Docencia de Arquitectura Orientada a Servicios

P. García-Sánchez, J. González, P. Castillo, M.G. Arenas
Departamento de Arquitectura y Tecnología de los Computadores
Universidad de Granada
ETS. Informática y Telecomunicación, C/Periodista Daniel Saucedo s/n
18071 Granada
{pgarcia,jesus,pedro,maribel}@atc.ugr.es

M. A. López
Fundación I+D del Software Libre
BIC Granada-CEEI
Avenida de la Innovación, 1
18100 Granada
malopez@fidesol.org

Resumen

Este trabajo presenta los contenidos del curso "Web 2.0: Arquitectura Orientada a Servicios en Java" de la Escuela de Posgrado de la Universidad de Granada. El objetivo del curso es familiarizar al alumno con la programación de Servicios Web. Dada la gran variedad de técnicas disponibles para utilizar Arquitectura Orientada a Servicios, se presentan los siguientes temas: utilización de protocolos bien definidos para comunicación y contrato (SOAP y WSDL), creación de Web Services con JAX-WS y orquestación de Servicios Web con BPEL. Al final del curso, el alumno será capaz de crear, utilizar y mantener Servicios Web para el desarrollo de aplicaciones interempresariales, utilizando servicios creados o ya disponibles en la web, así como la orquestación lógica de los mismos.

Summary

This work presents the contents of the course "Web 2.0: Service Oriented Architecture on Java" from the Graduate School of the University of Granada. The course objective is to familarize students with Web Services programming. Due to the wide variety of available technologies, several subjects are presented: the usage of well-defined protocols to contract and communication (SOAP and WSDL), web services creation using JAX-WS, and service orchestration with BPEL. At the end of the course, students will be capable to create, use and manage Web Services for business applications, using new or available services in the web, and also their logical orchestration.

Palabras clave

Arquitectura Orientada a Servicios, Web Services, Java, BPEL, Curso

1. Introducción

Los problemas más comunes en el desarrollo del software suelen ser la incompatibilidad entre aplicaciones, modelos de datos, lenguajes de programación y sistemas de comunicación, lo que obliga a rediseñar todas las aplicaciones y reescribirlas para que operen entre sí. Por lo tanto una aplicación que desee crecer en un futuro debería obviar características restrictivas y partir de un buen diseño que permita la extensibilidad y la comunicación con el mayor nivel de abstracción posible. De esta idea surge la Arquitectura Orientada a Servicios (SOA) [5, 9], ya que se hace necesaria una forma de comunicación eficiente y escalable, independiente del lenguaje de programación y plataforma de cada una de las aplicaciones que deseen intercomunicarse. Los elementos básicos que conforman SOA son:

- Proveedores de servicios: Una aplicación expone operaciones que cualquier otra puede usar
- Consumidores de servicios: Utilizan las operaciones de los proveedores para obtener información
- Bus de servicios empresariales: para integrar los servicios de forma lógica y ampliable.

Estos servicios, llamados Servicios Web (Web Services), son un sistema software diseñado para soportar interacción Máquina a Máquina sobre una red, es decir, son interfaces (APIs) que pueden ser accedidas remotamente. Utilizan protocolos para comunicación bien definidos, como SOAP y sus interfaces se publican utilizando el formato WSDL (donde se indican las operaciones y tipos de dato que se pueden intercambiar), siendo su implementación realizada cualquier lenguaje. Esto permite por ejemplo que una aplicación escrita en Java reciba información generada por otra aplicación realizada con C++, PHP o cualquier otro lenguaje de programación.

Debido al auge de estas tecnologías en el mundo empresarial se proyectó la realización del curso

"Web 2.0: Arquitectura Orientada a Servicios en Java", organizado por la Escuela de Posgrado de la Universidad de Granada, para familiarizar a los estudiantes de carreras técnicas (sobre todo los de Ingeniería en Informática o Telecomunicación)[1] en estas tecnologías, ya que no están presentes en el plan de estudios de la Universidad de Granada. Sin embargo, es tal la adaptación de la Arquitectura Orientada a Servicios que otras universidades, como la Universidad Autónoma de Barcelona, ya cuentan con un plan docente basado este paradigma, el Grado en Informática y Servicios [2].

El resto del artículo se estructura como sigue: inicialmente se introduce el concepto de Arquitectura Orientada a Servicios. A continuación, en la Sección 3, se presentan los contenidos del curso y las tecnologías utilizadas (Java, XML, PHP, JAX-WS y BPEL), para finalmente mostrar la recepción del curso por parte de los alumnos y las conclusiones a este trabajo.

