Definición de Web Service

- Es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.
- Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos.

Características de los Web Services

- Independientes de la plataforma. Se necesita un formato de intercambio de datos neutro y de texto (no binario).
- Arquitectura cliente/servidor:
  - El servidor tiene un conjunto de operaciones disponibles.
  - El cliente puede solicitar la ejecución de una operación en el servidor.
- En la mayoría de los casos se usa HTTP como transporte.

Tipos de Web Services

- SOAP: Fuertemente basados en estándares.
  - Se transmiten los datos en formato XML a través de HTTP usando una petición POST.
  - Los datos indican el método y los parámetros.
  - Se genera una respuesta XML con el resultado del método.
  - Un documento WSDL (Web Services Definition Language) contiene las características del web service.

Tipos de Web Services

- REST: Más ligeros
  - Usan los comandos HTTP como nombre de los métodos (GET, POST, PUT, DELETE).
  - Hay diferentes URLs para diferentes recursos.
  - Los parámetros y los resultados principalmente en XML o JSON.

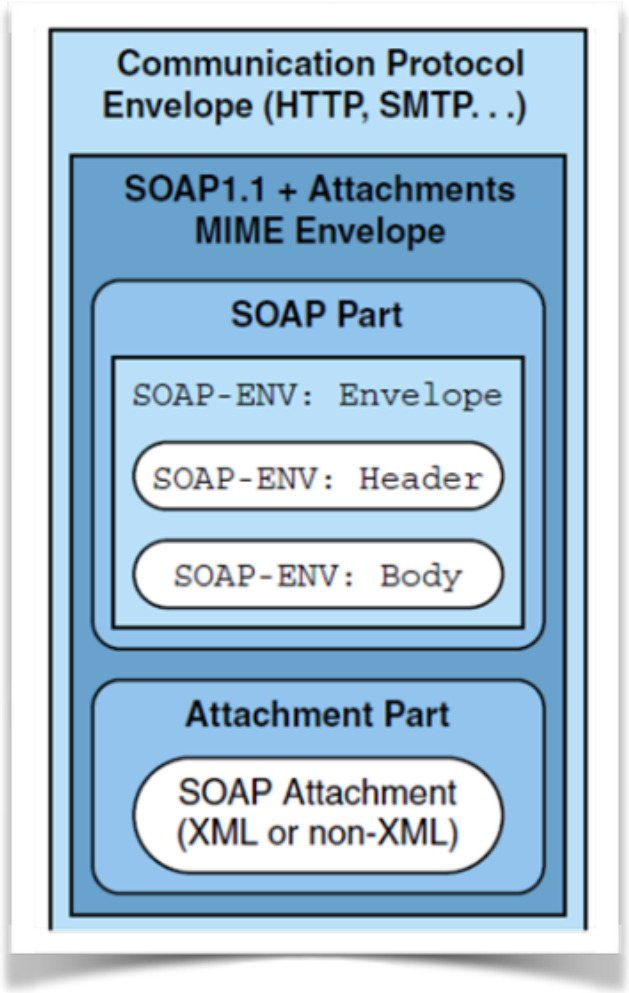
Implementaciónes Java de Web Services

- JAX-WS es el API para SOAP Web Services
  - Forma parte de Java SE.
  - Pero se necesita un servidor HTTP de producción para un entorno real.
  - GlassFish y WebLogic usan la implementación METRO.
- JAX-RS es el API para RESTful Web Services
  - No forma parte de Java SE
  - La implementación de referencia es Jersey.

Web Services en contenedores Java EE

- Los Web Services necesitan un contenedor web para poder dar servicio en un entorno real.
- Los servidores Java EE Full Profile soportan JAX-WS y JAX-RS por omisión.
- Si se necesitan transacciones, seguridad o cualquier otra característica proporcionada por un servidor Java EE se usarán EJBs o clases que delegan en EJBs.

