Manejador de Contenidos

Por Dr. Armando Sarmiento Rojas, Jefe Centro
Lic. Juan Carlos Olamendy Turruellas, Profesional en Adiestramiento
Centro Regional de Capacitación de Villa Clara, ETECSA
sarmiento@tel.etecsa.cu, john@vcl.tel.etecsa.cu

Tanto en Internet como en las intranets o redes corporativas, se dispone de gran cantidad de información almacenada en diferentes medios y formatos, que debe ser manipulada y procesada antes de presentarla en un Sitio Web. Pueden encontrarse bases de datos en todas sus variantes —SQL Server, Oracle, Access, FoxPro—, el sistema de archivo de la PC, el sistema de archivos distribuido de la entidad corporativa —*Distributed File System* (DFS)—[1], los buzones de correo, y otros recursos de la red como FTP, Gopher, WAIS, listas de discusiones, etc.

Por su estructura lógica, además su localización física tan disímil, la información es muy difícil de publicar y de administrar —ya sea añadir, actualizar, eliminar o asociar metadatos a dichos datos— de una forma homogénea. También existen varios clientes que necesitan acceder a la información, y estos usan distintas tecnologías de presentación (*rendering*), notándose entre las más comunes, las aplicaciones clientes de sobremesa que ejecutan bajo el subsistema de ventanas del Sistema Operativo Windows, las aplicaciones Web usando los navegadores —*Microsoft Internet Explorer, Opera, Lynx, Netscape's Navigator*—, o las aplicaciones sobre dispositivos móviles —*Pocket PC, Smart Devices*—[2], etc.

En una aplicación de este tipo los objetos están separados conceptualmente según el papel que desempeñan, la tecnología que sustenta o su distribución, también atendiendo a la configuración del actor, quien en un momento determinado hace uso del Sistema.

Para que el especialista responsabilizado con el manejo y la presentación de los contenidos, que están en el Sitio Web, pueda realizar su trabajo de una forma eficiente, debe disponer de herramientas administrativas que le permitan gestionar todo el Sistema de una forma homogénea, de manera que en una parte se encuentren los flujos de información, definidos como contenidos que provienen de distintas fuentes y necesitan la realización de un procesamiento previo o ser cualificados según reglas empresariales; y del otro lado se tengan clientes que se definen como actores activos y esperan una salida agradable, independiente de los medios empleados para acceder a los datos.

En la figura 1 puede observarse el flujo de trabajo que sigue el Manejador de Contenidos. Desde que el usuario solicita la información, esta se pasa al Sistema —Manejador de Contenido—, que posee un contexto de ejecución determinado por la plataforma en la cual ejecuta —*runtime engine*—. Luego, la petición es analizada y decide qué contenedores de salida asociarle a partir del conocimiento de la tecnología actual del cliente, qué transformación hacerle y qué flujo de información debe ser obtenido. Cuando se conozcan todos los objetos que van a utilizarse para la creación de la respuesta final, se hará un procesamiento y la salida es enviada al originario de la petición.

Para facilitar la compresión del problema se definen nuevos conceptos como las plantillas (*template*) —objetos que contienen elementos



Figura 1 Flujo de trabajo del manejador de contenidos

estáticos prediseñados con anterioridad— y los contenedores (*containers*) —receptores de la información de interés, generadas dinámicamente por los transformadores—. Estos son los primeros elementos que forman parte del flujo de trabajo del Sistema.

La generación de elementos dinámicos es muy amplia. Existen tipos bien definidos y extensibles que especifican cómo generar información dinámicamente —*mappers* / transformadores— y conociendo una fuente, se le aplica la operación transformar y la salida se asocia a un contenedor. La plataforma contiene una colección de transformadores por defectos, que posibilita que se agreguen más transformadores.

Se tienen, además, fuentes de información que son los puntos de donde se extraerán los datos.

Todas estas ideas básicas crean un modelo de programación avanzado, menos abstracto, y orientado a especificar el problema sin importar cómo va a implementarse.

Para la gestión total del proyecto se hace uso del RUP —*Rational Unified Process* / Proceso Unificado Racional—[3] que es un conjunto de procedimientos y métricas para mostrar el estado actual del desarrollo de un proyecto, y permite responder a las preguntas: ¿cuál sería la próxima fase?, ¿qué fases pueden desarrollarse en paralelo?, ¿en qué momento puede ocurrir un hecho significativo o hito como la entrega de una versión de software o el cambio de una fase a otra, por ejemplo, pasar de la etapa del diseño a la etapa de implementación?, ¿cuándo se podría dar conclusión al trabajo?, ¿cuántas personas deben trabajar en el proyecto y cuáles son sus responsabilidades?

