

# Capítulo 6: Web Services

---



*Sistemas Distribuidos*

Universidad Nacional de Asunción  
Facultad Politécnica  
Ingeniería Informática

---

*Ing. Fernando Mancía*

# Web Services

---

Introducción

Concepto

Protocolos

☐ XML

☐ SOAP

☐ WSDL

☐ UDDI

Servicios RestFul

Funcionamiento

Utilidad, Aplicaciones, implementaciones

## Web Services - Introducción

---

Los servicios web o **Web Services** por sus siglas en inglés, son interfaces de programación disponibles para la comunicación de aplicaciones en la World Wide Web.

Proporcionan una forma estándar de interoperabilidad entre diferentes aplicaciones de software que se ejecuta en una variedad de plataformas y/o frameworks.

Son componentes software que utilizan protocolos abiertos y son independientes y autónomos.

## Web Services - Introducción

---

Proporciona una interfaz de servicio que permite a los clientes interactuar con servidores de una manera más general que los navegadores web.

Los clientes acceden a las operaciones en la interfaz de un servicio web mediante solicitudes y respuestas formateadas en XML y normalmente transmitidas a través de **HTTP**.

Las interfaces de los servicios web se pueden describir en un IDL. Se debe proporcionar información adicional, incluyendo los protocolos de codificación y comunicación en uso y los lugares de servicio.

## Web Services - Introducción

---

Se necesitan medios seguros para crear, almacenar y modificar documentos e intercambiarlos a través de Internet. Los canales seguros de capa de transporte (TLS) no proporcionan todos los requisitos necesarios. La seguridad XML tiene como objetivo cerrar esta brecha.

<https://www.w3.org/standards/xml/security>

## Web Services - Introducción

---

Los servicios web son cada vez más importantes en los sistemas distribuidos: soportan la interoperabilidad a través de Internet, incluyendo el área clave de la integración entre empresas y también la emergente cultura 'mashup'.

Permite a los desarrolladores crear software creativo innovador además del existente en base a servicios.

Los servicios web también proporcionan el middleware subyacente tanto para Grid como para cloud computing.

## Web Services

---

**Web Services o Servicio Web**, es un sistema de software diseñado para apoyar la interacción interoperable máquina a máquina sobre una red de computadoras. Tiene una interfaz descrita en un formato procesable por máquina (específicamente Web Services Description Language WSDL).

Otros sistemas interactúan con el servicio Web de una manera prescrita por su descripción

- ☐ usando mensajes SOAP (Simple Object Access Protocol)
- ☐ típicamente transmitido a través de HTTP (Hypertext Transfer Protocol)
- ☐ Con una serialización XML (Extensible Markup Language) en conjunción con otras normas relacionadas.

*www.w3.org (W3C - The World Wide Web Consortium, 2010)*

## URI, URL y URN

---

*El URI (Uniform Resource Identifier) es un identificador de recurso general, cuyo valor puede ser una URL o una URN.*