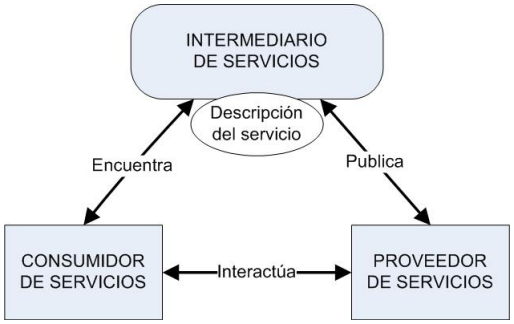

Figura 1: Esquema de interacción de servicios. El proveedor de servicios publica una descripción del servicio que es utilizada por el consumidor para encontrar y usar servicios.

2. Introducción a los Servicios Web

Actualmente las Arquitecturas Orientadas a Servicios (*Service Oriented Architecture*, SOA) están en auge, debido a los beneficios que proporcionan a la hora de desarrollar e integrar aplicaciones distribuidas o modulares. El principal concepto de SOA es el de *servicio*. Podemos ver un servicio como una llamada a una función, que se ejecutará local o remotamente, y que es independiente del lenguaje de programación y plataforma en la que se ejecuta. Este servicio consta de una interfaz bien definida y que depende de la tecnología que se desea utilizar para implementar SOA.

El resto de elementos básicos que conforman SOA, y cuyas relaciones pueden verse en la Figura 1, son los Proveedores, Consumidores y Publicadores de Servicios. El *Proveedor de Servicios* es un ente (nodo, clase, programa, etc.) que brinda un servicio en respuesta a una llamada o petición desde un *Consumidor de Servicios*. Éste utiliza el *Publicador de Servicios* para obtener información sobre los servicios que estén disponibles para su uso y sobre las interfaces (*Descripción del servicio*) para invocarlos.

A la hora de desarrollar sistemas software se hace necesario que sean compatible con las implementaciones SOA más extendidas, como por ejemplo los *Servicios Web* (*Web services*) [9]. Su arquitectura está diseñada para soportar interacción máquina a máquina sobre una red, utilizando sobre todo el protocolo SOAP (*Simple Object Access Protocol*)[13] para transmitir mensajes entre los diferentes computadores. Las interfaces de los servicios están descritas en WSDL (*Web Service Description Language*) [12], un lenguaje basado en XML que proporciona un modelo para describir Servicios Web y la manera de comunicarse utilizando éstos (equivale a *Descripción del servicio* en la Figura 1). Estos servicios pueden ser listados usando UDDI (Universal Description, Discovery, and Integration) [7], un registro basado en XML independiente de la plataforma (*Intermediario de servicios* en la Figura 1).

En un ambiente SOA, los nodos de la red suelen poner disponibles sus recursos a otros participantes en la red como servicios independientes, a los que tienen acceso de un modo estandarizado. La mayoría de las definiciones de SOA identifican la utilización de Servicios Web (empleando SOAP y WSDL) en su implementación, pero no obstante se puede implementar SOA utilizando cualquier tecnología basada en servicios, como por ejemplo, OSGi [8].

[1] Cualquiera se puede apuntar, pero por los conocimientos previos que se recomiendan, está orientado especialmente a alumnos de esas dos titulaciones.

3. Temario

Esta sección presenta los contenidos teóricos y prácticos del curso. El curso dura 50 horas, divididas en cinco bloques: Introducción a los Servicios Web y Java (10 horas), XML (5 horas), Creación de servicios web en Java utilizando Jax-WS (15 horas), PHP (5 horas), Orquestación de Servicios Web con BPEL (10 horas) y finalmente un día para presentar otras arquitecturas SOA y examen (5 horas).

3.1. Introducción a los Web Services y a Java

Como introducción al curso se le presenta al alumno de forma sucinta el funcionamiento de los Servicios Web, para luego profundizar en una introducción al lenguaje Java, ya que será el que se utilizará durante el curso. En esta sección se explican las peculiaridades del lenguaje y se realizan distintos ejercicios simples para comprender el funcionamiento y uso. Asimismo se explica el uso de NetBeans [2], entorno de desarrollo libre usado para los ejercicios en el curso y algunas de las funcionalidades básicas (compilación, depuración, creación de proyectos...).

3.2. XML

En la segunda parte del curso, se les muestra a los alumnos en qué consiste XML (Extensible Markup Language) junto con herramientas para su manejo y ejemplos prácticos variados de utilización en diversos ámbitos de la programación web.

