Desarrollo de Aplicaciones Web con HTML::Mason

Francesc Guasch Ortiz

Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona
Univesitat Politècnica de Catalunya

Introducción

Mason es un sistema de desarrollo de webs muy potente y completo. Veremos brevemente algunas de las características más importantes. También se explicarán técnicas de gestión de proyectos y utilidades relacionadas.

Esta introducción al sistema HTML::Mason está orientada a responsables de proyectos web que quieran conocer una herramienta nueva y a programadores que quieran desarrollar aplicaciones en este entorno. Saber Perl es imprescindible para utilizar Mason, pero no es necesario conocerlo para entender este texto. Sin embargo, es necesario un mínimo de conocimientos de lo que es un servidor de web y para que sirve.

Panorama

Mason sirve para crear páginas web generando código desde el servidor. Nos permite mezclar Perl y HTML en el mismo archivo fuente. Éste código es ejecutado en el servidor de web, y se convierte finalmente en código HTML que se muestra en la pantalla del usuario final.

La unidad central de código en Mason es el componente. Un componente es un archivo de texto que contiene las fuentes de Perl y HTML. Éste fichero, permite aceptar parámetros y puede generar un resultado. Cualquier componente puede llamar a otro.

Mason es mucho más que un sistema para tener perl embebido en HTML. Además de las funciones de programación, tiene herramientas que ayudan a la resolución de los problemas típicos en la creación de aplicaciones web : caché, debug, plantillas, sesión, etc.

Mason se instala sobre el servidor de web Apache, y requiere un módulo de Apache llamado mod_perl. Funciona para todos los sistemas operativos en los que haya perl y apache. El sistema más popular donde se utiliza Mason es Linux, pero se puede instalar en cualquier otro Unix, e incluso Windows.

