ACSCV.net - Aplicación práctica del mercado de los contenidos.

Fidel Santiago
Network-Sec.com

www.network-sec.com
jfsantial@network-sec.com

Abstract

Esta comunicación trata sobre la integración de contenidos proporcionados por un agente externo dentro de un portal web. La tecnología usada para el intercambio es XML. En este documento se evalúan distintas tecnologías (XSLT, Perl, PHP, Java) con el ánimo de encontrar la más adecuada al proyecto en su conjunto. Al final se presentan las conclusiones del estudio.

This communication is about the integration of contents provided by an extern agent in a web portal. XML is the technology used for the information. In this document we evaluate technologies as XSLT, Perl, PHP and Java looking for the best technology for the project as a whole. Finally, we present the conclusions of the work.

Palabras clave: XML, XSLT, Perl, PHP, Cocoon, programación orientada a objetos.

Índice:

Introducción :: ACS

LA ASOCIACIÓN DE CORREDORES DE SEGUROS DE LA COMUNIDAD VALENCIANA (A.C.S.) es la primera agrupación de Corredores de Seguros creada en la Comunidad Valenciana integrada en la Federación Española de Corredores de Seguros (FECOR) y en la Confederación Empresarial Valenciana (CEV). A.C.S. es una ASOCIACIÓN DE EMPRESARIOS constituida como entidad asociativa desde 1994 con el fin de actuar en defensa de los intereses generales de carácter profesional y llevar a cabo la gestión de intereses comunes a los Asociados.

Establecer principios y normas de actuación profesional, independientes y cualificados de la actividad tales como los relativos a la imagen pública; la edición de folletos de prestigio; el asesoramiento jurídico; la formación; la información; la creación de un banco de publicaciones y datos; el establecimiento de acuerdos de cooperación con especialistas de prestigio, entidades bancarias y de crédito, así como con proveedores de materiales y servicios; defender frente al mercado la imagen fiel y actual de los servicios profesionales de los Corredores de Seguros actuando como referencia y punto de encuentro entre los representantes sociales y empresariales más importantes del sector de la intermediación aseguradora y ofrecer así a sus asociados servicios de mayor valor añadido.

En la sociedad actual, para una profesión como la de Corredor de Seguros, nada es más importante que la credibilidad. Por ello los profesionales se asocian, crean mecanismos de comunicación propios, instrumento de defensa y servicios comunes para incrementar el grado general de credibilidad.

Por ello, conociendo la evolución tan competitiva del mercado, insistimos en la colaboración de todo el sector, y especialmente de los Corredores, para compartir juntos estos objetivos de un proyecto necesario y común como es la defensa de un sector profesional como el nuestro.

Introducción :: El proyecto

Este proyecto pretende construir un Intranet para dar un mejor servicio a los asociados de ACS. Es necesario para una asociación que cuenta con una gran dispersión de los asociados, tener un sistema de gestión de la información eficiente.

A su vez, este proyecto pretende fomentar la difusión de las nuevas tecnologías entre los asociados. Para ello se crea un portal con diversos servicios telemáticos entre los que se destaca el apartado de intercambio de contenidos on-line.

Este proyecto contempla formación on-line, información económica diaria, bases de datos de subvenciones, bolsa de empleo, posibilidad de intercambio de bienes, etc. Una serie de servicios destinados al mejor uso por parte del asociado de la información y la infraestructura que la asociación es capaz de poner en su poder.

Introducción :: El integrador :: Network-Sec

Network-Sec es una consultora en nuevas tecnologías que viene operando en Valencia desde mediados de 1999.

Entre sus áreas de desarrollo se encuentra el desarrollo de portales web, la integración de aplicaciones y el área de sistemas para soporte web. Para ello cuenta con un grupo de profesionales multidisciplinar que engloba tanto a diseñadores con formación artística como a técnicos informáticos, de telecomunicaciones, expertos en usabilidad, etc.

Introducción :: El proveedor de contenidos :: Ifedes NetFactory

Grupo Ifedes nace en 1.991 con un claro objetivo: “constituirse en una sólida estructura de servicios orientada al desarrollo organizacional como elemento de mejora de la competitividad empresarial”.

Grupo Ifedes congrega en su organización a un equipo multidisciplinar de profesionales para la prestación de servicios.

Ifedes Netfactory

IFEDES Netfactory se consolida como la división para Internet de Grupo IFEDES, aglutinando la experiencia, los conocimientos y el equipo humano de esta compañía.

