CaFe Perl v0.9 - Periódico de la Comunidad Perl de Capital Federal

Editorial

Hola Perl Mongers, bienvenidos a un nuevo número de CaFe Perl !!

En este último número del año quiero darles a todos un gran saludo, mis felicitaciones y deseos de lo mejor para cada uno de ustedes.

Simplemente comentarles que en este número no va a estar la sección Mordiditas de aquí y de allá, ni tampoco va a salir a mitad de mes como estuvo sucediendo últimamente. No se asusten, sólo es un pequeño receso.

Por otro lado tenemos una jugosa entrevista a Sebastian Riedel, principal arquitecto perteneciente al Core Developer Team de Catalyst

Espero que lo disfruten.
Hasta la próxima taza de CaFe Perl !!! ... eso sí, café del bueno ;-).

Víctor A. Rodríguez (Bit-Man)

PERlitas

Catalyst (http://catalyst.perl.org/) es un Framework para Web elegante, que soporta el patrón MVC así como un número de patrones web experimentales. Está altamente inspirado por frameworks como Ruby On Rails, Maypole, y Spring. Sebastian Riedel, miembro del Core Developer Team, es nuestro invitado en este número.

Por favor Sebastian, una introducción para el grupo CaFe.pm

Mi nombre es Sebastian Riedel y soy un joven hacker de Perl5/Parrot/Perl6 de 24 años que vive en el Norte de Alemania. Aprendí Perl hace 5 años, y trabajo como desarrollador Perl para T- Systems International.

Cuál fue su motivación para construir Catalyst ??

Comencé luego de mantener Maypole por algunos meses, y rápidamente llegué a la conclusión que había demasiados defectos, como el build en acciones crud, integración TT/CDBI demasiado estrecha, objeto request pero no response, mapping uri/acción demasiado estático ... la gente usaba varios trucos bastante feos para poner sus aplicaciones alrededor de Maypole y crud, lo que resultaba en más trabajo y código casi inmantenible.

Así que decidí escribirlo desde cero. Todo pasó demasiado rápido para la mayoría de los usuarios de Maypole, así que tuve que pasar el mantenimiento (a otra persona), y hacer un fork que nombramos Catalyst[1]

Muchos de los desarrolladores de Maypole me siguieron pronto, y el nuevo código atrajo nuevos desarrolladores. Hoy estoy muy contento que Catalyst es más popular que su predecesor.

Que habilidades (relacionadas a Perl o no) obtuvo mientras construía Catalyst ??

Aprendí mucho sobre los bus pre 5.8.x :) Catalyst es inteligente en cuanto a los features de Perl, casi al límite, usando atributos, herencia múltiple y cosas por el estilo.

También estudié los demás frameworks del mercado como Ruby-on- Rails, OI2, CGI::App, Struts, SpringMVC, Seaside... para robarles todos los buenas funcionalidades. :)

Qué fortalezas encuentra en Catalyst que no están en otros proyectos similares ?

De la que estoy más orgulloso es del mapping uri -> método, en mi modesta opinión no hay framework en el mercado que lo haga de una forma tan elegante. La mayoría de los frameworks usan un esquema fijo como el de /clase/método/argumento y re-escriben las uris entrantes con algo como mod_rewrite de Apache, en Ruby-on-Rails it's se llama Routes.

---

1 N. del T. Catalyst : catalizador, elemento que acelera la reacción que se produce entre otros dos

Algunos ejemplos de Catalyst vs. Ruby-on-Rails. :)

Mapping simple /controller/method/attr

```
package MyApp::C::Foo;
sub bar : Local {
    my ( $self, $c, @attrs ) = @_;
}
1;
vs.

class FooController < ApplicationController
    def bar
    end

end
```

El mapping de más alto nivel (toplevel mapping), coincide con /bar

```
package MyApp::C::Foo;
sub bar : Global {
    my ( $self, $c, @attrs ) = @_;
}
1;
vs.

class FooController < ApplicationController
    def bar
    end

end
ActionController::Routing::Routes.draw do |map|
      map.connect "bar", :controller => "foo", :action =>
"bar"
```