El orden del temario que se incluye en esta parte es el siguiente:

- Introducción: Simplemente se presenta al docente, sus datos, datos de donde encontrar la documentación necesaria para esta parte y se sitúa esta parte del curso dentro de la totalidad del temario.
- XML: Donde se introduce en qué consiste XML, su utilidad, ventajas frente a otros lenguajes de marcas y cada uno de las partes que lo componen.
- Protocolos XML: Donde se mencionan varios de los protocolos de comunicación más utilizados en la actualidad y que están basados en XML, como son: XML-RPC, SOAP y RSS.
- XML y Java: Java es la herramienta que se utiliza en todas las partes del curso para programar, por lo que en esta parte se introduce lo que ofrece Java para tratar documentos XML. Se mencionan aspectos básicos para el tratamiento de la información como el análisis de la formalidad de los documentos, la aplicación de "plantillas" a los documentos para comprobar si están correctamente formados, etc. También es en esta parte donde se mencionan la gran cantidad de herramientas que Java proporciona para tratar XML. Y concretamente, se detalla el funcionamiento básico de dos de ellas en los siguientes apartados del curso.
- SAX: Es una de las APIs disponibles en Java para el tratamiento de documentos XML. Durante el curso se ve la estructura general de SAX, así como diversos ejemplos de utilización que comienzan siendo sencillos para ir complicándose a lo largo del temario.
- DOM: Es la segunda API que se muestra a los alumnos puesto que el funcionamiento es bastante diferente a la anterior, dando así dos enfoques totalmente distintos a los asistentes al curso de cómo tratar documentos XML con Java.

A lo largo de todo el temario se le propone a los alumnos la realización de numerosos ejemplos relacionados con la materia que se está introduciendo y son ellos los que los realizan aunque al final del curso se les proporciona la dirección web donde pueden encontrarlos ya resueltos.

Entre estos ejercicios se encuentra realizar un programa Java que se conecte a un servidor remoto donde se encuentran disponibles varios servicios web ya programados y en funcionamiento. El alumno puede aprender como realizar con Java una conexión remota de estas características, cómo acceder a los servicios web disponibles y cómo obtener la respuesta de estos servicios a la petición que ellos formulan.

Para esta propuesta se utiliza una herramienta adicional denominada SoapUI (`http://www.soapui.org/`). Se trata de una herramienta de software libre que nos permite testear el funcionamiento de servicios webs y que los alumnos comienzan a utilizar en esta parte del curso y continúan en las partes siguientes.

[2] `http://www.netbeans.org`

3.3. Desarrollo de Servicios Web con Jax-WS

Sabiendo que la teoría sobre Java, XML y los Servicios Web ya se han impartido en los primeros capítulos del curso, este apartado se centra estructurar conocimientos que permitan relacionar conceptos cuando estemos desarrollando servicios web.

Para realizar un desarrollo de servicios web haciendo uso de Netbeans es necesario explicar nuevos conceptos de la herramienta Netbeans (ya que se usarán nuevas funcionalidades). Además, se explica el uso del servidor de aplicaciones web donde nuestros servicios se desplegarán para su ejecución, que en este caso es el servidor Glassfish [3]. El alumno aprende conceptos básicos del servidor como son: arrancar, parar, cambiar a modo de depuración, desplegar nuevos proyectos y sincronizarlo con el Netbeans.

El temario comienza con la explicación del desarrollo de un servicio web sin utilizar la ayuda de un entorno de desarrollo moderno. En este tema se introduce al alumno en el desarrollo de un WSDL (xml de descripción de un servicio web) escribiendo cada uno de los elementos necesarios para que el sistema pueda comprender y levantar un servicio web. Es decir, a partir de un WSDL se crea automáticamente el código a rellenar con su comportamiento. También se explica el paso contrario: a partir de una clase Java se genera automáticamente el WSDL que la representa.

El siguiente tema presenta la creación de clientes para web services, introduciendo al alumno en las técnicas de programación necesarias para la invocación de servicios web desde cualquier programa Java (una aplicación de escritorio, una aplicación web u otro servicio web). Durante la explicación de este apartado el alumno puede crear clientes para todos los ejemplos realizados en el curso y comparar los resultados por los ofrecidos por un cliente genérico como es SoapUI.

También se profundiza en la creación de Servicios Web y clientes más completos, con estructuras de datos reales y más complejas.

Como conclusión de esta sección se introduce al alumno en el desarrollo de una aplicación final haciendo uso de servicios web desarrollados por ellos mismo que realizan operaciones simples. Como ejemplos de proyecto se propone el desarrollo de interacciones de personajes de una conocida serie de televisión, utilizando objetos complejos, como Usuarios, Listas de Usuarios y otros objetos compuestos.

