WEBBER: USO DE COMPONENTES PARA LA ARMONIZACIÓN DE CONTENIDOS Y METADATOS

Autores: Diego R. López
RedIRIS
diego.lopez@rediris.es

Introducción

El trabajo necesario para mantener un servidor de información en Internet se ha ido haciendo más complejo a medida que la cantidad de información que debía ser ofrecida y el número de fuentes para la misma han ido creciendo. Una de las aproximaciones comunes para afrontar esta situación ha sido el uso de métodos de *deconstrucción*, cuya característica común es mantener separados (total o parcialmente) los datos que deben ser presentados de las especificaciones acerca de su representación. A la hora de generar la versión final del contenido se emplean procedimientos para transformar los datos de acuerdo con las especificaciones. Aunque la mayor parte de estas soluciones han sido desarrolladas de manera completamente específica para un entorno, o incluso un servidor, determinados, este tipo de aproximación deconstructiva es la idea que subyace en soluciones que se están proponiendo recientemente, como es el uso de hojas de estilo en documentos HTML, o el de XML en conjunción con XSLT.

Por otra parte, el empleo de tecnologías basadas en componentes es una alternativa que cada vez gana más aceptación para construir de sistemas distribuidos y/o complejos. Un buen ejemplo de ello son modelos de desarrollo como DCOM o JavaBeans. Cuando se trata de trabajar con los contenidos de un servidor de información, el uso de programas simples y pequeños, cada uno de los cuales con una *visión* distinta e independiente de la información [1], que colaboran para producir un resultado final común se hace más atractivo a medida que crece tanto la heterogeneidad de las fuentes de información como la complejidad en la estructura de la misma.

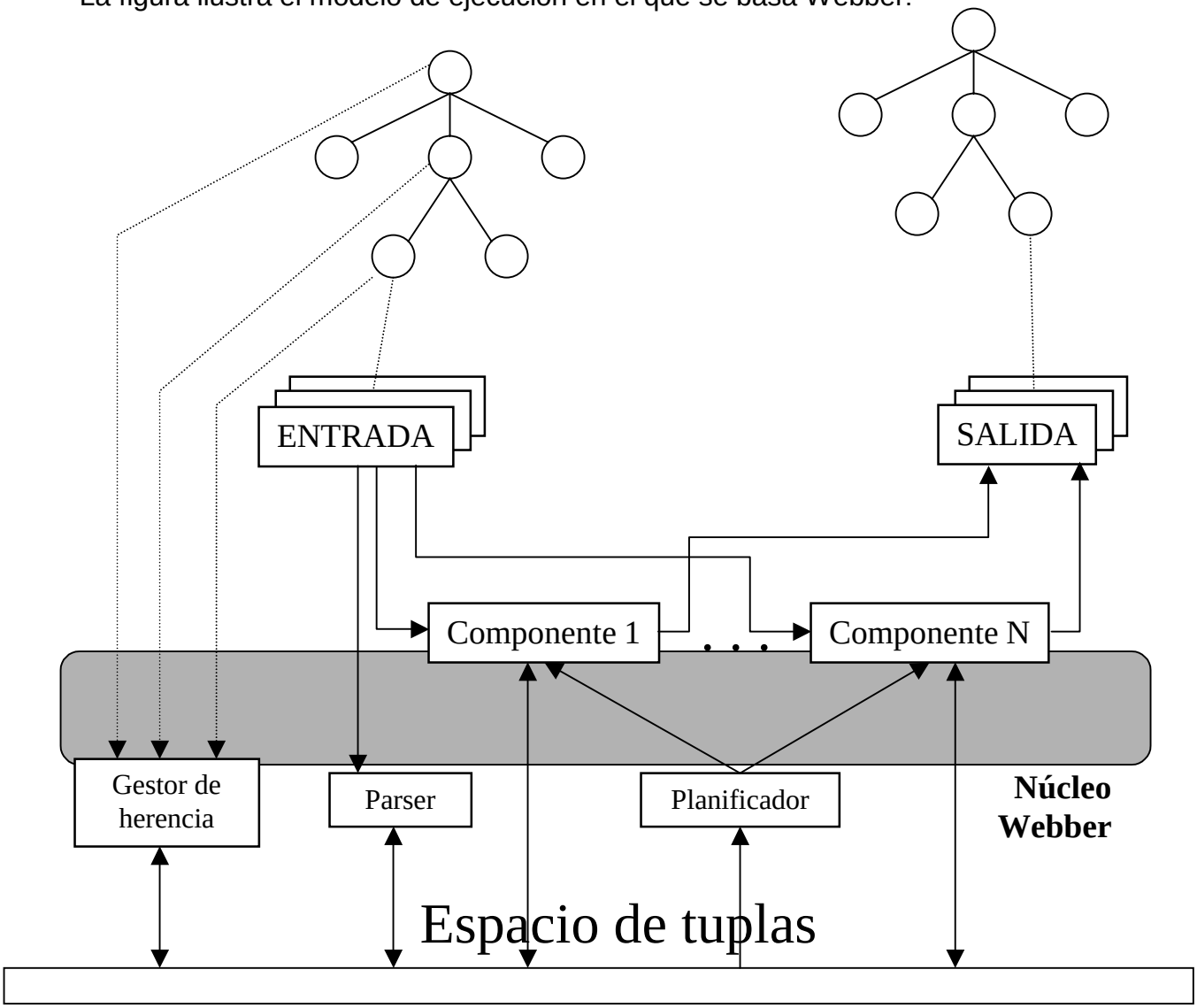
Aquí presentamos un entorno (al que llamamos Webber) que utiliza ambos métodos. Está basado en componentes y es capaz de emplear diferentes fuentes de información para armonizar y dotar de una estructura común a los contenidos que ofrece un servidor de información. En paralelo, el modelo facilita el uso de componentes para gestionar la metainformación acerca de estos contenidos.