Mason al Microscopio

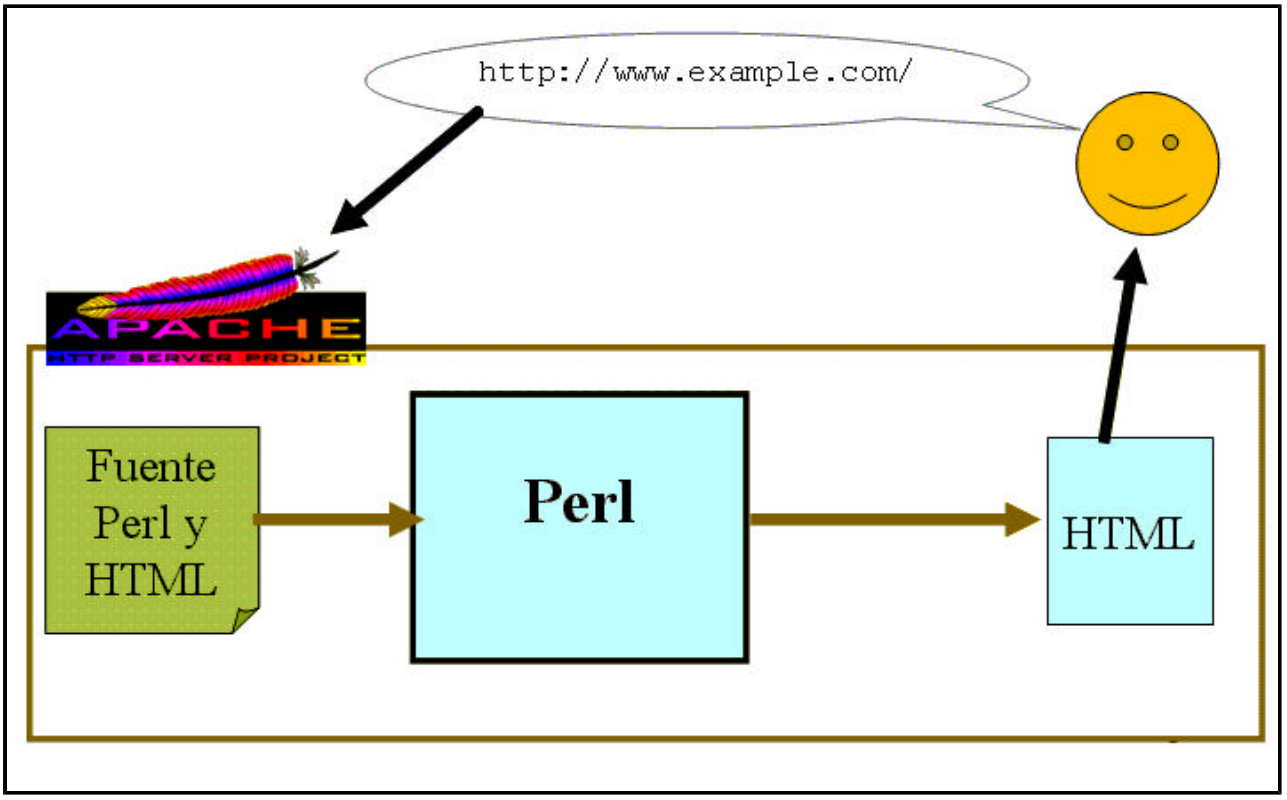

Fig. 1: *Fases de la Petición de una página.*

Desde que el usuario pide una página web, hasta que su navegador le muestra el código HTML generado: (*Fig. 1*)

1. El usuario pide a su navegador una página de web.
2. El navegador se conecta al servidor apache. Le pide la página que el usuario quería.
3. Dentro del servidor la página es un componente formado por Perl y HTML. Es el código fuente que ha escrito el programador. Mason lo transforma en perl puro.
4. El código perl generado se ejecuta y genera código HTML.
5. La página HTML se envía al navegador del cliente en respuesta a la petición, y se dibuja en la pantalla del usuario final.

El único trabajo del programador, dentro de éste laborioso proceso, es escribir el código fuente del componente Mason del punto 3.

Características y Beneficios

La principal ventaja que tenemos al utilizar Mason es que podemos programar en Perl.

Con Mason podemos crear componentes modulares reusables. Nos facilita el acceso a los parámetros de los formularios. Podemos crear plantillas sofisticadas, webs virtuales y filtros. Usando Mason evitamos repetir código, gracias a la modularidad de los componentes.

Además tenemos herramientas que van más allá de lo necesario para programar. Hay Gestión de Sesiones y técnicas de Orientación a Objeto. También nos da acceso a funciones internas avanzadas, como por ejemplo para Caché, que nos sirve para guardar el resultado generado por un componente.

Perl

Mason nos permite programar en Perl, y mezclarlo en el archivo fuente con código HTML para generar páginas web dinámicamente. Esto nos permite utilizar todas las características del lenguaje de programación

Perl en las aplicaciones web.

Perl es un lenguaje muy completo y evolucionado. Perl ya existía mucho antes de Internet, y tiene herramientas especiales que lo hacen ideal para éste entorno. Podemos bajar desde el web miles de módulos para todo tipo de funciones, ya implementados. Dispone de objetos, expresiones regulares, y muchas utilidades para hacer la vida fácil al programador. Por ejemplo, existen drivers para acceder a cualquier base de datos del mercado.

Con Perl podemos crear pequeños scripts que nos salvan la papeleta rápidamente, o se puede utilizar para aplicaciones complejas de cientos de miles de líneas de código con objetos. Por ejemplo, se utiliza en el proyecto del Genoma Humano o para gestionar webs tan complejos como amazon.com .

Con Mason, se puede incluir Perl dentro del código HTML. La sintaxis está totalmente integrada, con lo que no hay que aprender ningún otro lenguaje intermedio. Si necesitamos hacer comparaciones, bucles, o cualquier operación que se escapa al nivel HTML, lo podemos hacer directamente en Perl.

Componentes

Fig. 2: *Divisiones o Componentes de una página*

Mason tiene un sistema de componentes reusables. Con ésto vamos más allá de un script en el que podemos incluir otros archivos. Los componentes son sofisticados elementos, a los que podemos pasarles parámetros y devuelven un resultado. Los componentes pueden llamarse entre ellos, como si fueran subrutinas. Puede haber componentes que contengan código perl y HTML, o que sólo tengan código y no devuelvan ningún resultado visible para el usuario final.

Separemos las partes de una página de web típica (*Fig. 2*). Probablemente habrá una cabecera, un índice a la izquierda, enlaces a la derecha, un pequeño pié de página, y en el centro el contenido que ha pedido el usuario. Cada uno de estos apartados puede ser un componente separado. Al generarse el código HTML completo de la página, Mason va llamando por orden a cada uno de los componentes, y se insertan en el lugar apropiado.