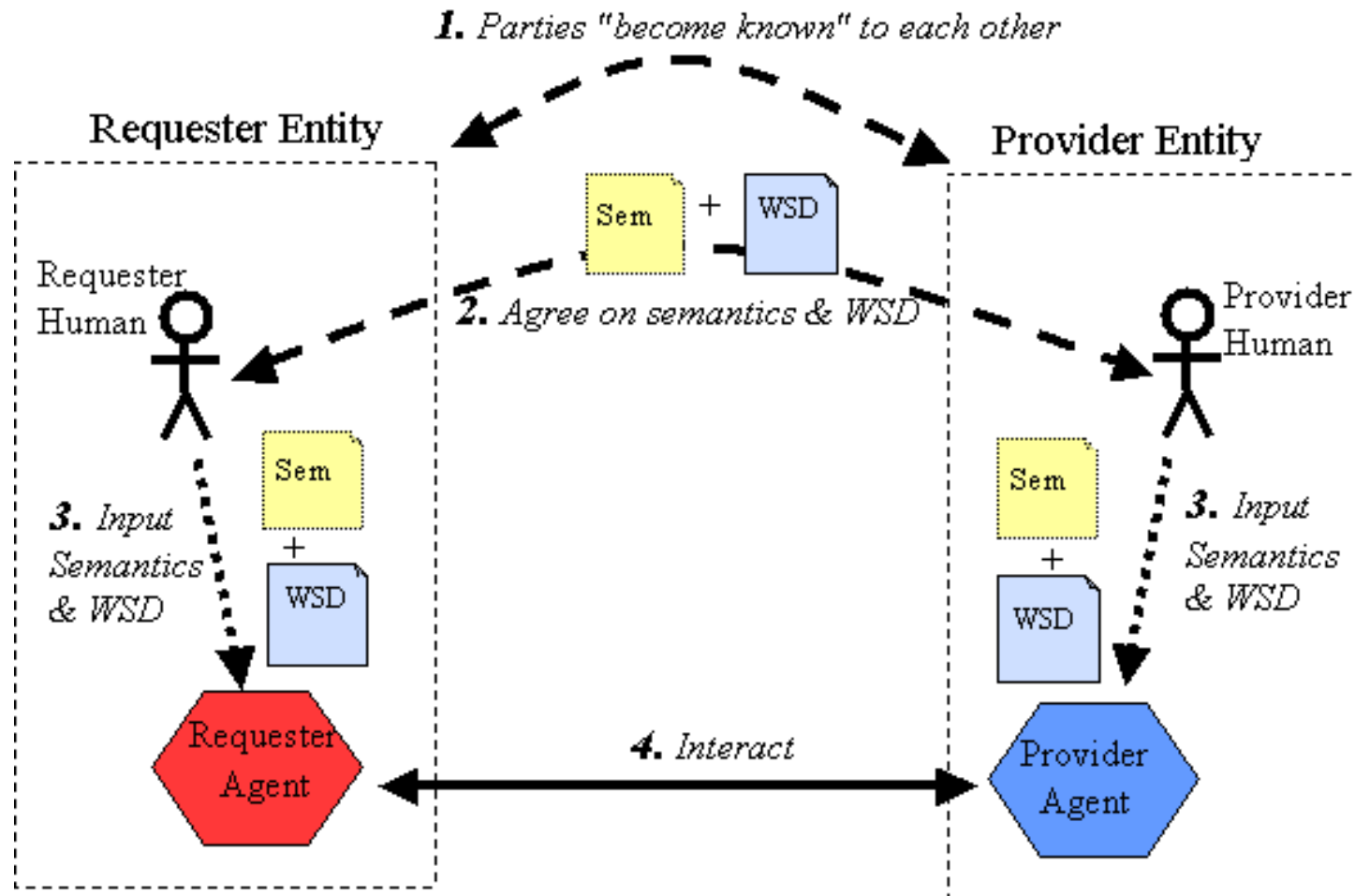
*Las URL, incluyen información de ubicación de recursos como el nombre de dominio del servidor de un recurso, son bien conocidas por los usuarios de la web.*

*<http://www.una.py/>*

*Los nombres de recursos uniformes (URN) no dependen de la ubicación; dependen de un servicio de búsqueda para asignarlos a las URL de los recursos. Ej: [urn:isbn:0451450523](#)      [urn:ISSN:0167-6423](#)*



