Cursos de Extensión Universitaria
UNIVERSIDAD DE OVIEDO

PROGRAMACIÓN ORIENTADA A OBJETOS CON C# EN LA PLATAFORMA .NET

Servicios Web (II)

César Fernández Acebal
acebal@ieee.org

Dpto. de Informática
OOTLab - Laboratorio de Tecnologías Orientadas a Objetos
www.ootlab.uniovi.es

Creación de un cliente de un servicio Web

Retomaremos la clase de ayer estudiando las distintas formas que tenemos de crear un servicio Web.

Más concretamente, nos centraremos en las facilidades que Visual Studio y la plataforma .NET proveen para ello.

Servicio Web: HolaMundo

- En efecto, tomaremos el servicio Web ya creado, `HolaMundo`, y trataremos de hacer un cliente para él
- En primer lugar, atendamos a la documentación generada para dicho servicio

```
hhtp://localhost/HolaMundo/HolaMundo.asmx
```

Espacio de nombres

- La página de documentación del servicio nos da una lista de todas las operaciones disponibles
- También nos dice cómo podemos cambiar el espacio de nombres temporal a otro diferente, si queremos publicar el servicio Web
- Para ello se usa el atributo WebService

```
[WebService(Namespace="http://cesaracebal.com/webservices/")]
public class MiServicioWeb { … }
```

Descripción del servicio (WSDL)

- Pulsando en el enlace `Service Description`, nos muestra el fichero WSDL generado automáticamente por .NET para ese servicio

  `hhtp://localhost/HolaMundo/HolaMundo.asmx?WSDL`

Cliente de prueba

- Además, para cada método del servicio Web, .NET genera automáticamente una página de documentación para él
- Dicha página incluye un cliente que nos permiten probar el servicio Web
- Así como documentación de cómo serían los distintos tipos de clientes

Tipos de clientes

Veamos cuáles son los distintos tipos de clientes de un servicio Web.

Caso 1: HTTP GET

- El servicio Web acepta una simple petición HTTP GET
  - Igual que una petición de página HTML, sólo que en vez de solicitar un página HTML se indica el URL de un servicio

    ```
    http://localhost/HolaMundo/HolaMundo.asmx/Saludar
    ```
- Para probarlo, bastaría con introducir dicho URL en el navegador
  - El resultado es el XML generado por ASP .NET para envolver el resultado de dicho método, tal y como decíamos ayer

Caso 1: HTTP GET (II)

- Cuando la petición llega al servidor, éste detecta que es para ASP .NET y se la pasa
- ASP .NET analiza los parámetros de la cadena del URL, crea el objeto `HolaMundo` y realiza una llamada al método `Saludar`
- Finalmente, toma el valor de retorno de este método, lo envuelve en XML y devuelve el mensaje así formado al cliente

Caso 1: HTTP GET. Parámetros

- ¿Qué pasa si el método tomase parámetros?
- Usando GET, los parámetros se envían con el propio URL
  - Lo habréis observado en numerosas páginas Web, cosas del estilo de:

```
http://localhost/Calculadora/Calculadora.asmx/Sumar?x=3?y=5
```

- Cualquier formulario Web nos serviría para probar el servicio

```
<form
  action="http://localhost/Conversor/ConversorEuros.asmx/Cambiar"
  method="get">
  <p>Cantidad a cambiar: <input name="cantidad" size="9"/></p>
  <p><input type="submit" value="Convertir"/></p>
</form>
```

Caso 2: HTTP POST

- HTTP también acepta otro método de enviar peticiones, conocido como POST
  - La diferencia es que los parámetros no van con el URL sino dentro de la petición en sí
- Para probarlo podemos hacer un formulario HTML:

```
<form
  action="http://localhost/Conversor/ConversorEuros.asmx/Cambiar"
  method="post">
  <p>Cantidad a cambiar: <input name="cantidad" size="9"/></p>
  <p><input type="submit" value="Convertir"/></p>
</form>
```


Caso 3: SOAP

- En realidad, empleando .NET, este tercer método podríamos dividirlo en varios, atendiendo al modo de implementación
- Pero antes, necesitamos ver algunas nociones de SOAP, para entender su relación con los servicios Web

SOAP

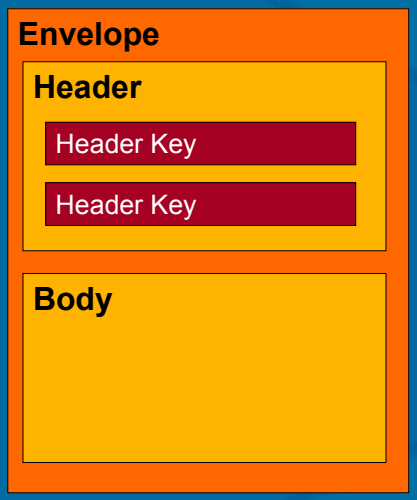
Muy rápidamente, daremos una noción de qué es SOAP y cómo se usa con los servicios Web.

Más importante que eso es comprender que, a pesar de lo que se lee habitualmente, los servicios Web no están ligados en modo alguno a SOAP, y ya hay estilos alternativos de intercambio de mensajes (verbigracia, REST).