Ofrece un servicio global para el desarrollo de actividades en Internet, facilitando la apertura de nuevos canales de comercialización y desarrollando nuevos modelos de negocio.

Introducción :: La tecnología base :: XML

En los últimos años se ha estado hablando del Lenguaje Extensible de "Etiquetado", eXtensible Markup Language, o XML, y en todos los foros de Internet que se precien de estar a la última no se puede tratar ningún tema sin hacer mención a este nuevo estándar.

La versión 1.0 del lenguaje XML es una recomendación del W3C desde Febrero de 1998. Está basado en el anterior estándar SGML (Standard Generalized Markup Language, ISO 8879), que data de 1986, pero que empezó a gestarse desde principios de los años 70, y a su vez basado en el GML creado por IBM en 1969. Esto significa que aunque XML pueda parecer moderno, sus conceptos están más que asentados y aceptados de forma amplia. Está además asociado a la recomendación del W3C DOM (Document Object Model), aprobado también en 1998.

SGML proporciona un modo consistente y preciso de aplicar etiquetas para describir las partes que componen un documento. XML se derivó de SGML como un subconjunto simplificado, eliminando las partes más engorrosas y menos útiles.

XML constituye la capa más baja dentro del nivel de aplicación, sobre el que se puede montar cualquier estructura de tratamiento de documentos, hasta llegar a la presentación. Así podemos ver la compartición de documentos entre dos aplicaciones como intercambio de datos a ese nivel.

Este es el verdadero potencial de XML: la posibilidad de compartir documentos cargados con información y permitir que sea el receptor el que decida que hacer, como representar, esa información.

XML se ha convertido así en la herramienta fundamental para intercambio de contenidos en la Internet moderna. Gracias a la separación total entre representación y contenidos podemos vender estos últimos sin que esto conlleve la responsabilidad de adaptar la representación de los mismos al cliente. El cliente ahora es libre de procesar esos datos para obtener otros datos a partir de ellos o para presentarlos.

XSL es un lenguaje para construir hojas de estilo. Consta de tres partes: XSLT (XSL Transformations): un lenguaje para transformar documentos XML; el lenguaje XML Path (XPath), un lenguaje usado por XSLT para acceder o referirse a distintas partes de un documento XML. La tercera parte es XSL Formatting objects: un vocabulario XML para especificar semánticas de formato. Una hoja XSL especifica la representación de una clase de documentos XML describiendo como una instancia de la clase se transforma en otro documento XML.

Evaluación :: Características a evaluar

Como características fundamentales a evaluar dentro de las diversas tecnologías a probar tenemos la velocidad de proceso de las solicitudes que se presentan al servidor y la facilidad de uso y/o integración dentro del portal desarrollado.

A través de los ejemplo de código que se proporcionan a lo largo de esta comunicación se podrá apreciar la facilidad o dificultad para integrar el documento XML dentro del portal gracias a las distintas tecnologías.

Para evaluar la velocidad de proceso de las distintas soluciones propuestas se ha desarrollado un sencillo programa en Perl para medir el tiempo que tarda el servidor en dar las respuestas. En este programa se puede configurar el número de repeticiones de la prueba (*$num_cycles*), el número de peticiones consecutivas (*$burst*) y el tiempo de espera entre una ráfaga y otra (*$time_between_bursts*).

```
#!/usr/bin/perl -w

use LWP::UserAgent;
use Time::HiRes qw ( time );

$ua=LWP::UserAgent->new;
$ua->agent("CALSIPROBE/0.1");

my $req = HTTP::Request->new(GET => 'URL DE PRUEBA');
my $num_cycles=1000;
my $time_between_bursts=10;
my $burst=5;
open(DATA_OUT,">data");

for ($i=0; $i<$num_cycles; $i++) {
for ($j=0; $j<$burst; $j++) {
        my $time_0=time();
        my $res = $ua->request($req);
        my $time_1=time();
        if ($res->is_success) {
                my $time=$time_1-$time_0;
                print DATA_OUT "$time\n"
        }
}
        sleep($time_between_bursts);
}
close DATA_OUT;
```

Evaluación :: Opciones tecnológicas :: PHP y XSLT

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje "Open Source" interpretado de alto nivel, especialmente pensado para desarrollos web y el cual puede ser embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas web páginas dinámicas de una manera rápida y fácil.

PHP implementa XSLT como un bloque de funciones basadas en los paquetes *Sablotron* y *expat*.