# Web Services



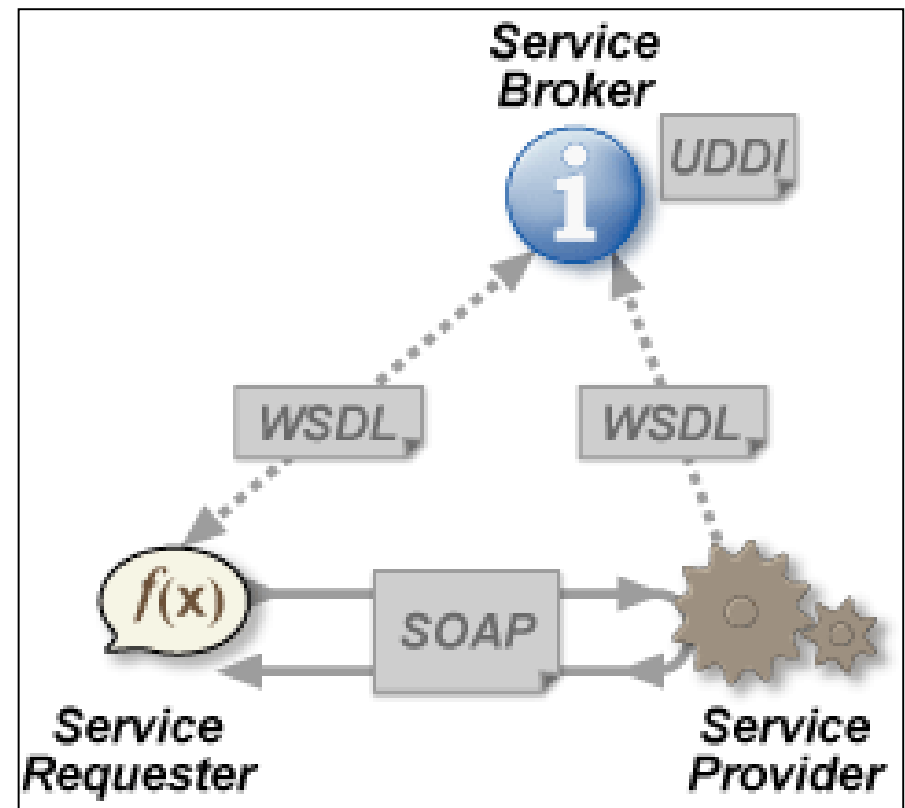
<http://www.w3.org/>

# Protocolos

---

Su diseño está basado en mensajes sobre protocolos abiertos, se basan en tecnologías tales como:

- ☐ HTTP
- ☐ XML
- ☐ SOAP
- ☐ WSDL
- ☐ SPARQL, y otros.



# XML – Extensible Markup Language

---

Es un lenguaje extensible de etiquetas desarrollado por el (W3C).

Permite definir una gramática de lenguajes específicos de forma extensible y personalizable.

No ha nacido sólo para su aplicación sobre Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier aplicación en representación de datos.

En la actualidad permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

# XML – Extensible Markup Language

---

```
<?xml version="1.0" ?>
<!doctype email system "http://www.sitio.es/DTDs/email.dtd">
<email id="E1X108">
  <head>
    <from>
      <name>Jesús Vegas</name>
      <address>jvegas@infor.uva.es</address>
    </from>
    <to>
      <name>Fulanito</name>
      <address>fulanito@unsitio.es</address>
    </to>
    <subject>Introducción a XML</subject>
  </head>
  <body>
    <p>Este es el guión de la conferencia sobre XML. Mira a
ver qué te parece. Saludos, jvegas.</p>
    <attach encoding="mime" name="ixml.html"/>
  </body>
</email>
```

# XML – Extensible Markup Language

---

Ofrece un formato de datos estándar, flexible y extensible.

Reduce significativamente la carga de la implementación de las muchas tecnologías necesarias para garantizar el éxito de los servicios Web.

El conjunto de información XML no es un formato de datos en sí, sino un conjunto formal de elementos de información y sus propiedades asociadas que conforman una descripción abstracta de un documento XML.

# SOAP (Simple Object Access Protocol)

---

SOAP es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

Proporciona un framework (marco de trabajo) estándar y extensible para el empaquetado e intercambio de mensajes XML.

En el contexto de los Web Services, SOAP también proporciona un mecanismo conveniente para referenciar capacidades o estructuras (normalmente por el uso de los encabezados).

# SOAP (Simple Object Access Protocol)

---

Los mensajes SOAP pueden ser transportados por una variedad de protocolos de red, tales como HTTP, SMTP, FTP, RMI / IIOP, o un protocolo propio de mensajería.

En la definición, se cuenta con componentes opcionales:

- ☐ Un conjunto de reglas de codificación para expresar instancias de tipos de datos definidos por la aplicación.
- ☐ Una convención para representar llamadas a procedimiento remoto (RPC) y respuestas.
- ☐ Un conjunto de reglas para el uso de SOAP con HTTP.

# WSDL (Web Services Description Language)

---

WSDL (Web Services Description Language) es un lenguaje para describir servicios Web.

Es un lenguaje de interfaz pública para los servicios Web de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.

Describe servicios Web a partir de los mensajes que se intercambian entre los agentes del request (solicitante) y el provider (proveedor). El mensaje en sí se describe de forma abstracta y luego relacionado a un protocolo de red y formato de mensajes concreto.