La mayoría de las páginas de un web tendrán elementos iguales, por ejemplo, la cabecera. Teniéndola dentro de un componente aparte, nos evita tener que copiar la cabecera en cada página.

Puede haber componentes que no muestren ningún resultado visual, por ejemplo, un componente que guarde información en una base de datos. También pueden ejecutarse estos componente desde Mason

Parámetros

Los desarrolladores de Mason han resuelto de manera única y brillante el problema del paso de parámetros. Tanto los datos que el usuario rellena en los formularios, como los argumentos de llamada en los componentes, se tratan igual. El programador no tiene que preocuparse por los detalles internos de los parámetros de las páginas web, tanto si son tipo POST o GET, se accede de la misma manera sin complicaciones.

Cuando un usuario accede a una página en la que hay un formulario, rellena los datos y los envía al servidor. Al ejecutarse el componente correspondiente, el programador puede acceder a los datos enviados por el

usuario como si fueran parámetros de una subrutina.

De la misma manera, cuando un componente llama a otro, le puede pasar parámetros. El componente ejecutado puede acceder a estos datos de la misma manera que cuando un usuario rellena un formulario.

El sistema de paso de parámetros está pensado para facilitar la modularidad del web. Podemos separar cada bloque de la página en partes y ejecutarlos por orden.

Plantillas

Hay unos supercompontes en Mason que facilitan la creación de plantillas. Se acabó repetir código, copiar y pegar, o incluir páginas. Todo ésto se gestiona automáticamente.

Las páginas de un web tienen bloques comunes (*Fig. 3*), por ejemplo una cabecera o un pié de página. Podemos hacer llamadas desde cada página para mostrar la cabecera y el pié, pero eso nos obligaría a repetir el código de esa llamada en todas las páginas.

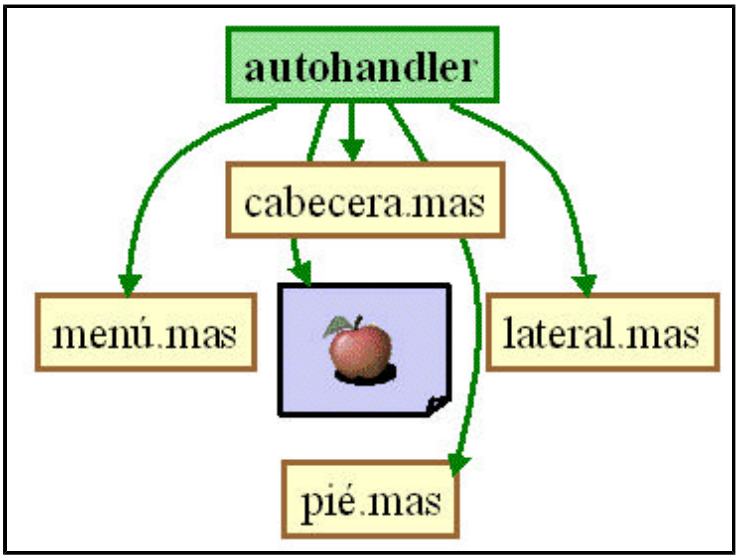

Fig. 3: *autohandler*

Hay una técnica que nos evita incluso ese trabajo repetitivo. Es un supercomponente de Mason llamado autohandler. Se ejecuta antes de la página real, y se encarga de hacer las llamadas a todos los componentes comunes , y en el momento justo mostrar la página real que se ha pedido.

URLs Virtuales

Existe otro supercomponente de Mason, llamado dhandler, que se ejecuta si alguien pide una página que no existe. Una de las utilidades es la creación de webs con URLs virtuales.

Podemos crear un apartado en el web que no esté formado por componentes reales. Al usuario final, le parecerá que existe un sistema de directorios y archivos, pero en realidad no existen.

Por ejemplo, una web de una revista de periodicidad mensual tendría unas URLs parecidas a esta, o incluso mucho más horribles:

```
http://revista.example.com/articulos?año=2003&mes=junio
```

En la página "artículos" habría una llamada a una base de datos para buscar los contenidos que hacen referencia a ese mes y ese año. Si no existen se mostraría un mensaje de error.