Veamos el código fuente utilizado para la prueba:

```
<?php

// Allocate a new XSLT processor
$xh = xslt_create();

// Process the document, returning the result into the $result variable
$result = xslt_process($xh, 'prueba.xml', 'prueba.xsl');
if ($result) {
   print $result;
}
else {
   print "Sorry, the xml could not be transformed by the xsl into";
   print "  the \$result variable the reason is that " . xslt_error($xh) .
   print " and the error code is " . xslt_errno($xh);
}

xslt_free($xh);
?>
```

Como se puede ver en el mismo, el uso de XSLT en PHP es muy sencillo. En el resto de soluciones tecnológicas esto es una constante.

Primero se crea un procesador de XSL (*$xh = xslt_create()*). Luego se invoca la función *xslt_process* a la cual se le pasan los ficheros a procesar (*prueba.xml* y *prueba.xsl*) y el procesador creado en el paso anterior. Esta función devuelve el documento final.

La función *xslt_process* admite como entrada un par de nombres de fichero o un par de variables donde se ha almacenado el documento XML o el XSL para realizar la transformación. Esto es especialmente útil en casos como el que nos ocupa puesto que así podemos “cargar” el documento XML proporcionado por el proveedor en una variable de PHP para procesarlo.

Evaluación :: Opciones tecnológicas :: Perl y XSLT

Perl es un lenguaje optimizado para el análisis de ficheros de texto, extraer información de esos ficheros e imprimir informes basados en esa información. El lenguaje pretende ser práctico (fácil de usar, eficiente y completo) antes que "hermoso".

El módulo XML::XSLT implementa la especificación XSLT del W3C.

XML::XSLT hace uso de otros módulos Perl como son XML::DOM, LWP::Simple y XML::Parser.

Los documentos XML y XSL pueden ser pasados a las funciones de procesamiento como nombres de fichero, manejadores de fichero, cadenas o *DOM-trees*.

Analicemos el código fuente:

```
#!/usr/bin/perl -w

use XML::XSLT;

## global vars ##
my $xslfile = "";
my $xmlfile = "";

$xslfile="prueba.xsl";
$xmlfile="prueba.xml";

my $XSLTparser = XML::XSLT->new ($xslfile);
print $XSLTparser->serve($xmlfile);
;
```

Al igual que en el apartado anterior la estructura es muy sencilla: se crea un procesador (aquí *parser*), esta vez asociado a una XSL, y luego se usa ese *parser* para procesar un documento XML. El método *serve* del objeto XML::XSLT proporciona, a su salida, una cadena con el documento procesado.

Evaluación :: Opciones tecnológicas :: JavaServlets, XSLT y Cocoon

Dentro del proyecto Apache, Cocoon es un marco de publicación que lleva el uso de XML y XSLT a un nuevo nivel. Diseñado para obtener la máxima escalabilidad y rendimiento alrededor del procesado SAX, Cocoon ofrece un entorno flexible basado en la separación entre contenido, lógica y estilo.

Cocoon se relaciona con la mayoría de las fuentes de datos, incluyendo sistemas de ficheros, bases de datos, LDAP, bases de datos XML nativas y fuentes de datos en red. Es capaz de adaptar el envío de la información basandose en el dispositivo capaz de recibirla. Puede generar HTML, WML, PDF, SVG, etc. Cocoon está especialmente preparado para ejecutarse como un servlet.

Su uso es extremadamente sencillo, más aún que las soluciones vistas. Eso sí, un paquete tan potente es capaz de realizar tareas mucho más complejas que las aquí presentadas.

Una vez configurado correctamente el software: el servidor de *servlets* y el propio Cocoon, el mismo servidor intercepta las llamadas a cualquier fichero .xml y lo procesa acorde a su definición de cabecera XSL.

Así un documento como el siguiente (*hello.xml*):

```
<?xml version="1.0"?>

<?xml-stylesheet href="hello.xsl" type="text/xsl"?>

<page>
 <title>Hello World!</title>
 <content>
  <paragraph>This is my first Cocoon page!</paragraph>
 </content>
</page>
```

Sería automáticamente procesado por el sistema aplicando el XSL “*hello.xsl*” al propio documento.

Evaluación :: Opciones tecnológicas :: PHP y objetos

Otra posibilidad contemplada, radicalmente diferente a las anteriores, es procesar el documento XML proporcionado por el proveedor de servicios y cargar el resultado en un objeto a través del cual podemos acceder a la información donde y cuando sea necesario.

Para conseguir esto hacemos uso de un *parser*: el expat de James Clark, para procesar los ficheros XML y cargar luego la estructura “parseada” dentro de un objeto.