Mapping del mundo real, coincide con /index.html

```
package MyApp::C::Foo;
sub bar : Path(/index.html) { my ( $self, $c, @attrs ) = @_; }
1;
vs.
class FooController < ApplicationController
    def bar
    end

end
ActionController::Routing::Routes.draw do |map|
       map.connect "index.html", :controller => "foo", :
action =>  "bar"
```

Mapping de expresión regular, coincide con todo lo que termina con .html (y captura el snippet coincidente)

```
package MyApp::C::Foo;
sub bar : Regex(^(*.\.html)$) {
    my ( $self, $c, @attrs ) = @_;
}
1;
vs.
No estoy seguro que sea posible hacerlo en Routes
```

Detectó que algún módulo generado por el proyecto haya sido usado fuera de este ??

Si, especialmente HTML::Prototype es bastante popular.

Planea liberar algunos módulos como una ramificación de Catalyst ??

Lo hicimos con Module::Pluggable::Fast, HTML::Prototype... y se está trabajando en nuevo módulos como HTTP::Body y DBIx::Class (no es realmente una ramificación de Catalyst sino parte de la familia).

Qué limitaciones impuso Perl al proyecto ?

Recientemente estuvimos estuvimos experimentando con procesamiento asincrónico (piensa en POE) y Seaside (Smalltalk MVC) como continuación y reemplazo de las viejas sesiones, pero llegamos a la conclusión de que es muy difícil implementarlo en Perl 5.

Estoy esperando Perl 6, que hará todo esto terriblemente simple :)

Tiene alguna opinión o consejo sobre Perl 6 ?

Úselo ! (http://pugscode.org)

Paa alguien que recién comienza, le recomendaría olvidarse de Perl 5 y comenzar con Perl 6 ?

Todavía no, la implementación actual (Pugs) tiene un alto nivel de calidad, pero todavía hay bugs y funcionalidades pendientes. Sin contar con la falta de treeware[1].

Qué consejo le daría a los futuros diseñadores y hobistas que se enfrentan a un nuevo proyecto ??

No perderse en los detalles, tratá de darle un api bien diseñada y hacerla funcionar, así se puede hacer un "release early, release often". Si es lo suficientemente buena, más gente se unirá a tu proyecto y lo limpiará por vos :)

Esto no significa “escribir código horrible” :)

1 Software dentro del arbol perteneciente al desarrollo de Pugs

En qué partes del código le aconsejaría a un novato en Perl que mire, para tener un aprendizaje placentero ?

Bueno, hace poco tuvimos una discusión con perin en perlmonks sobre eso, hay demasiadas técnicas avanzadas en el código de Catalyst, así que no le aconsejaría a los codificadores "novatos" mirar en el. Es fácil usarlo para una codificador "novato", pero no entenderlo internamente.

Cómo se puede colaborar con Catalyst ??

Simplemente unirse a #catalyst en irc.perl.org y/o a la lista de e-mail, http://lists.rawmode.org/mailman/listinfo/catalyst

Hay alguna habilidad preferida para colaborar en Catalyst ?

Si, IRC es una habilidad que hay que tener, ya que discutimos casi todo en nuestro muy cativo canal #catalyst (irc.perl.org)... :)

Qué funcionalidades cree que no están presentes, y cuáles agregará pronto ??

Estoy portando Catalyst a Perl6.

Por lejos no soy el único trabajando sobre Catalyst, de hecho la mayoría de los commit son hechos por otros miembros del equipo. Nuevos plugins surgen cada día :)

Algún módulo favorito de CPAN ??

Muchos! :)

Devel::ebug

SVK

LWP

Test::Pod

Test::Pod::Coverage

Template::Toolkit

HTML::Mason

Class::DBI

DBD::SQLite?

Inline

YAML

Locale::Maketext::Simple

Class::Accessor::Fast

Class::Data::Inheritable

POE

DateTime

RPC::XML

Alguna experiencia (graciosa o no) que tuvo mientras construía Catalyst y que quiera compartir con nosotros ??

Hay algunos puntos que vienen a mi cabeza.