Con Mason podemos hacer que las URLs tengan un formato más elegante:

```
http://revista.example.com/articulos/2003/junio.html
```

Esta página no existe en realidad. El supercomponente dhandler se ejecuta, el programador escribe código para descomponer la URL y realiza el acceso a la base de datos. Se obtiene el contenido y se muestra en pantalla, si no existe el artículo responde con un error al usuario.

Filtros

Recordemos los pasos de ejecución de un componente. Mason transforma el código perl+html en sólo perl, y luego se transforma en el HTML que es lo que ve el usuario final.

Podemos añadir un filtro al componente para que transforme el texto generado. Por ejemplo, queremos transformar todas las llamadas a las imágenes de la página para que se busquen en otro servidor. Haríamos un filtro que hiciera una transformación. El código se parecería un poco a :

```
s|  img src="(/.*)"  |  img src="http://img.example.com/$1" |
```

No hace falta que intentemos tan pronto saber como hacer filtros. Pensemos que simplemente, busca el nombre de la imagen, la guarda en la variable $1, y esta variable se añade al web en el que hemos puesto los archivos.

Orientación a Objeto

Mason permite la utilización de herencia entre los componentes. Podemos programar métodos y atributos , y definir de que otro componentes se heredan.

Los objetos de Mason, no tienen nada que ver con los objetos de perl. Se utilizan para crear una jerarquía de padres e hijos entre los componentes.

Uno de los usos es la creación de plantillas sofisticadas. Podemos definir un título para cada página, las páginas que quieran tener su propio título, pueden crear el método para ello. En las que no lo tengan, se mostrará el título por defecto, el del padre.

Sesión

La Sesión del usuario se gestiona mediante un módulo de perl opcional. Es un sistema completo que permite guardar datos tales como números, textos y objetos.

La utilización de la sesión es muy simple, pensemos en ella como una variable global a todos los componentes. Y lo que guardemos dentro, permanece para siempre. Hasta que lo borremos.

Internamente, la sesión suele manetenerse con una característica de los navegadores web llamada *cookies*. Pero existen otras posibilidades para mantenerla, por ejemplo, añadir un identificador a la URL. Es un detalle técnico para la creación de webs que soporten cualquier tipo de navegadores.

El contenido de los datos de la sesión se guarda normalmente en una base de datos. Esto nos facilita la creación de granjas de servidores. Pensemos en un entorno con varias máquinas del mismo web, y un servidor de base de datos. Si alguno de los servidores de la granja de web no funciona, el usuario no lo notará. Por que otro de los servidores de repuesto tomará el lugar del que ha fallado. El nuevo servidor puede acceder a la sesión que está almacenada en la base de datos.

Caché

La ejecución de los componentes puede ser un proceso complicado, En algunos casos puede requerir largas operaciones, por ejemplo consultas a bases de datos o que requieren mucho tiempo de CPU.

Recordemos el apartado *Mason al Microscopio*, y la figura 1. El proceso de ejecución de una página hasta que genera el HTML tiene varias fases. La caché guarda el resultado final del componente. En muchos casos, podemos pedirle a Mason que recuerde durante un tiempo el resultado de esas operaciones complicadas. Así no hay que realizarlas cada vez.

Esta caché puede definirse para todos los documentos del web, y hacer que las páginas en general tengan un tiempo de expiración. Además, podemos declarar la caché para cada componente por separado.

El control de expiración de la cache, no tiene por que ser sólo un límite de tiempo. Podemos programar una función que indique si el contenido de una página debe volver a generarse. Por ejemplo, una consulta a una base de datos para comprobar si ha habido algún cambio. Sólo entonces refrescaremos el contenido de la página.

En webs que tienen identificación de usuarios, las páginas tienen un contenido diferente para cada persona que accede. Puede configurarse la clave de la caché para que tenga en cuenta el usuario que accede, y guarda el contenido diferente para cada uno.

Filosofía de Trabajo

Hasta aquí hemos visto algunas de las características más importantes de Mason, y su aplicación en el entorno web. Ahora veremos técnicas de trabajo profesionales con perl, y que posibilidades nos da para gestionar equipos de trabajo con muchos desarrolladores.