Veamos un ejemplo:

```
<?
class xml_class {
	var $url;

  function xml_class ($url)
  {
	$this->url = $url;
  }

  function query ($query)
  {
	$file = $this->url.$query;
	$fp = fopen($file, "r");
	while ($data = fread($fp, 4096)) {
		$contents=$contents.$data;
	}
	fclose ($fp);

	$xml_parser = xml_parser_create();
	xml_parse_into_struct($xml_parser,$contents,$vals,$index);
	xml_parser_free($xml_parser);

	foreach ($vals as $element) {
		if ($element["type"]=="complete") {
			$aux["tag"]=$element["tag"];
			$aux["value"]=$element["value"];
			$result[]=$aux;
		}
	}
	return $result;
  }
}
?>
```

Creamos una clase *xml_class* con una funcionalidad mínima: descarga de la red una URL, con parámetros si es necesario, y los procesa a través del parser expat (funciones *xml_parser_*). La función *xml_parser_into_struct* devuelve una serie de vectores con la información contenida dentro del documento XML. Se depura esta estructura y se crea un vector bidimensional con los valores “útiles” de los vectores anteriores y sus etiquetas correspondientes.

```
<?
class prueba extends xml_class {
	var $elementos;

	function prueba ($url) {
		$this->xml_class ($url);
	}

	function load ($query) {
		$elementos=NULL;
		$result=$this->query($query);
		$j=-1;
		foreach ($result as $element) {
			switch($element["tag"]):
				case "TAG1":
					$j++;
					$this->elementos[$j]= new elemento();
					$this->elementos[$j]->TAG1=$element["value"];
					break;
				case "TAG2":
					$this->elementos[$j]->TAG2=$element["value"];
					break;
				case "TAG3":
					$this->elementos[$j]->TAG3=$element["value"];
					break;
			endswitch;
		}
	}
}
?>
```

Creamos una nueva clase *prueba* la cual hereda a la clase *xml_class* y obtiene de ella el método *query($query)* y la variable *$url*. El método *load($query)* de la nueva clase se encarga de descargar y procesar el documento, a través del método heredado *query($query)*. Una vez obtenido el vector *$result* basado en el documento pasamos a recorrerlo y a “poblar” el vector de objetos que componen nuestro elemento.

De esta manera construimos un objeto en memoria con la información necesaria para nuestra aplicación. Puesto que tenemos un objeto podemos construir sobre el otros métodos para implementar conceptos de lógica de negocio o de representación.

Evaluación :: Comparación

Vamos a comparar los tiempos obtenidos por los distintos métodos.

Estos tiempos están basados en un fichero XML de noticias proporcionado por el proveedor de contenidos junto con una XSL para procesarlo. Las pruebas se han realizado sobre un máquina con SO Debian GNU/Linux 3.0 (Woody), Apache 1.3.26, Perl 5.6.1, PHP 4.1.2 y Cocoon 1.8.

	Cocoon	Perl::XSLT	PHP::XSLT	PHP::OO
Tiempo medio de respuesta (s)	0,065836751	8,762991898	0,37488363	0,541535433
Varianza del tiempo de respuesta (s^2)	5,08877E-07	1,336353434	0,002713808	0,025408015

Tabla 1 :: Resultados de la prueba

En el gráfico (Figura 1 :: Comparación de las distintas tecnologías) se puede ver graficamente la tabla anterior (Tabla 1 :: Resultados de la prueba)