## WSDL (Web Services Description Language)

---

Las definiciones de Web Services pueden ser asignadas a cualquier lenguaje de implementación, plataforma, modelo de objetos, o sistema de mensajería.

Extensiones simples sobre la infraestructura de Internet existente pueden implementar servicios Web para la interacción a través de navegadores o directamente dentro de una aplicación.

La aplicación puede ser implementada usando librerías como por ejemplo COM, JMS, CORBA, COBOL, etc. Mientras que el emisor y el receptor estén de acuerdo en la descripción del servicio (por ejemplo, el archivo WSDL), detrás de las implementaciones de los servicios Web pueden realizar las acciones que desean.

# UDDI: Universal Description, Discovery and Integration

---

UDDI es un proyecto propuesto por Ariba, Microsoft e IBM adoptado por OASIS (Organización para el Avance de Estándares de Información Estructurada)

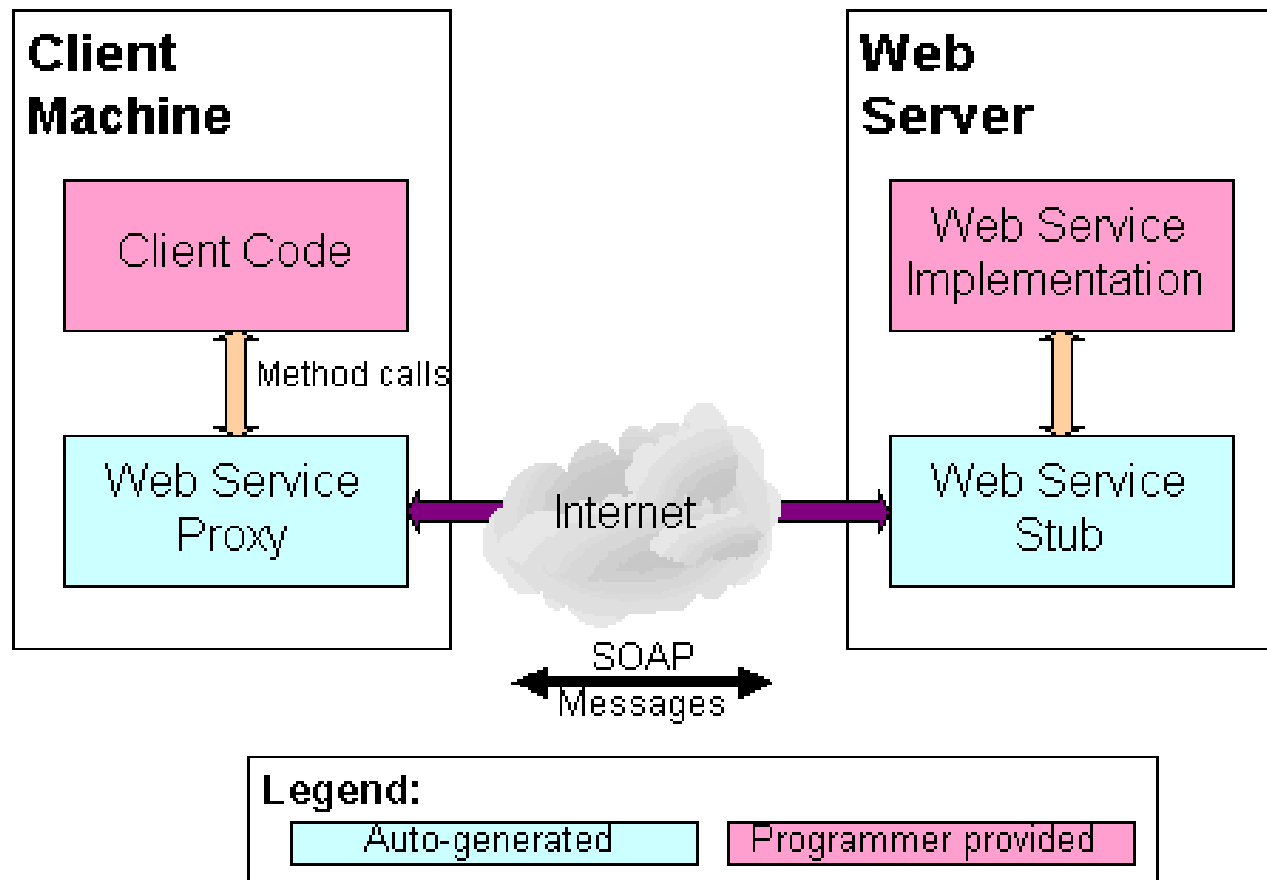
Es un estándar para registrar y descubrir los Web Services.