Daremos consejos de como organizar el código perl y la mejor manera de escribir fuentes claras y aplicaciones fáciles de mantener. También se recomienda la utilización exhaustiva de objetos. Es más complicado que escribir directamente todo el código en los componentes, pero nos da muchos otros beneficios.

Es recomendable tener varios servidores, separando los de producción de los de desarrollo. Con Mason y Apache se puede configurar un entorno de programación separado para cada programador, y también del sitio web al que acceden los usuarios.

La Pirámide del Código

Las manera más usual de programar webs es escribir código HTML mezclado con el lenguaje de programación que usemos. Con Mason también podemos programar así, intercalando los dos lenguajes. Sin embargo tenemos herramientas que podemos usar para mejorar el estilo de programación y trabajar de una manera más profesional.

Mezcla de código HTML y Perl

```
<%args>
        $familia => 10
</%args>
<form method="post">

<table border="0">
%       if ($opciones) {
                <& btn_guardar.mas &>
%       }
```

Pirámide del Código

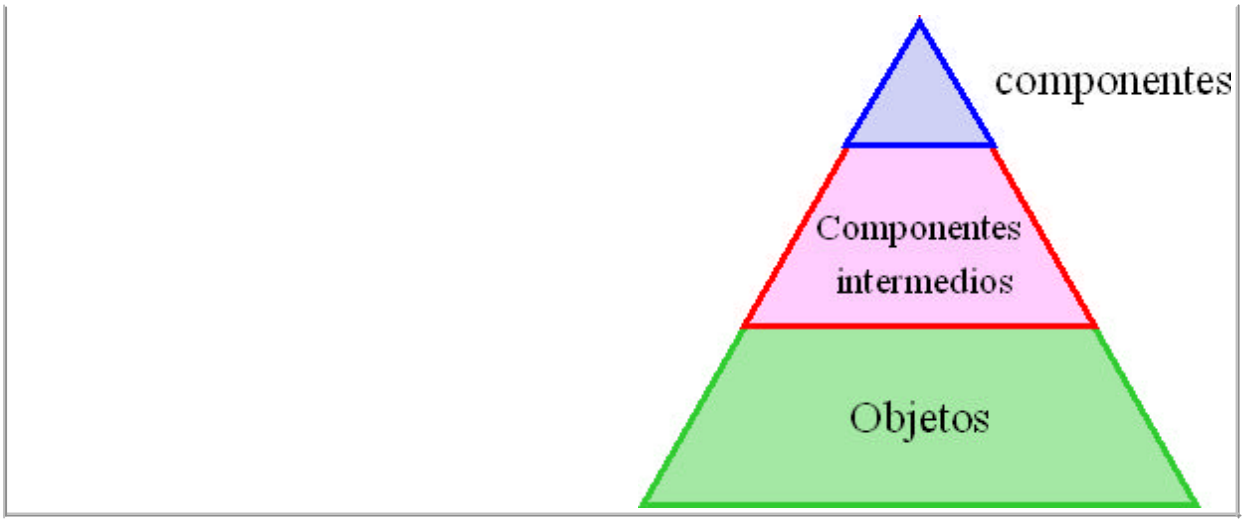

Fig. 4: *La Pirámide del Código*

La nueva directriz de programación nos aconseja que escribamos el código fuente de una manera piramidal (*Fig 4.*). En la punta tenemos los componentes más externos que muestran la página al usuario. En ese nivel tendríamos que utilizar el mínimo de Perl posible, y estaría la mayor parte del código HTML. En el nivel central de la pirámide tenemos componentes intermedios de Mason. En esos archivos deberíamos poner código HTML y Perl, con un poco más de Perl que en los componentes externos. Lo ideal sería que en los componentes intermedios hubiera las llamadas a los métodos de los objetos y poca cosa más. En la base de la pirámide tenemos los objetos, donde debe ir la parte más importante de código Perl. Estos objetos serían llamados desde Mason.

¿ Por que programar de esta manera ?

Si utilizamos asiduamente los objetos perl, nos beneficiamos de una aplicación más fácil de mantener, mejor probada y con mejor reutilización.