Introducción

- SOAP = Simple Object Access Protocol
- Es un vocabulario de XML que describe llamadas a métodos y sus parámetros
  - Similar a XML-RPC
- Es un estándar del W3C
  - www.w3.org

Anatomía de un mensaje SOAP



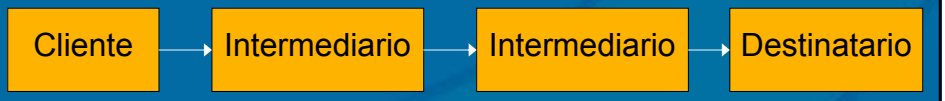
- Un mensaje SOAP sigue la metáfora del sobre
- La cabecera (opcional) proporciona información auxiliar
  - Remitente, destinatario…
  - Qué hacer en caso de error
  - Cualquier cosa que dé contexto al mensaje sin ser parte de él
- El cuerpo envuelve al mensaje en sí que se quiere transmitir

Anatomía de un mensaje SOAP

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
      <!-- Aquí va información opcional -->
      <To>Kent</To>
      <From>Erich</From>
    </soap:Header>
    <soap:Body>
      <!-- Aquí va el mensaje -->
    </soap:Header>
</soap:Envelope>
```

Actores de SOAP

- Un actor de SOAP es cualquier cosa que actúa sobre el contenido de un mensaje SOAP
  - El actor predeterminado es el receptor final pretendido del mensaje
  - Un intermediario recibe un mensaje SOAP podría actuar sobre él antes de reenviarlo por la ruta propia del mensaje



Elemento 'Header'

- El elemento opcional `Header` se utiliza para pasar datos que podría no ser apropiado codificar en el cuerpo
  - Por ejemplo, si el mensaje está comprimido, habría que indicar aquí el algoritmo de compresión
- Otros usos:
  - Autenticación
  - Información de seguridad
  - Información de enrutamiento
  - Transacciones
  - Información de pago
  - etc.

Elemento 'Body'

- Un mensaje SOAP válido debe tener un elemento `Body`
- Contiene el mensaje en sí a transmitir
- No hay ninguna restricción
  - Podría ser una simple cadena de caracteres, un array de bytes codificado o XML
  - El único requisito es que no invalide el documento XML resultante
    - Es decir, que el mensaje SOAP sea un documento XML válido

Características de SOAP

- No está asociado a ningún lenguaje
- Ni a ningún protocolo de transporte
  - Aunque lo habitual será usar HTTP, dado que un mensaje SOAP no es más que un documento XML, podrá enviarse con cualquier protocolo que permita transmitir texto
- No está ligado a ninguna infraestructura de objetos distribuidos
- Aprovecha los estándares existentes en la industria
  - XML, Esquema XML, protocolos (HTTP, SMTP…)

Categorías de mensajes SOAP

- Hay dos tipos de mensajes que podemos intercambiar (y podríamos hablar también así de dos tipos de servicios Web):
  - los orientados a procedimientos
    - Estilo RPC (Remote Procedure Call)
  - los orientados a documentos
    - Cuando sólo queremos intercambiar un documento XML tal cual
    - Aunque SOAP también permite este estilo, parece más orientado al primer tipo de servicios

Caso 3: SOAP

Una vez adquiridas unas nociones básicas de SOAP, pasemos a ver cómo podemos hacer clientes SOAP con .NET

Caso 3.1 SOAP “a mano”

- La primera opción sería ver cuál es el formato de mensaje SOAP que debe enviarse al servicio Web y hacerlo a mano
  - Generar primero el mensaje SOAP
  - Transmitirlo luego vía HTTP
- Con ser el método más tedioso, resulta no obstante facilitado por las clases que incorpora la biblioteca de .NET y que permiten lidiar con:
  - Los mensajes SOAP
  - La comunicación HTTP

Caso 3.2 Creación de un *proxy*

- Ésta será la forma más habitual de trabajar con servicios Web
- La propia herramienta de desarrollo (en este caso Visual Studio) genera una clase *proxy* que encapsula todos los detalles de la implementación
- Nosotros simplemente llamaríamos a los métodos de esa clase local

Creación de un proxy en Visual Studio .NET

- Creamos un nuevo proyecto, C#, vacío
- Añadimos una referencia Web
  - Nos aparece una ventana donde podemos:
    - O bien buscar un servicio Web (a través de UDDI, el mecanismo de descubrimiento de servicios Web)
    - O bien introducir directamente la dirección del servicio
  - Naturalmente, como no hemos publicado el servicio, tendremos que escoger esta segunda opción:

    `http://localhost/ServiciosWeb/HolaMundo/HolaMundo.asmx`
  - Añadimos la referencia

Creación de un proxy en Visual Studio .NET

- Para ver la clase generada, seleccionamos la opción de ver todos los ficheros
  - (el segundo botón de los que aparecen en el explorador de soluciones)
- Dicha clase provee:
  - Un método de invocación síncrona `X` para cada método `X` del servicio Web
  - Un par de métodos `BeginX` y `EndX` para la invocación asíncrona

Ejercicios

Hacer un cliente de Hola Mundo empleando una clase Proxy.

Después, crear otro servicio Web a elección del alumno y su correspondiente cliente (podría reutilizarse alguno de los ejercicios ya vistos durante el curso y darles una interfaz de servicio Web).