Webber fue concebido originalmente como un pre-procesador de HTML para dar un aspecto homogéneo a las páginas Web del servidor de RedIRIS. A partir de ahí, ha evolucionado hasta un entorno completo capaz de emplear diferentes entidades de proceso (los componentes), haciéndolos cooperar de acuerdo con las especificaciones del usuario, para transformar un conjunto de elementos fuente en la versión final de los contenidos que ofrece el servidor, junto con la metainformación relativa a los mismos.

De esta manera, las personas a cargo de escribir los contenidos pueden concentrarse en la información en sí, sin preocuparse acerca de la localización o aspecto final de la misma. Por su lado, los administradores del servidor disponen de una herramienta potente y simple de utilizar para asegurarse de que se ofrecen unos contenidos coherentes. El uso de componentes simplifica los mecanismos de procesado de los contenidos, dado que la cooperación de elementos pequeños que se comunican entre sí facilita la introducción de funcionalidades nuevas, el desarrollo de prestaciones específicas, o la reutilización de elementos ya existentes. En particular, veremos como es posible insertar componentes *inteligentes* para hacer más potente el procesado de la información y ofrecer una mejor interacción con el usuario. Es más, el uso de componentes hace que el entorno sea altamente *extensible*, lo que permite la introducción de nuevas tecnologías para la producción de contenidos o la descripción de metadatos a medida que estén disponibles.

Modelo de ejecución

La figura ilustra el modelo de ejecución en el que se basa Webber:



El entorno se basa en un espacio de datos compartidos (un *espacio de tuplas*) que permite la comunicación entre los componentes. El espacio de tuplas se organiza en variables, cada una de las cuales puede ser accedida de manera asociativa [2], y leída o escrita por los componentes en ejecución, de manera que éstos puedan dejar información para que sea usada por otros componentes. La ejecución del propio núcleo de Webber se controla por medio de alguna de estas variables. Como es lógico, las fuentes de información con las que trabajan los componentes se consideran asimismo variables en el espacio de tuplas.

Los valores iniciales de las variables del espacio de tuplas son asignados dentro de los elementos fuente o bien dentro de plantillas. A la hora de hablar de estos elementos fuente, conviene tener en cuenta que no son necesariamente las fuentes que se utilizan para construir los contenidos del servidor: un elemento fuente Webber puede contener en sus variables referencias a otras fuentes de datos, que son los que emplearán los componentes para producir la versión final de la información ofrecida por el servidor. Las plantillas ofrecen un mecanismo para establecer valores comunes para un conjunto de elementos fuente determinado. Puede definirse uno de estas plantillas para cada una de las localizaciones (directorios) que contenga el servidor.

Una de las características más importantes de Webber es que ofrece mecanismos de *herencia*, tanto a lo largo de los URL como por medio de declaraciones explícitas. En el primer caso, los valores de las variables que han sido fijados en una plantilla en un nivel superior, se mantienen en los niveles hijos si no son reescritos. En el segundo, es posible utilizar los valores definidos en algún otro elemento fuente mediante una referencia al mismo [3]. De esta manera, se pueden aplicar valores por defecto a todos los contenidos del servidor (o un subconjunto determinado), así como a *clases* específicas de contenidos.