Mensaje SOAP



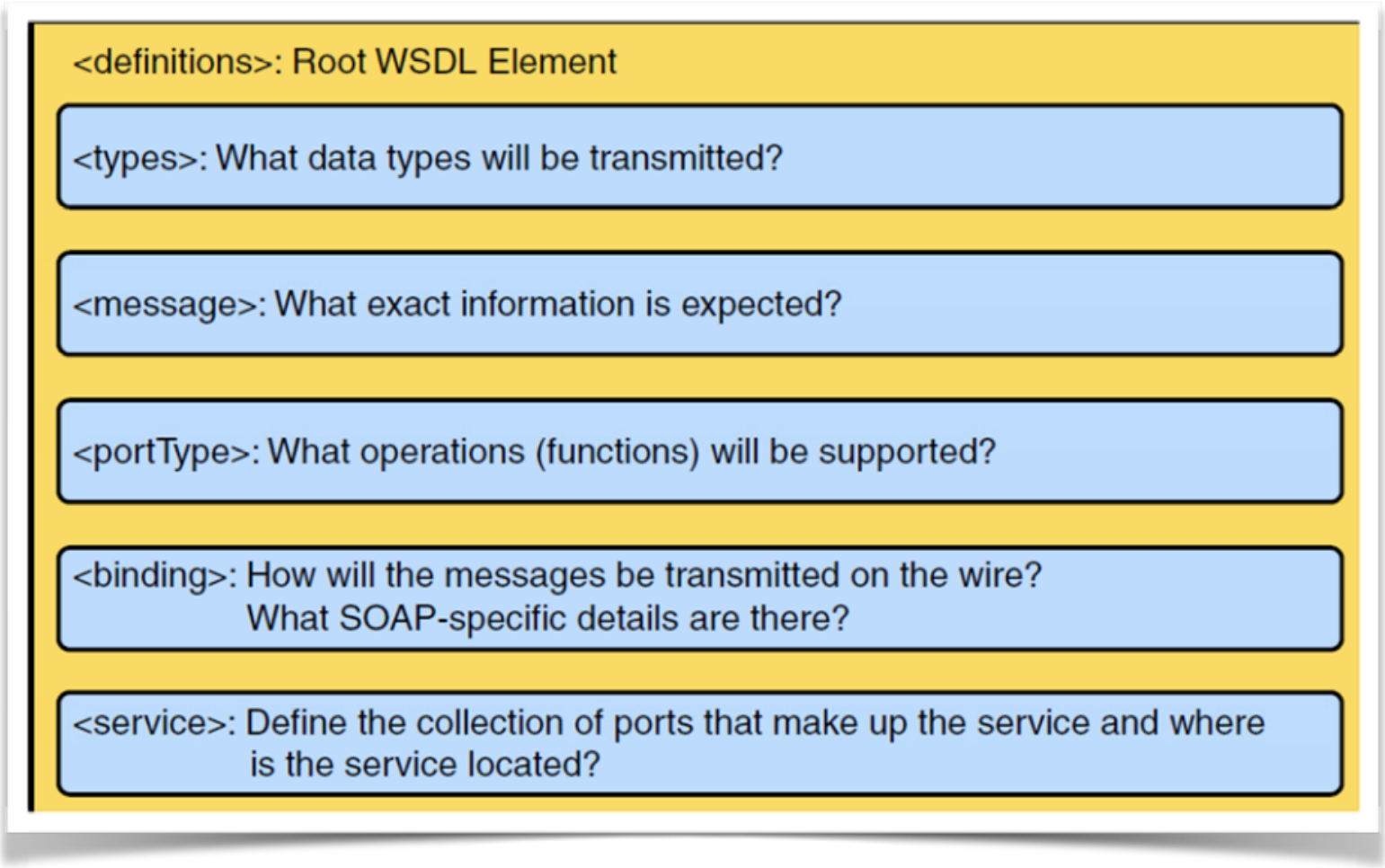
Petición SOAP

```
<?xml version="1.0" encoding=“UTF-8"?>
<S:Envelope xmlns:…”>
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:getOne
                xmlns:ns2=“http://predictions/">
            <arg0>1</arg0>
        </ns2:getOne>
    </S:Body>
</S:Envelope>
```

Respuesta SOAP

```
<?xml version="1.0" encoding=“UTF-8"?>
<S:Envelope xmlns:…”>
    <SOAP-ENV:Header/>
    <S:Body>
        <ns2:getOneResponse xmlns:ns2=“http:…”>
            <return>
                <id>1</id>
                <what>Managed holistic…</what>
                <who>Cornelius Tillman</who>
            </return>
        </ns2:getOneResponse>
    </S:Body>
</S:Envelope>
```

