente los fundamentos de *Simple Object Application Protocol* (SOAP). Se trata de un lenguaje XML que se emplea para **definir la estructura y formato de los documentos XML que se van a intercambiar el servicio web con las aplicaciones clientes**. Estos documentos viajan en el cuerpo de las peticiones y respuestas HTTP, y se les conoce como **mensajes SOAP.** Un mensaje SOAP incluye:

- ***Envelope***. Se trata del elemento raíz de un documento SOAP. Define las **referencias a los distintos espacios de nombres utilizados por el documento**.
- ***Header***. Es un elemento opcional que contiene **información de control** sobre el documento.
- ***Body***. Es el elemento más importante de un documento SOAP, ya que contiene la parte principal del mensaje SOAP. Su estructura es muy sencilla; en el caso de los mensajes de llamada a una operación, el interior del *body* **incluirá un elemento con el nombre de la operación a invocar y, dentro de este, un grupo de subelementos con los argumentos de la llamada**. En el mensaje de respuesta, el *body* incluirá tan solo un elemento con el resultado devuelto por el método.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <ns:comprobarPrecio xmlns:ns="http://MyServices/">
            <cod>B7AW</cod>
        </ns:comprobarStock>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

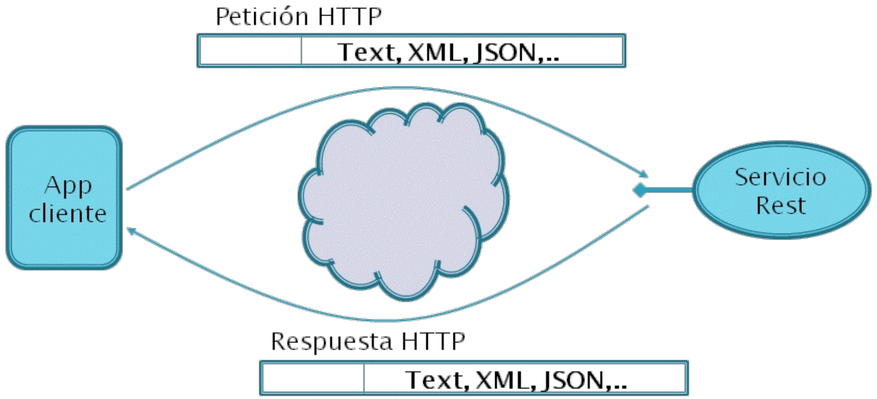
Servicios REST

Fundamentos

Los servicios tipo REST (Representational State Transfer) constituyen una nueva generación de servicios web, basados en exponer a través de Internet una serie de recursos a los que se puede acceder con peticiones HTTP sencillas (tipo GET, PUT, POST, DELETE, etc.).

Estos **recursos consisten en datos** que se ofrecen (pedidos, pedido de un determinado identificador) u **operaciones** sobre esos datos (modificar un pedido, añadir un pedido,..). Cada uno de estos recursos **se identifica por una combinación de URL, método HTTP, variables de URL, tipo de respuesta, tipo consumido**. El acceso a estos recursos se realiza a través de una serie de métodos expuestos por el servicio REST, de modo que cada método es asociado a una combinación de URL, método HTTP, parámetros, tipo de respuesta, tipo consumido.

El formato de intercambio de datos entre servicio REST y el cliente es libre, aunque habitualmente se emplea JSON.



Servicios REST

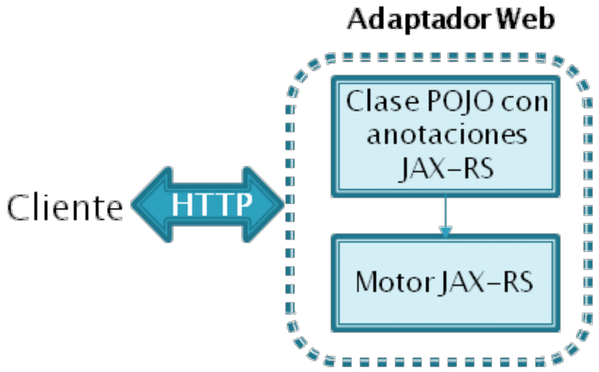
Ventajas

Entre las ventajas que nos proporciona la arquitectura de servicios REST frente a los servicios XML podemos destacar:

- **Acceso más simple**. Para acceder a un servicio REST **no necesitamos incluir ningún documento XML en el cuerpo de la petición**, basta con enviar una simple petición HTTP, que hará que se ejecute una operación dentro del servicio.
- **Flexibilidad**. Los servicios REST nos ofrecen una gran flexibilidad a la hora de intercambiar datos con la aplicación cliente, puesto que se admiten múltiples formatos de. Los servicios web REST no solo admiten datos en XML, sino también en **JSON**, binario, texto plano, etc. Además, la forma de estos datos puede ser cualquiera, no tiene que ajustarse a una determinada estructura. Y lo mismo sucede con los datos de la respuesta.
- **Eficiencia**. La construcción y manipulación de documentos XML resulta bastante tediosa y consume muchos recursos de la máquina. Al poder emplear otros formatos de datos mucho más eficientes se mejora el rendimiento de las aplicaciones.

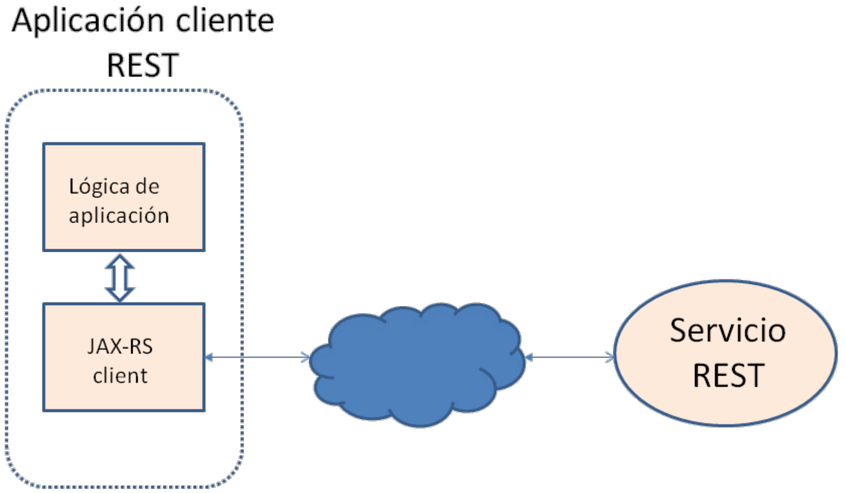
Servicios REST

Tecnologías: El API JAX-RS



JAX-RS es la tecnologías base para la construcción de servicios Web y aplicaciones clientes. Se trata de una especificación que forma parte de Java EE, basada en la utilización de una serie de anotaciones que delegan las tareas de interacción del servicio REST y el cliente en el llamado **motor JAX-RS**.

Existen muchas implementaciones de este API, entre las que se pueden destacar **Jersey y resteasy**. Estas implementaciones también incluyen un conjunto de librerías para la creación de las aplicaciones clientes.