Las empresas registras sus servicios. Dicho registro está basado en XML

La idea es una imputación en el registro de empresas UDDI, un servicio centralizado y replicado a distintos nodos en forma regular, quedando disponible para ser descubierta por otras empresas.

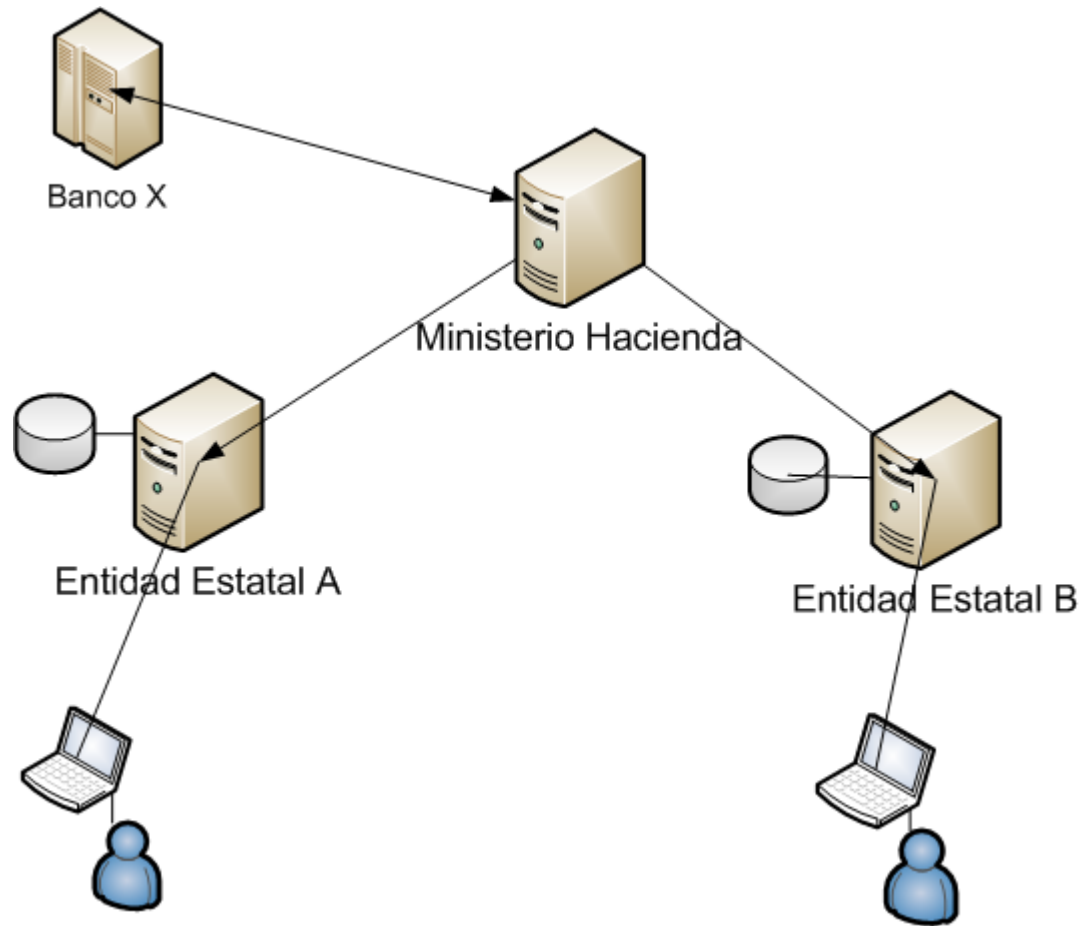
# Funcionamiento

---



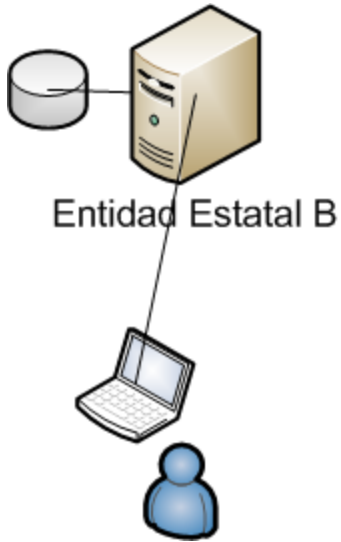
# Funcionamiento

---



# Funcionamiento

---



Liquidacion liq = **liquidarSalario**(Funcionario, unFuncionario,  
Periodo unPeriodo)

Salario salario = **consultarSalario**(Funcionario unFuncionario)

Boolean resultado = **vincularFuncionario**(Funcionario unFuncionario,  
String codigoEntidad)

Boolean resultado = **desVincularFuncionario**(Funcionario unFuncionario,  
String codigoEntidad)

## Origenes y Utilidad de los WebServices

---

**Google:** El producto denominado "Google Web API" permite a los desarrolladores poder interrogar y tomar información de casi tres mil millones de documentos Web directamente desde Google. Para lograr eso, Google usa SOAP y WSDL de forma que los desarrolladores puedan programar en su entorno favorito tal como PHP, Java, Perl, o Visual Studio .NET.

**NetworkSolutions** si queremos que nuestro programa sepa a quién pertenece un determinado dominio de Internet es un Servicio Web prestado por NetworkSolutions.

**Barnes and Noble:** Si queremos que nuestra aplicación sepa el precio de un determinado libro dado su ISBN es un Servicio Web que ofrece Barnes and Noble

**Dollar rent-a-car:** automatizó sus sistemas de reservas, que ya se podían hacer vía web.. se conecten con los de Southwest Airlines, de manera que en el mismo sitio de SWA los pasajero al reservar sus pasajes puedan reservar autos.

**Líneas Aéreas Escandinavas:** Estas líneas aéreas han desarrollado un servicio web que permite a los usuarios comprar billetes y chequear el estado de los vuelos, mediante el uso del teléfono móvil.

**Microsoft:** MapPoint .Net. Mediante este servicio, el usuario puedo conocer su localización exacta y otros datos adicionales relacionados con su posición actual, como información de tráfico, rutas posibles o puntos comerciales cercanos.

## Utilidad de los WebServices

---