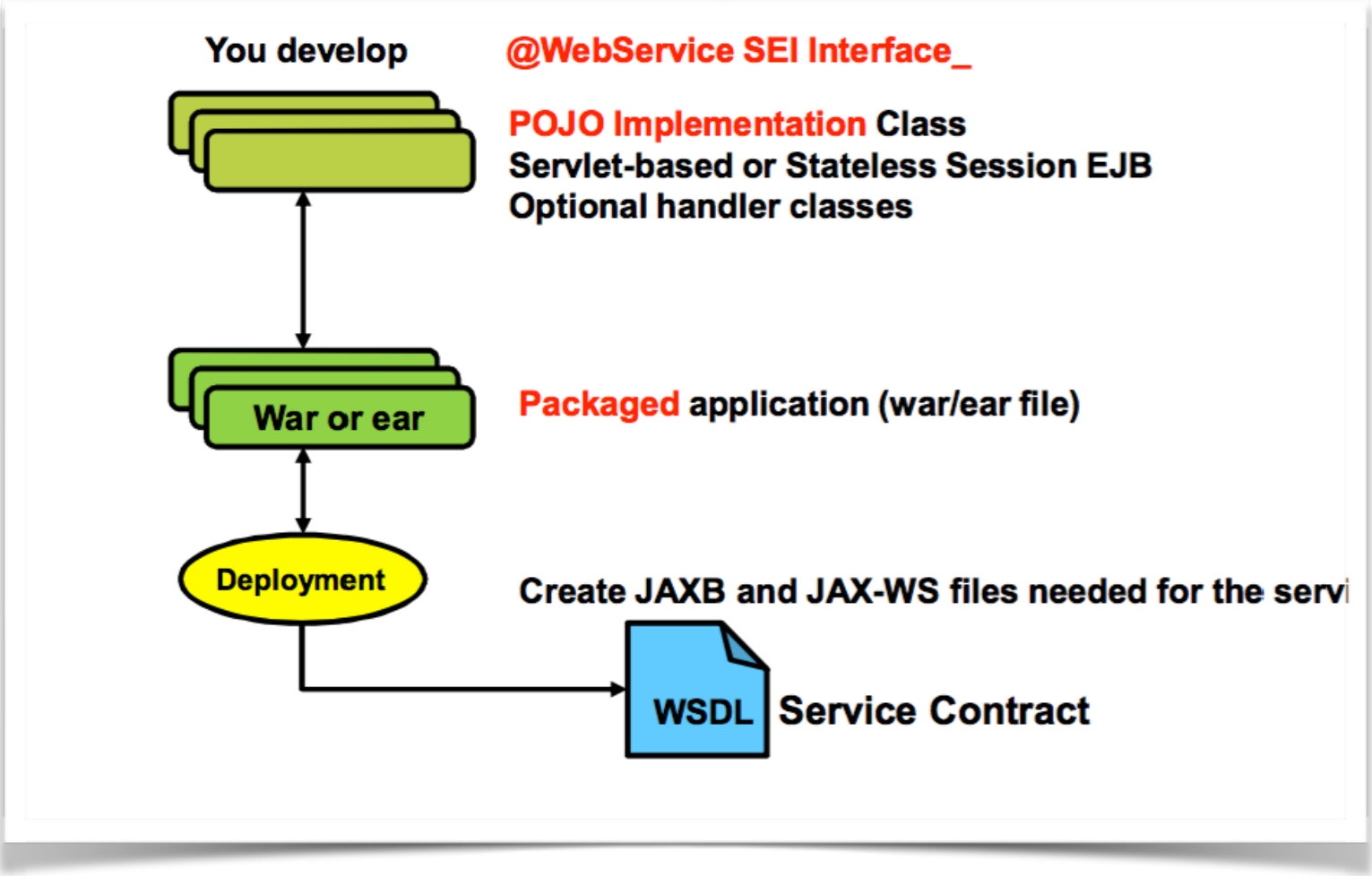
Estructura de un fichero WSDL



WS-I Basic Profile

Es una especificación que consta de un conjunto de especificaciones de servicios web no propietario junto con aclaraciones, ajustes, interpretaciones y ampliaciones de las especificaciones que promueven la interoperabilidad, como SOAP y WSDL (Wikipedia).