Estos procedimientos tienen fundamentos matemáticos, e incluyen la heurística desarrollada a partir de la experiencia del éxito o del fallo de grandes proyectos anteriormente desarrollados. A la especificación de dicho proceso, compañías productoras de software como Microsoft, IBM y HP han hecho grandes contribuciones.

La idea general del Sistema consiste en crear capas donde se agrupan objetos conceptualmente relacionados. Siguiendo las prácticas del diseño de aplicaciones, documentadas como Patrones de Diseño —*Design Pattern*—[4,5], la creación de aplicaciones de varias capas —*n-tier applications*— permite, entre otras cosas, independizar partes de la aplicación, según el papel global que juegue, y distribuirlas entre varios entornos de ejecución, ya sean Servidores de Aplicaciones —*Application Servers*—, servidores, PC, dispositivos móviles, que posibilitan el necesario aislamiento físico y lógico a los componentes computacionales y, en caso de presentar problema uno de ellos, no fallará la aplicación completa, permitirá al usuario el funcionamiento de su aplicación sin a penas darse cuenta. Este tipo de arquitectura se caracteriza por ser escalable, distribuida, más estable y garantiza la independencia y coherencia de los módulos del Sistema. Además, se divide el problema grande en subproblemas fáciles de resolver, y la buena armonía de la comunicación entre estos —*orchestration* / orquestación—[6] permite dar una solución final, según el principio "divide y vencerás" —*divide and conquer*—. De esta forma se desarrolla un modelo donde se establecen capas bien definidas que separan los objetos de acuerdo con su función en la aplicación:

1. La capa de presentación —*Presentation Layer*— contiene los objetos que participan en la presentación del contenido, es decir, los controles Web del lado del cliente y del servidor; transformaciones XSL sobre documentos XML u otros objetos que definan su mecanismo de persistencia en dicho formato; controles clientes sobre formularios de Windows; controles para aplicaciones móviles; u otros tipos de componentes con tecnología capaz de generar una presentación —*rendering technology*— a partir de los datos que se le asocien y transformaciones relacionadas con estos. Además se especifican los componentes controladores del proceso de presentación, según el patrón de diseño Modelo-Vista-Controlador —*Model-View-Controller*—[7].

2. La capa de negocio o lógica —*Business Layer*—[8] contiene los objetos que participan en la lógica de la aplicación, o sea, los componentes que definen el dominio semántico de la aplicación: Entidades Empresariales —*Business Entity*—[9] que representan los objetos con información asociados a un mecanismo y medio de persistencia —bases de datos, documentos XML, flujos, etc.—; los Objetos Empresariales —*Business Objects*—[10] que incluyen la lógica, las reglas y los casos de usos de la aplicación; y los Flujos Empresariales —*Business Workflows*—[11] los cuales contienen las definiciones de los pasos necesarios para la perfecta ejecución de los casos de usos de la aplicación (*orchestration*). Aquí se construye un diagrama de interfases, las clases y sus relaciones, y se modelan los conceptos del Manejador de Contenidos como los flujos

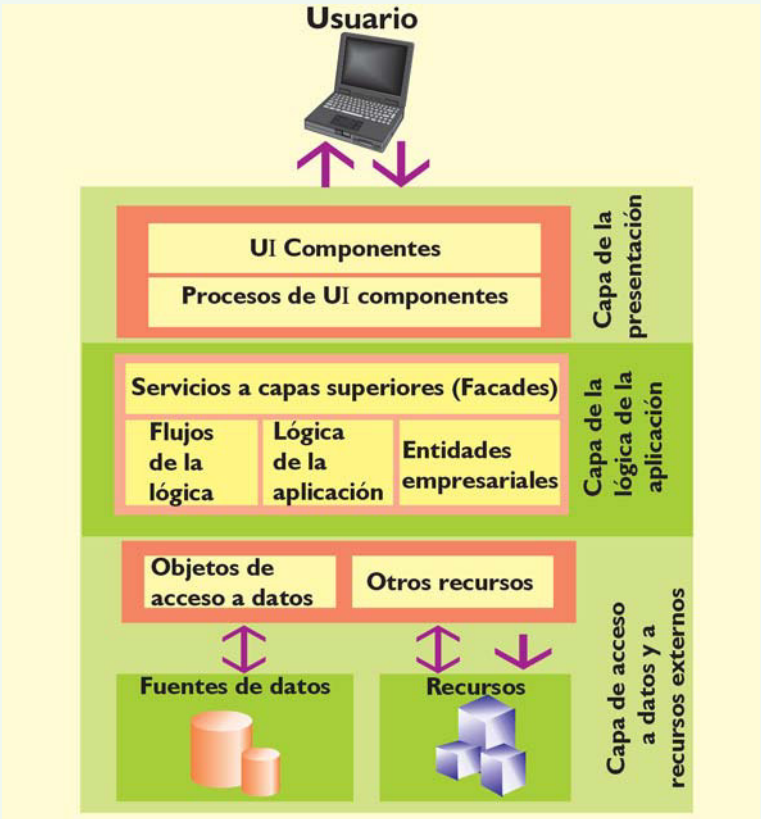
de datos, los metadatos de ellos —datos asociados a datos, o datos de datos—, los posibles casos de usos de la aplicación y sus escenarios.