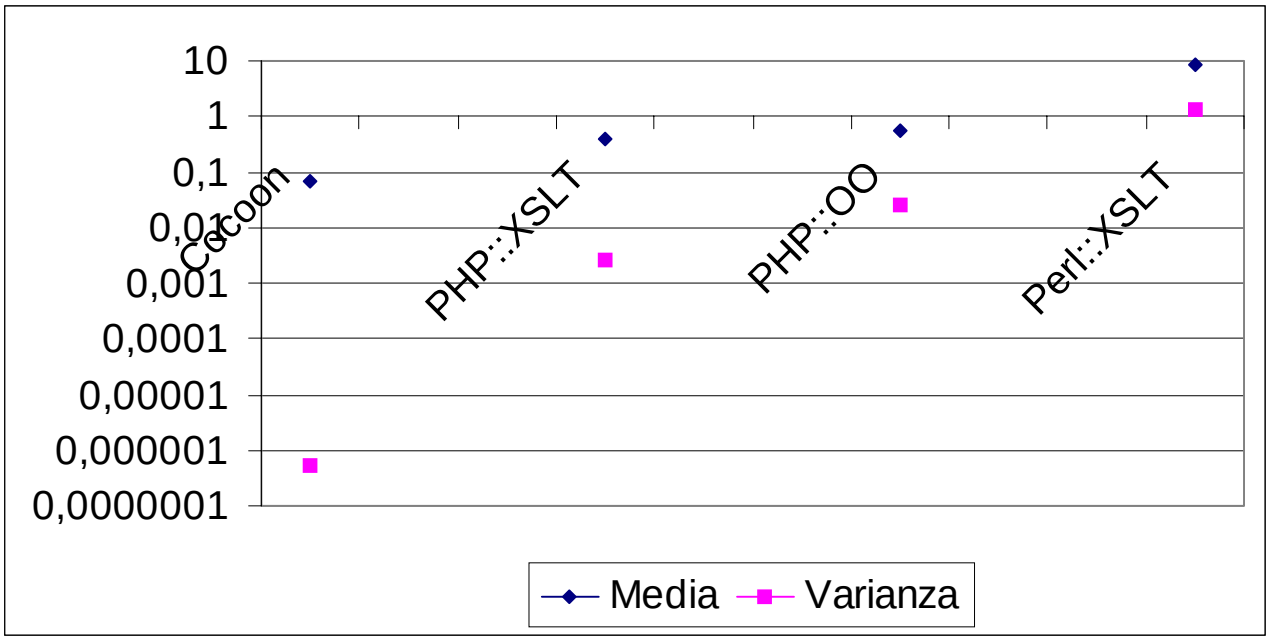

Figura 1 :: Comparación de las distintas tecnologías

Es fácil apreciar como Cocoon obtiene la mejor velocidad en las pruebas realizadas. Puede apreciarse como hay un orden de magnitud de diferencia entra la solución basada en Java::Cocoon y PHP, y entre PHP y Perl.

Conclusiones

Después de evaluar las distintas soluciones en cuestión de rendimiento y de facilidad de uso podemos obtener las siguientes conclusiones:

Rendimiento: Cocoon ha obtenido los mejores resultados en cuanto a velocidad y por mucha diferencia con el resto tecnologías probadas. La combinación CGI::Perl ha demostrado ser la solución más lenta de todas las analizadas. La diferencia entre PHP::XSLT y PHP::OO no es importante.

Facilidad de uso: Todas las soluciones que implican XSLT necesitan la construcción de su XSL asociada. Este es el verdadero trabajo a la hora de usar estas soluciones puesto que, como se ha podido comprobar, el uso de los distintos procesadores es terriblemente fácil. Por otro lado, la sobrecarga que implica el uso de PHP::OO a la hora de construir el vector de información se ve compensada por una integración con la página web más sencilla tanto para un programador desconocedor de XML como para el equipo de diseño.

Soporte tecnológico: Es indiscutible que la solución basada en Apache::JServ::Cocoon es la que mejor rendimiento es capaz de proporcionar de todas las evaluadas. Dejando aparte la facilidad o dificultad de enfrentarse a la generación de una XSL encontramos otro escollo a la hora de implantar una solución basada en esta tecnología: la infraestructura software necesaria. El soporte que las distintas empresas de alojamiento están dando al XML es muy reducido, a lo más, se encuentran soluciones de “parsing”. Integrar tecnologías como Cocoon basadas en servlets o incluso PHP::XSLT es, la mayoría de las veces, imposible.

Valorando todo lo anterior y dentro del proyecto web que nos ocupa, se ha tomado la decisión de optar por PHP::OO. Dado que el desarrollo del resto del portal estaba orientado a PHP::OO tanto el equipo de programación como el de diseño se encontraban familiarizados con esa tecnología. Por otra parte, las necesidades tecnológicas que implica son pequeñas, un parser XML, lo cual facilita la elección del alojamiento más oportuno sin que la integración XML sea limitante. Aunque no es la mejor solución en cuanto a rendimiento se ha considerado que es lo bastante buena como para que el rendimiento global del sitio web no se vea afectada.

Referencias

W3C::XML
www.w3.org/XML/

W3C::XSL :: W3C::XSLT
www.w3.org/Style/XSL

Perl
www.perl.com

theoryx5.uwinnipeg.ca/CPAN/data/XML-XSLT/XML/XSLT.html

PHP
www.php.net/

www.php.net/manual/en/ref.xslt.php

www.php.net/manual/en/ref.xml.php

www.php.net/manual/en/language.oop.php

Cocoon
xml.apache.org/cocoon/