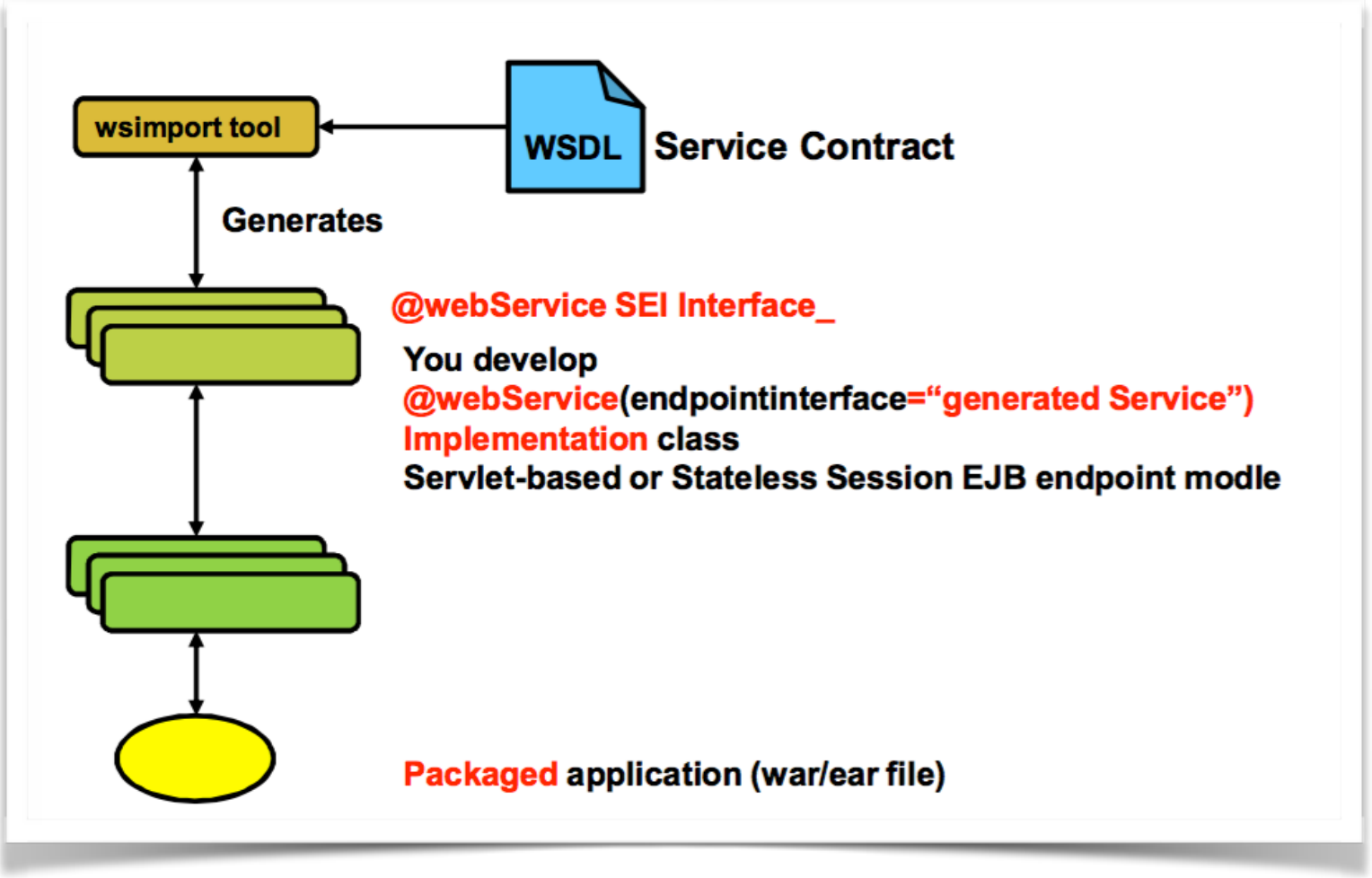
SOAP Web Services “Bottom-Up”



SOAP Web Services “Bottom-Up” Beneficios

- Desarrollo rápido sobre todo cuando se tiene ya la lógica implementada.
- Se puede mapear los modelos existentes a WSDL sin ningún esfuerzo.
- Se puede reusar una fachada de servicios como un mediador para la lógica de la aplicación para otros tipos de aplicaciones y servicios.

SOAP Web Services “Top-Down”



Creación de un cliente JAX-WS

- La implementación de referencia de JAX-WS se incluye en Java SE.
- Se pueden usar SOAP Web Services desde un cliente Java SE
- Se utiliza el comando `wsimport` para generar los artefactos en el cliente.
- La entrada a `wsimport` es el WSDL.

REST

- **RE**presentational **S**tate **T**ransfer.
- Originalmente, se refería a un conjunto de principios de arquitectura.
- Hoy en día se usa para describir cualquier interfaz web simple que utiliza XML y HTTP, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como SOAP.
- Es un estilo o arquitectura.

Recursos REST

- El concepto fundamental de REST es el de recurso.
- Cualquier ítem de información con nombre puede ser un recurso.
- Un recurso se identifica con un URI.
- Los clientes piden y envían representaciones de recursos.
- Puede haber distintas representaciones disponibles del mismo recurso.

Diseño de recursos

- Un RESTful web service contiene múltiples recursos.
- Los recursos se enlazan entre ellos.
- El diseño suele ser un árbol de recursos con un recurso básico en la raíz.
- Cada recurso soporta un número limitado de operaciones (GET, POST, PUT, DELETE).

Diseño de recursos

- Un recurso se identifica unívocamente por un URL:
  - http://localhost:8080/myapp/resources/users es un recurso asociado a una colección.
  - http://localhost:8080/myapp/resources/users/3 es un recurso específico de una colección.

Formatos de representación de recursos

- Un web service RESTful puede soportar uno o más formatos de representación.
- Se usan las cabeceras Accept y Context-Type para negociar e identificar los recursos.
- Formatos más habituales: XML y JSON.

Comandos e idempotencia

- Un comando HTTP es idempotente si al volverse a ejecutar no produce efectos secundarios.
- **GET**: Lectura - idempotente.
- **PUT**: Inserción o actualización - idempotente.
- **DELETE**: Borrado - idempotente.
- **POST**: “Todo vale” (normalmente inserción) - no idempotente.