[3] http://www.glassfish.org

3.4. PHP

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas.

Es un lenguaje interpretado de propósito general ámpliamente usado y diseñado especialmente para desarrollo web que puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Veamos un ejemplo sencillo:

```
<html>
<body>
   <?php
          echo ‘‘Esto es un script PHP’’;
   ?>
</body>
</html>
```

Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP. Éste procesa el script solicitado, que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. Así, al ejecutar el script anterior, el cliente recibirá sólo los resultados de la ejecución por lo que es imposible para el cliente acceder al código que generó la página.

A lo largo del curso se muestran diversos ejemplos, partiendo de algunos muy sencillos para estudiar la sintaxis básica, definición de tipos de datos y uso de estructuras de control. Más adelante se estudian conceptos y ejemplos avanzados, tales como orientación a objetos, tratamiento de formularios web, despliegue y uso de servicios web, y por último, acceso a bases de datos MySQL desde PHP. Con esta sección el alumno comprende que los Servicios Web son independientes del lenguaje, pudien-

do llamar a servicios hechos en Java desde programas PHP.

3.5. Orquestación de Servicios Web con BPEL

BPEL (*Business Process Execution Language*, Lenguaje de Ejecución de Procesos de Negocio) es un lenguaje basado en XML diseñado para el control centralizado de la invocación de diferentes servicios Web, con cierta lógica de negocio añadida que ayudan a la programación en gran escala (programming in the large).

Su objetivo es definir procesos de negocio que interactúen con entidades externas mediante operaciones de un servicio Web definidas usando WSDL y que se manifiestan a si mismas como un servicio Web. Para ello utiliza un lenguaje basado en XML. BPEL sirve para orquestar servicios web, ya que consume servicios existentes para crear nuevos servicios de grano grueso. Un ejemplo de un proceso BPEL puede verse en la Figura 2. Al estar definido en XML existen muchos programas que permiten desarrollar procesos BPEL de forma gráfica, como por ejemplo NetBeans.

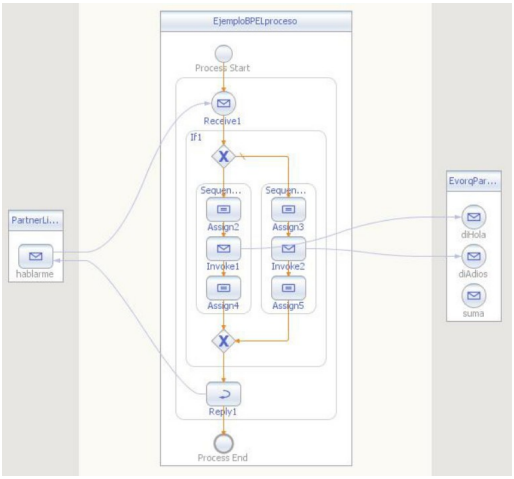

Figura 2: Ejemplo de proceso BPEL realizado en NetBeans. Este proceso llama a las operaciones diHola o diAdios dependiendo de un booleano, y devuelve un saludo

En la parte teórica de este curso se explica también el concepto de ESB (*Enterprise Service Bus*). Un ESB es un contenedor de aplicaciones en el que cualquier aplicación que entienda XML puede ser emisora/receptora de mensajes dentro del bus. Permite una enorme flexibilidad a la hora de diseñar la arquitectura de nuestro sistema, ya que se puede utilizar desde una arquitectura cliente/servidor hasta una arquitectura orientada a eventos. Los componentes son "enchufables", esto da la flexibilidad de añadir funcionalidad a un sistema en producción sin tener que pararlo y sin comprometer el funcionamiento de otros componentes del sistema. Por ejemplo, un componentes básico para ESB es un motor que ejecuta BPEL. Un ejemplo de ESB libre es OpenESB [11], incluido dentro del servidor Glassfish de Sun y utilizado en el curso.

En esta parte del curso se explican los distintos conceptos del lenguaje BPEL y se propone al alumno realizar varios ejercicios utilizando los servicios web creados en secciones anteriores del curso. Para ello el alumno utilizará el programa NetBeans para desarrollar procesos BPEL y desplegarlos dentro de OpenESB. Los primeros ejercicios son iterativos, los alumnos van añadiendo nuevos servicios simples ya desplegados en el servidor del curso a su proyecto BPEL para ir aprendiendo el uso de todas las actividades que se usan en este lenguaje (*for, if, invoke, reply*...). Una vez comprendidos estos conceptos, se les presenta un proyecto más completo en el que tienen que utilizar otros servicios desplegados más complejos para crear distintos procesos BPEL que los orquesten.