Estos mecanismos de herencia no solo tienen su aplicación en las cuestiones de representación de la información, como puede ser el caso de enlaces para facilitar la navegación dentro del servidor o el uso de formularios comunes. Es importante señalar también que buena parte de la metainformación puede ser derivada jerárquicamente a través del URL (datos como el autor o el área de conocimiento suelen ser comunes a una determinada rama en el servidor) o corresponder con una determinada clase de elementos.

En definitiva, una vez se dispone de un conjunto de componentes adecuados, Webber ofrece un interface de programación completamente *declarativo*, de manera que los gestores de la información pueden concentrarse en los modelos de datos y metadatos del servidor.

Componentes: Núcleo y procesadores

El núcleo de Webber está constituido por tres componentes *privilegiados* cuyas tareas comprenden el cargar los valores iniciales de las variables en el espacio de tuplas, ofrecer acceso a las variables, mantener la herencia a través de los diferentes URLs y clases, e instanciar y ejecutar los diferentes componentes encargados de procesar la información. Los únicos requisitos que Webber impone sobre estos componentes de proceso (*procesadores*) son relativos a su interface. Este interface debe estar escrito en Perl (como lo está el núcleo de Webber) y los procesadores deben acceder a las variables del espacio de tuplas a través de las funciones que ofrece el núcleo de Webber, de manera que se preserven las propiedades que Webber mantiene para ellas. Por tanto, un procesador Webber puede ser desde un procedimiento muy simple (básicamente un conjunto de llamadas a la función **print** de Perl que escriben el valor de una variable), a un interface específico que conecte con otros programas tan complejos como sea necesario.

El núcleo

Los tres componentes que conforman el núcleo de Webber son los encargados de dirigir la ejecución del programa y de guiar al resto de componentes para que colaboren en la producción de los contenidos. De acuerdo con el orden en el que toman control de la ejecución de Webber, estos componentes son:

- El gestor de herencia. Este componente se encarga de instanciar las variables dentro del espacio de tuplas definidos por la localización del elemento (herencia a través del URL) y por la(s) clase(s) a la que pertenece. Este es el primer componente que se ejecuta para un elemento fuente determinado, de manera que proporciona la base para el resto de componentes, ya sean del núcleo o procesadores.
- El parser lee las plantillas y elementos fuente para extraer los valores de las variables y asignarlos en el espacio de tuplas. Es invocado por el gestor de herencia cada vez que se atraviesa un nuevo elemento en la jerarquía de URLs, cuando es necesario acceder a una clase determinada, y cuando se accede a un nuevo elemento fuente. Si, por algún motivo, los elementos que se están empleando como fuentes Webber no se corresponden con la sintaxis Webber (por ejemplo, se están empleando ficheros escritos en HTML para su transformación) es posible evitar el uso del parser, permitiendo así que otro componente se encargue de analizar el elemento fuente e incorporar los valores adecuados al espacio de tuplas.
- El planificador se encarga de identificar, cargar y ejecutar los componentes que deben ser aplicados a un determinado elemento fuente. Este componente se encarga de pasar el control a los procesadores y de ofrecerles el interface de

acceso a las variables incluidas en el espacio de tuplas, garantizando el cumplimiento de las restricciones que se hayan impuesto sobre las mismas.

Como ya hemos dicho, el comportamiento de estos componentes del núcleo, como el de todos los componentes ejecutados por Webber, es controlado por medio de variables en el espacio de tuplas. Las variables que definen el comportamiento del núcleo constituyen, como es lógico, un conjunto de variables *reservadas* y sus identificadores comparten el prefijo común **wbb**. Las más significativas de estas variables son las siguientes:

- **wbbTemplateName** es el nombre de la plantilla aplicable a cada localización.
- **wbbSourceRegExp** es una expresión regular que define qué elementos van a ser considerados como fuentes por Webber para una determinada localización. Esto permite el que los árboles fuente y destino sean el mismo y evita el procesado de elementos que no sean fuentes Webber, como pueden ser imágenes o datos en formato binario.
- **wbbTarget** y **wbbTargetName** definen los nombres relativo (a la localización actual) y absoluto del elemento destino que se está generando.
- **wbbLang** se utiliza para almacenar un identificador de dos letras del lenguaje del elemento destino. Aparte de su uso para asignación de metainformación, puede ser empleado también con propósitos de organizar la información, como puede ser el incluir enlaces a versiones del documento en otros lenguajes, o incluir diferentes versiones de elementos de navegación.
- **wbbDate** contiene la fecha de creación del elemento destino.
- **wbbProcLib** define las localizaciones (directorios) donde pueden encontrarse los procesadores.
- **wbbClassLib** define las localizaciones (directorios) donde pueden encontrarse definiciones de clases para los elementos fuente.
- **wbbProc** define los procesadores que deben aplicarse para generar el elemento destino. Las características de estos procesadores son el objetivo del siguiente apartado.