Ya que los desarrollos actuales sobre web frameworks están mayormente dominados por la gente de Ruby tuve que lidiar con ellos un tanto, y encuentro un tanto divertido cómo tratan de publicitar la falta de alternativas como una funcionalidad, en contraste con nuestro TIMTOWTDI. Más divertido es que otro frameworks prefieren imitar a Catalyst en lugar de unir fuerzas y producir uno superior. No es que no me guste la competencia, pero preferiría que desarrollen nuevas funcionalidades para robarlas nosotros :)

SudorNews

Perl Program Repair Shop and Red Flags

Mark Jason Dominus está haciendo otra de las suyas. Básicamente se trata de un libro en el cual se va a abordar el tema de cómo reducir ese código a la mitad de tamaño y con la misma funcionalidad (o sea el doble de valor pro línea !!) Más info en http://perl.plover.com/

Parrot 0.4.0

Parrot, la máquina virtual sobre la que correrá Perl 6 y otros lenguajes, ya cuenta con la versión 0.4.0. si quieren saber lo que se viene, entonces apunten sus browsers a http://www.parrotcode.org

Mini-intro-Tutorial de Catalyst

En este número hay una entrevista a Sebastian Riedel, creador y arquitecto principal de Catalyst. Si quieren interesarse, una forma es mirando un tutorial para aprender de que se trata. Aquí van más de uno : http://libertonia.escomposlinux.org/story/2005/8/18/194122/414 y http://libertonia.escomposlinux.org/story/2005/12/1/22557/4136 . También hay un muy divertido calendario en http://catalyst.perl.org/calendar/2005/

Etapas de un programador Perl

Así como hace tiempo circulaba, y aún circula, un texto sobre cómo programan distintas jerarquías dentro de una empresa, también en Perl tenemos lo nuestro. Entrá en http://prometheus.frii.com/~gnat/yapc/2000-stages/ y divertite.

Entrevista a Richard Foley

Si estás mas o menos entrenado en Inglés no podés perderte este Perlcast a Richard Foley : http://www.perlcast.com/audio/Perlcast_Interview_014_Foley.mp3

Perl del lado del cliente

Si te sentís raro cuando te hablan de programación del lado del cliente, y cuando explicás que usás Perl, entonces este artículo es para vos : http://www.perl.com/pub/a/2005/12/01/client_side_success.html

Traducción de "How to be a Programmer"

En http://ieschandomonte.edu.es/~ohermlope/mediawiki/index.php/Traduccion_de:_How_to_be_a_Programmer van a poder encontrar un trabajo en proceso del que, incluso, van a poder participar.

Calendario Perl 2005

La versión no oficial del Calendario Perl 2005 la pueden encontrar en http://web.mit.edu/belg4mit/www/ y seguramente van a encontrar cada una de las entradas mas que útil e interesante.

Coca-Cola Blak

De acuerdo a un release de prensa (http://www2.coca-cola.com/presscenter/nr_20051207_corporate_blak.html) la empresa Coca-Cola estaría lanzando una bebida con café agregado a la ya tradicional Coca-Cola ... ideal para cualquier programador !!!

El futuro de Emacs

Emacs 22 se viene con todo : soporte para Mac OS X y Cygwin, rueda del mouse y muchos otros (http://cvs.savannah.gnu.org/viewcvs/emacs/emacs/etc/NEWS?rev=1.1295&view=auto). La versión final no estará pronto, pero si será de lo mejor.

Cómo probar archivos y módulos

Si generás código seguramente lo probás, y lo más probable es que uses alguno de los módulos del namespace Test:: ... si te estás dando de cabeza en como probar archivos y módulos entonces lo tuyo está en http://www.perl.com/pub/a/2005/12/08/test_files.html

ActiveState discontinúa VisualPerl

ActiveState, reconocida por su port de Perl hacia Windows, estaba desarrollando un plugin para utilizar Perl dentro de Visual Studio (IDE de Microsoft). Debido a problemas de licenciamiento (de continuar este plugin y otros relacionados con Python y XSLT no podrían ser licenciados como open source). La escueta historia en

http://www.activeperl.com/Products/Visual_Perl/?mp=1

Programación lógica con Perl y Prolog

Sabemos que de los paradigmas de programación imperativo, funcional, orientado a objetos y lógico, Perl es bueno en todos salvo el último. Si juntamos Prolog y Perl se cierra el círculo : http://www.perl.com/pub/a/2005/12/15/perl_prolog.html

Larry Wall habla sobre Perl 6

En una entrevista de la revista Linux Format, Larry Wall habló sobre Perl 6 : de que otra cosa sse puede hablar en este momento, eh ??? (http://www.linuxformat.co.uk/modules.php?op=modload&name=News&file=article&sid=189)

Lineamientos del proceso de discusión de GPL 3.0 ...