3.La capa de acceso a recursos externos —*Data Access Layer*— abarca los objetos que permiten acceder a recursos externos para buscar información, pueden ser bases de datos, recursos de la red, recursos del sistema operativo en el cual corre la aplicación, entre otros. Esta es la única capa que depende de la plataforma de despliegue —*deployment platform*—. Contiene un conjunto de adaptadores (*adapters*) que conocen los objetos externos, sus estructuras, el método de acceso a ellos, y se convierten en proveedores de servicios de los objetos que están en las capas superiores, de tal manera que se independiza la lógica de la aplicación de los objetos externos, creándose un buen mecanismo de portabilidad. Pueden definirse otras clases de adaptadores con sólo implementar una interfase, y los objetos instancias de dichas clases se agregan como *add-in* a la plataforma, incluso, en tiempo de corrida —*hot add-in*—.

Puede observarse en la figura 2 el modelo de la aplicación en capas y los objetos asociados a estas, donde cada una consume servicios de las capas inmediatas inferiores, y le provee de servicios a las capas inmediatas superiores, mediante un conjunto de interfases de programación, que se denominan caras de los servicios o *facades*.

En la etapa de diseño se emplea UML —*Unified Modeling Language* / Lenguaje de Modelado Unificado—[12,13], el cual es un lenguaje desarrollado por tres grandes teóricos de la programación —Grady Booch, Ivar Jacobson y James Rumbaugh—, permite modelar cualquier problema y expresarlo en términos de objetos y sus relaciones, con el empleo de un lenguaje sencillo e independiente del tipo de implementación. Además, se hace uso de metodologías emergentes en los últimos tiempos, como la Programación Orientada a Patrones de Diseños —*Pattern Design-Oriented Programming*—, la Programación Orientada a Aspectos —*Aspect-Oriented Programming*— y la Arquitectura Orientada a Servicios —*Service-Oriented Architecture*—.

Figura 2 Modelo de la aplicación en capas



Se describen, también, los escenarios de los casos de uso definidos en la etapa anterior, aparece un modelo de objetos y sus relaciones por cada capa, incluso, verticalmente —relaciones entre las capas—.

En la capa de presentación se usan controles que brinda la plataforma donde se desarrolla la solución. Además, se definen nuevos controles para extender la interfaz gráfica, las transformaciones que deben aplicarse a los flujos y los controladores del proceso de la interfaz gráfica.

En la capa de negocio o lógica se definen conceptos de la semántica de la aplicación, por ejemplo:

1.Una extensión del Sistema Archivo como raíz común de todos los flujos que se van a administrar permitiendo que se le asignen metadatos. Aparecen algunas *Business Entity*, como las categorías de información, que contienen a su vez otras categorías o información.

2.Los aspectos, o sea, todos los objetos que cualifiquen el Sistema y que por su naturaleza son globales.

3.Objetos que a partir de la definición de flujos de trabajos, orquestan otros objetos de lógica, de una forma coherente y espacial, para lograr una solución cooperativa final.

Además, se brindan interfases de programación, que permiten extender la plataforma con nuevas funciones, al crear componentes para la implementación de dichas interfases y, agregando sus instancias a zonas específicas del flujo de trabajo, sin rediseñarse ni parar el sistema.

En la capa de acceso a recursos externos se definen todos los adaptadores (*adapters*)[14] a recursos de los flujos de información. Se han definido flujos a bases de datos, a recursos de red y al Sistema de Archivo, además de interfases de programación, y si se desea añadir nuevas fuentes de flujos, se crea un componente que implemente dichas interfases y se le informa al administrador de flujos, y a quien se comunica con los flujos mediante dichas interfases.

Para la etapa de implementación se emplean tecnologías novedosas que permiten especificar lógicamente lo que se ha expresado en el modelo, sin establecer una ruptura con la semántica de la solución. Se ha escogido a Microsoft.NET como entorno de desarrollo y de ejecución, debido a la cantidad de tecnologías que esta plataforma brinda para la programación y las bondades en tiempo de corrida, como el uso del Recolector de Basura —*Garbage Collector*— que libera la memoria asignada a un objeto que ya no existe, y optimiza dicho recurso muy escaso.