Suele acusarse a perl de ser un lenguaje complicado, y cuyo código fuente es ilegible y difícil de mantener. Que alguien puede escribir programas horribles no es una razón para que lo hagamos nosotros también, todo lo contrario. Con perl, podemos estructurar la fuente con objetos y módulos. Planificando bien la aplicación que vamos a desarrollar, y programando con objetos, tendremos un código fuente más claro para el equipo de trabajo, o para las personas que tengan que mantener el programa en un futuro. Incluso para nosotros mismos.

Perl nos facilita hacer "tests" de los objetos. Si alguna vez has utilizado un módulo de perl que te has bajado de CPAN, habrás visto que antes de instalarse ejecuta unas pruebas. Con un simple comando, el objeto es verificado para comprobar que todo funciona bien:

```
$ make test
t/utf8.....................ok
t/whitelist_addrs..........ok
t/whitelist_to.............ok
t/zz_cleanup...............ok
All tests successful, 2 tests skipped.
Files=34, Tests=259
```

Hacer esto es muy sencillo, una vez se conoce la técnica. El programador tiene que escribir pruebas para los métodos que va creando. Uno puede pensar que es muy pesado ir escribiendo los tests, pero es un trabajo que compensa hacer. Sin las pruebas automatizadas, ¿ Cómo podemos saber que todo funciona ? Quizá podemos ejecutar la aplicación e ir probando, o pedirle a los usuarios que comprueben si todo les va bien. Si lo pensamos fríamente, probar de esta manera nos hace perder más tiempo, y es menos concluyente que crear tests. El tiempo que nos cuesta hacer las pruebas se recupera con creces.

Podemos reutilizar los objetos que hagamos. El mismo objeto puede llamarse desde diferentes componentes. Incluso desde componentes que utilicemos en otros webs. También se puede utilizar el mismo objeto desde diferentes webs, lo podemos instalar en otro servidor y hacer la llamada correspondiente. Es mucho mejor que no copiar y pegar el código fuente.

Pero hay más, perl es un lenguaje de programación que también existe fuera del web, podemos escribir aplicaciones que utilicen los mismo objetos. Por ejemplo, hacer un pequeño programa que cada día nos haga un informe del estado de los pedidos del web y nos lo envíe por correo electrónico. Utilizando el mismo objeto sería una tarea trivial.

Desarrollo y Producción

Las fuentes de los componentes Mason y de los objetos Perl son archivos de texto. Existen herramientas que nos permiten tener un repositorio centralizado de las fuentes, una de las más populares es CVS.

El Repositorio de Código nos permite compartir las fuentes entre el equipo de trabajo y también aislar el desarrollo de la aplicación. Los programadores obtienen cada uno una copia de las fuentes y van haciendo su parte. Cuando tienen trozos terminados los envían al repositorio para que los demás desarrolladores los puedan descargar.

Puede marcarse parte de la fuente como estable o inestable. En el servidor de producción se instala sólo la parte estable. También pueden hacerse versiones de los archivos fuente y recuperar el código de la aplicación tal como estaba en una fecha determinada.

Mediante CVS los programadores pueden usar cada uno por separado su copia de las fuentes. Pero además tienen que tener en sus estaciones de trabajo un servidor de web apache con mod_perl y Mason instalado. Una manera fácil de trabajar en equipo es tener un Servidor de Desarrollo. En esta máquina, todos los programadores tienen su copia del código en su directorio personal, adem´s cada uno tiene un apache privado del cual puede modificar la configuración, parar y lanzar a voluntad, sin molestar a los demás programadores. Es útil tener la configuración de apache también dentro del repositorio del CVS.

Conclusión

Hemos visto las características más importantes del entorno de programación de perl y las capacidades avanzadas que nos permite la utilización de Mason para el desarrollo de aplicaciones Web. Además se han establecido unas guías recomendadas para la creación de aplicaciones de una manera profesional. Hemos visto como coordinar equipos de trabajo y se ha hecho hincapié en crear código ordenado y fácil de verificar. Aprender el sistema no es difícil, y muy pronto estaremos capacitados para programar webs en muy poco tiempo.

Referencias

Web

Núcleo
www.masonhq.com
perl.apache.org
Gestores de Contenido basados en Mason
www.oasyssoft.com
www.bricolage.cc

Bibliografía

Embedding Perl in HTML with Mason de Dave Rolsky y Ken Williams, O'Reilly and Associates