3.6. Otras arquitecturas y metodologías SOA

Como cierre al curso se presentan de forma teórica otras implementaciones de SOA y metodologías de desarrollo. Entre las arquitecturas orientadas a servicio más extendidas está OSGi [8], que es una SOA para máquinas virtuales en Java, o ebXML [10], orientada sobre todo al intercambio de datos empresariales. Además, se presentan algunas metodologías para desarrollar sistemas basados en SOA, como BCM [6] o SOMA [1].

Finalmente se presentan brevemente algunos proyectos basados en SOA en los que los docentes del curso han trabajado tanto en el ámbito académico como el empresarial, como por ejemplo eIntegra [5], GAD, AmIVital[4] [3] u OSGiLiath [4].

[4] http://www.amivital.es

4. Respuesta de los alumnos y Conclusiones

El curso ha sido impartido en tres ediciones en la ETS. de Ingenierías en Informática y Telecomunicación de la Universidad de Granada. En todas ellas se ha cubierto el cupo de 27 alumnos. Al final de cada edición se pasó un cuestionario para evaluar el curso, siendo las notas 4,08, 4,5 y 4,8 (sobre 5) en las tres ediciones respectivamente. Además, se propuso a los alumnos qué mejoras habrían de añadirse en ediciones posteriores, siendo aplicadas en tales ediciones, como por ejemplo el aumento de la parte de prácticas respecto a teoría, y orientando la enseñanza a la realización de ejercicios iterativos.

Todo el software utilizado en el curso es libre, con lo que el coste para el alumno es nulo. Toda la información, software y ejemplos del curso se encuentran en la web `http://evorq.ugr.es/cursows`, a disposición de los alumnos y el resto de personas interesadas en la temática.

Agradecimientos

Trabajo financiado por proyecto CEI BioTIC GENIL (CEB09-0010), Programa CEI del MICINN (PYR-2010-13) y beca FPU AP2009-2942.

Referencias

[1] Arsanjani, A. Ghosh, S. Allam, A., Abdollah, T., Ganapathy, S., y Holley, K. SOMA: A method for developing service-oriented solutions. *IBM Systems Journal*, 47(3):377–396, 2008.

[2] Boixader, F., Guardiola, J., Albert, M., y Ribas, J. El Grado en Informática y Servicios. Una respuesta a la nueva demanda del contexto social. *XVI Jornadas de Enseñanza Universitaria de la Informática*, pages 129–135, 2010.

[3] García-Sánchez, P., González, J., Castillo, P. y Prieto, A. Using UN/CEFACT'S Modelling Methodology (UMM) in e-Health Projects. *Bio-Inspired Systems: Computational and Ambient Intelligence*, pages 925–932, 2009.

[4] García-Sánchez, P., González, J., Castillo, P., Merelo, J., Mora, A., Laredo, J. y Arenas, M. A Distributed Service Oriented Framework for Metaheuristics Using a Public Standard, Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), pages=211–222,2010,

[5] García-Sánchez, P., Merelo, J.J., Castillo, P., Sevilla, J.P., Martín, M. y López, M. Plataforma de integración de servicios para la administración basada en BPEL y SOA. En *Actas de las III Jornadas en Servicios Web y SOA (JSWEB 2007)*, pages 111–118, 2007.

[6] OASIS BCM TC. Business-Centric Methodology for Enterprise Agility and Interoperability. Executive White Paper. 2003. `http://businesscentricmethodology.com/`.

[7] Oasis Open. UDDI Specification. 2006. Disponible en: `http://www.uddi.org/specification.html`.

[8] OSGi Alliance. OSGi service platform release 4.2. 2010. Disponible en: `http://www.osgi.org/Release4/Download`.

[9] Papazoglou, M. P., y Van Den Heuvel. W. Service oriented architectures: Approaches, technologies and research issues. *VLDB Journal*, 16(3):389–415, 2007.

[10] Patil, S. y Newcomer. E. ebXML and Web Services. *IEEE Internet Computing*, 7(3):74–82, 2003.

[11] Salter, D. y Jennings, F. Building SOA-Based Composite Applications Using NetBeans IDE 6. *Birmingham-Mumbai: Packt Publishing*, 2008.

[12] World Wide Web Consortium. Web Services Description Language (WSDL) 1.1. 2001. Disponible en: `http://www.w3.org/TR/wsdl`.

[13] World Wide Web Consortium. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), 2007.