Comenzó a rodar, y ya se anunciaron los lineamientos para el proceso de discusión de GPLv3 (http://gplv3.fsf.org/gpl3-process.pdf), cuyo primer borrador será lanzado en la conferencia ad-doc a realizarse en el MIT el 16 y 17 de Enero del 2006

Cómo escribir comentarios

Una entrada de blog (http://dkrukovsky.blogspot.com/2005/07/how-to-write-comments.html) que habla sobre si deben escribirse comentarios, si lo puedo hacer en 5 minutos y cómo comentar una clase. Tal vez lo más interesante no esté en la entrada del blog sino en los comentarios ...

Vinos, quesos, cultura y open source

En Francia, la tierra gourmet por excelencia, parece ser que el open source tiene sus más fuertes detractores en el departamento de cultura (http://www.fsffrance.org/news/article2005-11-25.en.html)

Las Olimpiadas, también Open Source

El Comité Olímpico Internacional (http://software.silicon.com/os/0,39024651,39154775,00.htm) está bregando por cambiar su infraestructura a una basada en open source para las Olimpiadas de Beijing del año 2008.

Película que documenta la creación de software

Hay gente que se preocupa por hacer software, y otra por filmar como lo hacen. Eso fue lo que hicieron en Fog Creek durante 12 semanas, filmando desde la concepción hasta la finalización (http://www.projectaardvark.com/movie/)

Java es tan de los '90

Parece ser que este lenguaje está perdiendo terreno en lo que es aplicaciones web no-empresarial frente a LAMP (Linux-Apache-MySQL-Perl/Python/PHP) y .NET (http://www.businessweek.com/technology/content/dec2005/tc20051213_042973.htm)

Generosidad open source

Es bueno divertirse. Es bueno ganar plata. Es muy bueno divertirse y ganar plata !!! El site Bounty County (http://www.bountycounty.org/) tiene esto en mente y muestra sites donde puede ganarse dinero haciendo open source, del una forma ingeniosa.

Por qué usar Gtk+

Siguiendo con la onda Gtk+ la gente de IBM DeveloperWorks tiene un interesantísimo artículo sobre cómo usar Gtk+, consideraciones para usarlo y los beneficios que provee (http://www.ibm.com/developerworks/opensource/library/os-gtk1/?ca=dgr-lnxw01WhyGTK)

POC (peace of code)

Antes que nada las respuestas a las preguntas del número anterior :

```
perl -e '@n=qw(bytes K M G T); $i=int(log($ARGV[0])/log(1024)); print
$ARGV[0]/1024**$i.$n[$i]' 3456
```

- **qué es una línea de comandos Perl ?** : básicamente es usar el switch -e de perl indicándole que a continuación no viene el nombre del archivo que contiene el programa perl sino el programa en si
- **qué operación realiza este programa Perl ?** : convierte el número pasado como parámetro a notación científica utilizando los sufijos Kilo, Mega, Giga y Tera pero teniendo en cuenta que estamos en un contexto binario donde el sufijo kilo es 2**10 (en un contexto decimal este representa el equivalente a 10**3)
- **Para qué se usa la relación de logaritmos log() ?** : porque la relación entre el sufijo K y el número a convertir es de este tipo
- **Alguna forma distinta de hacerlo ? (recuerden que TMTOWTDI)** : la forma más directa es reemplazar las divisiones por loops con restas, pero en general todas las formas que pude ejercitar fueron mucho más largas y prácticamente incomprensibles

Ahora el peace of code de esta semana !!!!!

```
my $home = $ENV{HOME}
            || $ENV{LOGDIR}
            || (getpwuid($<))[7]
            || die "Homeless boy\n"
```

- qué significado tiene el hash %ENV ??
- qué son $ENV{HOME} y $ENV{LOGDIR} ??
- que devuelve la función getpwuid() ??
- qué contiene la variable $< ??
- qué significado tiene el valor alhojado en $home ??