Los servicios web de **Amazon** [[associates.amazon.com](https://associates.amazon.com)] pueden ser accedidos por SOAP o por REST. Esto permite a las aplicaciones de terceros crear servicios de valor añadido sobre los proporcionados por Amazon.com.

Por ejemplo, una aplicación de control de inventario y de compras podría ordenar suministros de varios productos según se necesiten de Amazon.com y realizar un seguimiento automático del estado cambiante de cada pedido. Más de 50.000 desarrolladores se registraron para usar estos servicios web en los primeros dos años después de su introducción [Greenfield y Dornan 2004].



---

REST [Fielding 2000]

**REST**

# REST

---

La característica clave de la mayoría de los servicios web es que pueden procesar mensajes SOAP con formato XML. Una alternativa es el enfoque REST. Cada servicio web utiliza su propia descripción de servicio para tratar las características específicas del servicio de los mensajes que recibe.

REST (Representational State Transfer) • REST [Fielding 2000] es un enfoque con un estilo de operación muy limitado, en el que los clientes usan URL y operaciones HTTP: GET, PUT, DELETE y POST para manipular recursos representados en XML.

# REST

---

El énfasis está en la manipulación de recursos de datos en lugar de en interfaces. Cuando se crea un nuevo recurso, tiene una nueva URL por la que se puede acceder o actualizar.

Los clientes reciben todo el estado de un recurso en lugar de llamar a una operación para obtener parte de ella. Fielding sostiene que en el contexto de Internet, la proliferación de diferentes interfaces de servicio no será tan útil como un simple conjunto mínimo de operaciones. Es interesante notar que, según Greenfield y Dornan [2004], el 80% de las solicitudes a los servicios web en Amazon.com son a través de la interfaz REST, y el 20% restante utiliza SOAP.