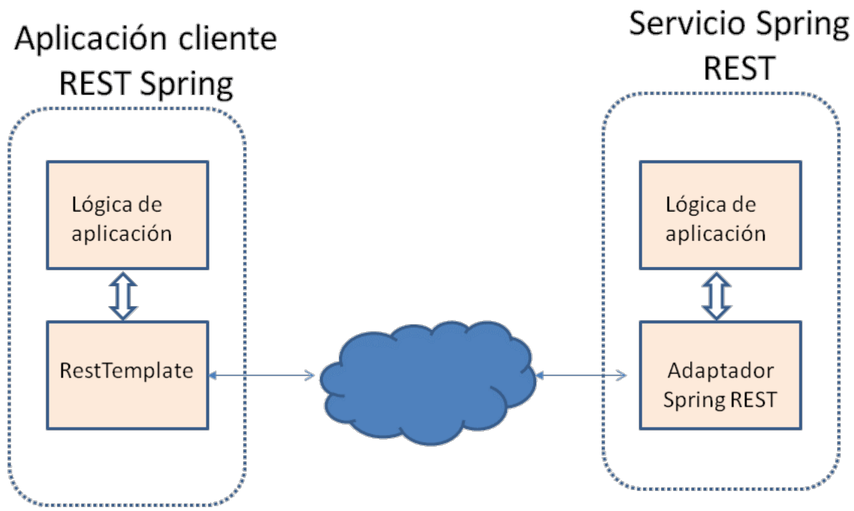


Servicios REST

Tecnologías: Spring REST

Spring es un framework de propósito general, organizado en una serie de módulos con los que se puede abordar el desarrollo de todas las capas de una aplicación.

Para el caso concreto de los servicios REST, **Spring dispone de los módulos Web y WebMVC**, que incluyen una serie de **anotaciones para la creación de aplicaciones basadas en servicios REST**, así como una implementación ligera de JAX-RS que permite desarrollar tanto el servicio como las aplicaciones clientes. A lo largo del siguiente capítulo veremos como utilizar esta parte del framework Spring en la creación de aplicaciones basadas en REST.



Servicios REST

El adaptador Web

Tanto si optamos por utilizar JAX-RS o Spring REST, la implementación de un adaptador en un servicio REST va a consistir en una **clase POJO con una serie de métodos**, en la que incluiremos **unas anotaciones** para mapear dichos métodos a las peticiones que lleguen desde el cliente. Será el motor de servicios el encargado de resolver estas llamadas, de modo que el programador se concentrará en implementaren estos métodos la funcionalidad que el servicio tiene que ofrecer. Estos métodos contendrás básicamente, llamadas a los componentes de negocio.

Los tipos Java devueltos por estos métodos son **convertidos automáticamente a JSON/XML**.

```
@Path(..)
public class ClaseServicio{
   @GET
   @Produces(..)
   public ..metodo(..){
      ..//llamadas a componentes de negocio
   }

   @POST
   @Consumes(..)
   public .. metodo2(..){
       ..//llamadas a componentes de negocio

   }
   ..
}
```

Servicios REST

Cliente

A la hora de implementar un cliente Java de un servicio REST, tenemos dos opciones**, independientemente de cómo esté implementado el servicio**. Recordemos que la interacción entre cliente y servicio se realiza utilizando estándares como HTTP, XML y JSON, de manera que ambos puedan abstraerse de la tecnología con la que el otro está implementado.

Estas opciones son:

- **Utilización de librería jax-rs cliente**. Tanto jersey como resteasy, proporcionan unas implementaciones de jax-rs que permiten acceder a servicios REST, mediante la llamada a simples métodos de clases Java. Dichas librerías se encargan de realizar de forma automática el mapeo de objetos Java a JSON/XML.

- **RestTemplate**. El módulo Web de Spring proporciona la clase RestTemplate con la que podemos acceder a un servicio REST desde una aplicación cliente, simplemente utilizando el juego de métodos que esta clase nos ofrece. Apoyándonos en las librerías jackson proporcionadas por Spring, el mapeo de objetos Java a JSON/XML es realizado de forma automática.

Recuerda

- La arquitectura orientada a servicios (SOA) se basa en la creación de componentes de software, conocidos como servicios, distribuidos en la nube y que pueden interconectarse para desarrollar aplicaciones distribuidas.
- Un tipo de servicio SOA son los servicios Web, que se exponen a través de HTTP para que puedan ser utilizados por otras aplicaciones.
- A la hora de crear un servicio Web hay dos aproximaciones: Servicios Web XML, que se basan en el intercambio de documentos XML con información de llamada a operación y datos de respuesta, y los servicios REST, que son invocados mediante peticiones HTTP y admiten diferentes formatos de datos, como XML y JSON.
- Los servicios REST pueden ser implementados mediante el estándar JAX-RS o el módulo Spring REST. Ambos elementos nos facilitan también la creación de aplicaciones clientes.