La arquitectura de la solución se ha enriquecido gracias a la plataforma Microsoft.NET y a la biblioteca de clases —*Base Class Library*— que esta incorpora.

La capa de presentación usa los controles Web —*WebControls* y *HtmlControls*— y define nuevos objetos gráficos.

La capa de negocio o lógica importa utilitarios e instancias bases de Microsoft.NET para la implementación de los escenarios.

La capa de acceso a bases de datos y recursos externos utiliza los adaptadores (*Adapters*) elementales predefinidos —Sql*DataAdapter*, Odbc*DataAdapter*, OleDb*DataAdapter*, Oracle*DataAdapter*— para crear nuevos componentes para flujos de datos como el correo electrónico, recursos de red FTP, Web, *Active Directory*, y el sistema de archivo.

Se considera que con el desarrollo del Manejador de Contenidos se ha creado una plataforma sólida para la gestión de la información en el desarrollo de sitios, que tendrá un impacto positivo no solamente en el ámbito de la Gerencia Territorial de Villa Clara, sino también en ETECSA, una vez generalizada su utilización. Es un producto con posibilidades de ser comercializable por su novedad y las tecnologías que incorpora.

Este Manejador de Contenidos, a diferencia de otros en el mercado, es capaz de administrar cualquier tipo de flujo de información y producir una salida independiente del dispositivo usado para acceder a la información, ya sea un navegador en una PC, un dispositivo móvil, un dispositivo inteligente o una terminal de información; y no contiene ningún elemento que invalide su uso extensivo, tanto en la Intranet Corporativa como en otras redes nacionales. Además, es una plataforma extensible capaz de asimilar sitios ya existentes y proveerlos de nuevas funcionalidades.

Notas

[1]Daniel A. Menascé. *Distributed File System*. 1st Edition. USA: Addison-Wesley Professional, 2004.

[2]John Kanalakis. *Developing .NET Enterprise Applications*. USA: Apress. 2004.

[3]"Rational Unified Process". Diponible en: http://www.rational.com/rup/index.jsp (Consultado: diciembre, 2004).

[4]Christopher Alexander. *The Timeless Way of Buildings*. UK: Oxford University Press. 1979.

[5]Gamma, Helm, Johnson, and Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1st Edition. USA: Addison-Wesley Professional, 1995.

[6]Juric Matjaz, Benny Mathew, Poornachandra Sarang. Business Process Execution Language For Web Services. UK: Packt Publishing Limited, 2004.

[7]Programming using Model-View-Controller. Scott R. Weiner, Stephen Asbury, Scott Weiner. USA: John Wiley & Sons. 1998.

[8]Enterprise Applications. Best Practices. Microsoft Press. 2004.

[9]Ibídem.

[10]Ibídem.

[11]Ibídem.

[12]"Rational Unified Process". Disponible en: http://www.uml.org. (Consultado: diciembre, 2004).

[13]Grady Booch; James Rumbaugh; Ivar Jacobson. *The Unified Modeling Language User Guide*. 1st Edition. USA: Addison-Wesley Professional, 1998.

[14]Hasan Jeffrey. *Expert Service-Oriented Architecture in C#*. USA: Addison-Wesley, 2005.

Bibliografía

Alexander, Christopher. *The Timeless Way of Buildings*. UK: Oxford University Press, 1979.

Booch, Grady; Rumbaugh, James ; Ivar Jacobson. *The Unified Modeling Language User Guide*. 1st Edition. USA: Addison-Wesley Professional, 1998.

Enterprise Applications. Best Practices. Microsoft Press, 2004.

Gamma, Erich; Helm, Richard; Johnson, Ralph; and Vlissides, John. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1st Edition. USA: Addison-Wesley Professional, 1995.

Jeffrey, Hasan. *Expert Service-Oriented Architecture in C#*. USA: Addison-Wesley, 2005.

Juric, Matjaz B.; Mathew, Benny; Sarang, Poornachandra. *Business Process Execution Language For Web Services*. UK: Packt Publishing Limited, 2004.

Kanalakis, John . *Developing .NET Enterprise Applications*. USA: Apress, 2004.

Menascé, Daniel A. *Distributed File System*. 1st Edition. USA: Addison-Wesley Professional, 2004.

"Rational Unified Process". Disponible en: http://www.rational.com/rup/index.jsp (Consultado: diciembre, 2004).

"Rational Unified Process". Disponible en: http://www.uml.org. (Consultado: diciembre, 2004).

Weiner, Scott R., Asbury, Stephen, Weiner, Scott. *Programming using Model-View-Controller*. USA: John Wiley & Sons., 1998.