Procesadores

Ya ha quedado dicho que los componentes de proceso (los procesadores) constituyen la esencia de las capacidades de tratamiento de la información del entorno Webber. Los únicos requisitos que un procesador debe cumplir son:

- El planificador debe poder invocarlo a través de métodos Perl accesibles a través de un módulo Perl. Es importante notar aquí que esto no implica restricción alguna sobre el lenguaje que se ha usado para codificar el procesador, que puede ser cualquiera. Únicamente el interface de invocación

por parte del planificador debe estar escrito en Perl.

- Las variables en el espacio de tuplas deben ser accedidas usando uno de los dos métodos que ofrece Webber: un árbol DOM o un array asociativo Perl enlazado (de manera transparente) a ese árbol. En general, la mayor parte de los procesadores utilizarán el array asociativo, dado que el interface que ofrece es mucho más sencillo que el del árbol DOM. Sin embargo, dado que Webber almacena el espacio de tuplas en forma de un árbol DOM, se ha optado por ofrecer también un acceso directo a dicho formato.

El planificador carga los módulos Perl que contienen los procesadores por medio de sentencias **require** de Perl, por lo que las referencias a los procesadores dentro de un elemento fuente deben hacerse en la forma:

NombreDeModulo::NombreDeMetodo

De la misma manera, la extensión estándar para los módulos Perl, ".pm", debe ser empleada para nombrar los ficheros que contienen interfaces de procesadores Webber [4].

Los procesadores son invocados secuencialmente por el planificador, de acuerdo con el orden en el que aparecen incluidos en la variable del núcleo **wbbProc**. Cualquier cosa que un procesador escriba en su salida estándar es automáticamente redirigida hacia el elemento fuente, por lo que los procesadores más sencillos pueden consistir en una serie de sentencias **print**, que tienen en cuenta el valor de algunas variables en el espacio de tuplas.

Procesadores estándar

Uno de los requisitos básicos para que un entorno de este tipo, basado en componentes, sea aplicable es proporcionar junto con sus elementos básicos un conjunto suficiente de componentes [5]. De esta manera, se cubren dos objetivos fundamentales. Por un lado, se facilita el uso del entorno "tal como es", sin requerir que sus potenciales nuevos usuarios comiencen a escribir componentes personalizados desde el principio. Por otro, se proporcionan ejemplos prácticos de cómo deben codificarse los componentes.

La distribución actual de Webber incluye un conjunto de procesadores que permiten realizar tareas como:

- La inclusión en el espacio de tuplas de valores extraídos a partir de fuentes HTML y XML.
- La inclusión en el espacio de tuplas del resultado de la ejecución de programas externos.
- El formato de la información de acuerdo a diferentes estilos bien conocidos, como presentaciones basadas en diapositivas, FAQs, índices para

documentos, etc.

- La interpolación de valores de determinadas variables en otras variables.
- La generación de una firma digital (por medio de PGP) de los contenidos (totales o parciales) del espacio de tuplas.
- La depuración del comportamiento de otros procesadores.

Aparte de estos procesadores de carácter general, la distribución de Webber incluye también otros componentes especialmente diseñados para servidores concretos, de manera que cualquier administrador de un servidor pueda tomarlos como ejemplo para diseñar sus procesadores de propósito específico.

Procesadores inteligentes

El modelo de componentes que ofrece Webber permite conectar de manera muy simple elementos con un comportamiento *inteligente*, simplificando los problemas de interface que habitualmente constituyen impedimentos para la utilización de este tipo de componentes. Los procesadores inteligentes pueden concentrarse en sus tareas, como la simplificación del interface con el usuario o la capacidad de adaptar su comportamiento, mientras que los aspectos que implican un procesamiento clásico se concentran en otros componentes, encargados de proporcionar los datos en un formato adecuado o de explotar los resultados. Además, la programación declarativa de Webber lo hace especialmente indicado para el uso de términos más próximos al lenguaje natural.