---

Web Services

# Implementaciones

## Implementaciones

---

Existen hoy en día muchas herramientas y lenguajes de programación que soportan el consumo y desarrollo de WebServices: Java, PHP, Python, la plataforma .NET, etc.

Como los WebServices se basan en XML es transparente para un consumidor la implementación del servicio.

Existen muchos servidores Web que incorporan frameworks para el desarrollo y publicación de WebServices.

# Java

---

La tendencia de utilizar Web-Services en Java radica en el uso de Servidores de Aplicaciones donde *deployarlos*, es decir, necesitamos un servidor donde alojar el “servicio web”.

Algunos de estos servidores pueden ser:

- ☐ Glassfish (<http://glassfish.java.net/> )
- ☐ Jboss (<http://www.jboss.org/> )
- ☐ Tomcat (<http://tomcat.apache.org/>)
- ☐ Oracle (<http://www.oracle.com/>)
  - ☐ Oracle WebLogic 11g
  - ☐ Oracle Application Server
  - ☐ OC4J

## Java – Protocolo SOAP

---

Existen IDE's que realizan la gestión y generación automática de WebServices, pero generalmente genera dependencia del servidor de aplicaciones elegido.

Como ejemplo de WebServices utilizando el IDE Netbeans, tenemos:

*<http://netbeans.org/kb/docs/websvc/jax-ws.html>*

Pueden utilizar el Servidor de aplicaciones Glassfish o Tomcat.

Buscar más información en: *<http://netbeans.org/features/web/web-services.html>*

# Python – Protocolo SOAP

---

Analizar Python Web Services

<http://pywebsvcs.sourceforge.net/>

Analizar librería SOAPpy



# Java – Servicios REST

---

## JAX-RS: Java API for RESTful Web Services

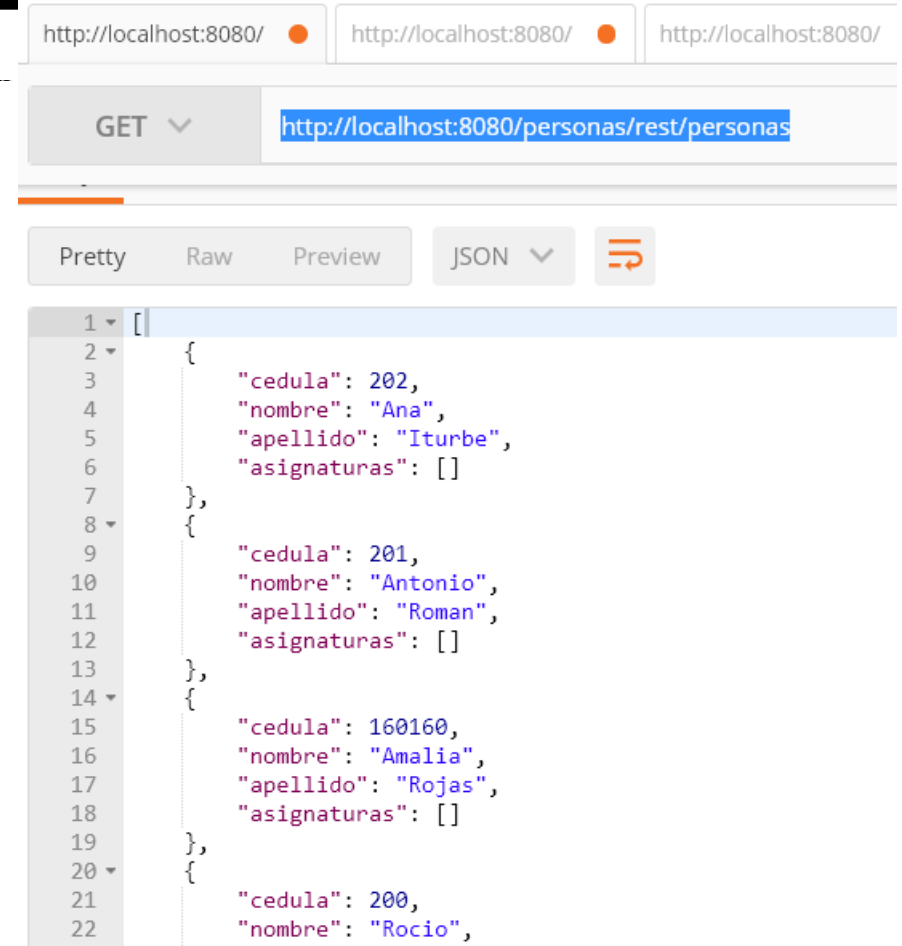
Es un API para desarrollar servicios REST en Java, incluye como parte de la especificación Java EE6.<http://docs.oracle.com/javaee/6/tutorial/doc/giepu.html>