En la actualidad, se dispone de dos de estos procesadores inteligentes. El primero es capaz de detectar el lenguaje que se ha empleado al redactar un determinado texto, utilizando para ello algoritmos de aprendizaje a partir de *textos canónicos*. El modelo de componentes que ofrece Webber permite conectar de manera muy simple elementos con un comportamiento *inteligente*, simplificando los problemas de interface que habitualmente constituyen impedimentos para la utilización de este tipo de componentes. Los procesadores inteligentes pueden concentrarse en sus tareas, como la simplificación del interface con el usuario o la capacidad de adaptar su comportamiento, mientras que los aspectos que implican un procesamiento clásico se concentran en otros componentes, encargados de proporcionar los datos en un formato adecuado o de explotar los resultados. Además, la programación declarativa de Webber lo hace especialmente indicado para el uso de términos más próximos al lenguaje natural.

En la actualidad, se dispone de dos de estos procesadores inteligentes. El primero es capaz de detectar el lenguaje que se ha empleado al redactar un determinado texto, utilizando para ello algoritmos de aprendizaje a partir de *textos canónicos* en los lenguajes sobre los que se desea que el procesador pueda trabajar. El procesador requiere una versión adecuadamente tratada del texto de

entrada y devuelve el código correspondiente de lenguaje en la variable del núcleo **wbbLang**.

El segundo emplea un motor de inferencia basado en lógica difusa para extraer las palabras claves de un texto. Este procesador es capaz de trabajar con textos que empleen cualquier lenguaje basado en marcas (típicamente, HTML o XML) y permite al usuario seleccionar hasta 10 marcas diferentes, a cada una de las cuales puede asignar un índice de precedencia expresable por los términos lingüísticos "MIN", "LOW, "MED", "HIG" y "MAX". El procesador emplea una base de reglas jerárquica [6] que utiliza como entradas las precedencias asignadas por el usuario y la frecuencia relativa de las palabras dentro de las marcas. El resultado de cada inferencia permite asignar un índice de relevancia a cada una de las palabras dentro del texto, de las que se seleccionarán como palabras clave aquéllas que superen un umbral de relevancia fijado por el usuario, usando los mismos términos lingüísticos descritos más arriba.

Como es obvio, este procesador es enormemente sensible a la disponibilidad de diccionarios: para la eliminación de palabras vacías, reducción a formas comunes, y empleo de tesauros. La tarea de la gestión de diccionarios puede encargarse, en el entorno Webber, a procesadores especializados, aumentando así la potencia del procesador sin necesidad de modificar su comportamiento.

Conclusiones

Hemos pasado revista aquí a las principales características de un entorno orientado a la armonización del contenido y metadatos de servidores de información. Este entorno está basado en el empleo de pequeñas unidades de proceso (los componentes) capaces de colaborar para obtener el resultado final. El uso de componentes tiene las siguientes ventajas:

- Se simplifica el desarrollo de nuevo software.
- Se fomenta la reutilización del software ya disponible.
- Se dispone de un interface declarativo para controlar el proceso de producción del contenido y los metadatos.
- Se facilita la integración de elementos inteligentes.
- Se garantiza la extensibilidad del entorno a medida que evolucionan las tecnologías.

Este entorno ha demostrado una extraordinaria utilidad en su aplicación a entornos complejos y en producción. El grupo que desarrolla Webber continúa su trabajo, mejorando sus prestaciones y su modelo formal, y creando nuevos componentes susceptibles de ser reutilizados.

Referencias

1. C. SZYPERSKI. *Component Software: Beyond Object-Oriented Programming.* ACM Press, 1998.
2. P. CIANCARINI y otros. *Coordinating Multiagent Applications on the WWW: A Reference Architecture*. IEEE Transactions on Software Engineering, vol. 2, n. 8. Mayo 1998.
3. H.-W. GELLERSEN y M. GAEDKE. *Object-Oriented Web Application Development.* IEEE Internet Computing, vol. 3, n. 1. Enero/Febrero 1999.
4. L. WALL y otros. *Programming Perl*. O'Reilly & Associates, 1996.
5. R. L. LEACH. *Software Reuse: Methods, Models, and Costs*. McGraw-Hill, 1997.
6. D. R. LÓPEZ y otros. *XFL: A Language for the Definition of Fuzzy Systems*. Proceedings of the 6th IEEE International Conference on Fuzzy Systems, pp. 1585-1591. Julio 1997