La especificación se basa en Annotations.

Existen implementaciones como Jersey and RESTEasy. Una comparativa en <https://www.genuitec.com/jersey-resteasy-comparison/>

# Java – Servicios REST

```
40 /
47 @Path("/personas")
48 @RequestScoped
49 public class PersonaRESTService {
50
51     @Inject
52     private Logger log;
53
54     @Inject
55     PersonaService personaService;
56
57     @GET
58     @Produces({ MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML })
59     public List<Persona> listar() {
60         return personaService.seleccionar();
61     }
62
63     @GET
64     @Path("/{cedula:[0-9][0-9]*}")
65     @Produces(MediaType.APPLICATION_JSON)
66     public Persona obtenerPorId(@PathParam("cedula") long cedula) {
67         Persona p = personaService.seleccionarPorCedula(cedula);
68         if (p == null) {
```



http://localhost:8080/ http://localhost:8080/ http://localhost:8080/

GET <http://localhost:8080/personas/rest/personas>

Pretty Raw Preview JSON

```
1 [
2   {
3     "cedula": 202,
4     "nombre": "Ana",
5     "apellido": "Iturbe",
6     "asignaturas": []
7   },
8   {
9     "cedula": 201,
10    "nombre": "Antonio",
11    "apellido": "Roman",
12    "asignaturas": []
13  },
14  {
15    "cedula": 160160,
16    "nombre": "Amalia",
17    "apellido": "Rojas",
18    "asignaturas": []
19  },
20  {
21    "cedula": 200,
22    "nombre": "Rocio",
```

# Java – Servicios REST

---

|             |  |
|-------------|--|
| @HEAD       | The @HEAD annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP HEAD requests. The behavior of a resource is determined by the HTTP method to which the resource is responding.  |
| @PathParam  | The @PathParam annotation is a type of parameter that you can extract for use in your resource class. URI path parameters are extracted from the request URI, and the parameter names correspond to the URI path template variable names specified in the @Path class-level annotation.  |
| @QueryParam | The @QueryParam annotation is a type of parameter that you can extract for use in your resource class. Query parameters are extracted from the request URI query parameters.   |
| @Consumes   | The @Consumes annotation is used to specify the MIME media types of representations a resource can consume that were sent by the client.   |
| @Produces   | The @Produces annotation is used to specify the MIME media types of representations a resource can produce and send back to the client: for example, "text/plain".   |
| @Provider   | The @Provider annotation is used for anything that is of interest to the JAX-RS runtime, such as MessageBodyReader and MessageBodyWriter. For HTTP requests, the MessageBodyReader is used to map an HTTP request entity body to method parameters. On the response side, a return value is mapped to an HTTP response entity body by using a MessageBodyWriter. If the application needs to supply additional metadata, such as HTTP headers or a different status code, a method can return a Response that wraps the entity and that can be built using Response.ResponseBuilder. |

# Herramienta

---

## SOAP UI

Herramienta para realizar pruebas y testing SOA y Web Services.  
Intefaz gráfica de fácil uso.

Práctica:

- ✓ Creación de Web Services
  - ✓ obtenerPaises()
  - ✓ obtenerPais()
- ✓ Deploy en servidor Glassfish.
- ✓ Testear el WebService con SOAP UI

---

# REFERENCIAS

## Referencias

---

W3C - The World Wide Web Consortium Web Services Architecture

<http://www.w3.org/TR/ws-arch/>

BRAY Tim [y otros] Extensible Markup Language (XML) 1.0 (Fifth Edition) [En línea]. - W3C, 2004. - <http://www.w3.org/TR/REC-xml>.