Perl on Rails

Post categories: Technology

Comments

Tom Scott | 17:30 UK time, Friday, 30 November 2007

Like most organisations the BBC has its own technical ecosystem; the BBC's is pretty much restricted to Perl and static files. This means that the vast majority of the BBC's website is statically published - in other words HTML is created internally and FTP'ed to the web servers. There are then a range of Perl scripts that are used to provide additional functionality and interactivity.

While there are some advantages to this ecosystem there are also some obvious disadvantages. And a couple of implication, including an effective hard limit on the number of files you can save in a single directory (many older, but still commonly used, filesystems just scan through every file in a directory to find a particular filename so performance rapidly degrades with thousands, or tens of thousands, of files in one directory), the inherent complexity of keeping the links between pages up to date and valid and, the sheer number of static files that would need to be generate to deliver the sort of aggregation pages we wanted to create when we launched /programmes; let alone our plans for /music and personalisation.

What we wanted was a dynamic publishing solution - in other words the ability to render webpages on the fly, when a user requests them. Now obviously there are already a number of existing frameworks out there that provide the sort of functionality that we needed, however none that provided the functionality and that could be run on the BBC servers. So we (the Audio and Music bit of Future Media and Technology - but more specifically Paul, Duncan, Michael and Jamie) embarked on building a Model-view-controller (MVC) framework in Perl.

For applications that run internally we use Ruby on Rail. Because we enjoy using it, its fast to develop with, straight forward to use and because we use it (i.e. to reduce knowledge transfer and training requirements) we decided to follow the same design patterns and coding conventions used in Rails when we built our MVC framework. Yes that's right we've built Perl on Rails.

This isn't quite as insane as it might appear. Remember that we have some rather specific non-functional requirements. We need to use Perl, there are restrictions on which libraries can and can't be installed on the live environment and we needed a framework that could handle significant load. What we've built ticks all those boxes. Our benchmarking figures point to significantly better performance than Ruby on Rails (at least for the applications we are building), it can live in the BBC technical ecosystem and it provides a familiar API to our web development and software engineering teams with a nice clean separation of duties with rendering completely separated from models and controllers.

Using this framework we have launched /programmes. And because the pages are generated dynamically we can aggregate and slice and dice the content in interesting ways. And nor do we have

About this blog

This is our new blog for BBC Radio Labs - a place where we show some of our prototypes for new sites and services. They are all at an early stage of development and some of them might not work quite right, some might look a bit sketchy and they may never be taken any further. They're what we call "betas". We'll write about every new beta we release on this blog so please play with them and come back here to let us know what you think. We'll also be writing about other things we're working on, how we do our work and anything else we think you might be interested in.

For the latest updates across BBC blogs, visit the Blogs homepage.

Subscribe to Radio Labs

You can stay up to date with Radio Labs via these feeds.

Radio Labs Feed (RSS)
Radio Labs Feed (ATOM)

If you aren't sure what RSS is you'll find our beginner's guide to RSS useful.

Other BBC blogs

- BBC Backstage Blog
- BBC Internet Blog
- BBC Journalism Blog
- BBC RAD Labs Blog

to sub divide our pages into arbitrary directories on the web server - the BBC broadcasts about 1,400 programmes a day which means if we created a single static file for each episode we would start to run into performance problems within a couple of weeks.

Now since we've gone to the effort of building this framework and because it can be used to deliver great, modern web products we want to use it elsewhere. As I've written about elsewhere we are working on building an enhanced music site built around a MusicBrainz spine. But that's just my department - what about the rest of the BBC?

In general the BBC's Software Engineering community is pretty good at sharing code. If one team has something that might be useful elsewhere then there's no problem in installing it and using it elsewhere. What we're not so good at is coordinating our effort so that we can all contribute to the same code base - in short we don't really have an open source mentality between teams - we're more cathedral and less bazaar even if we freely let each other into our cathedrals.

With the Perl on Rails framework I was keen to adopted a much more open source model - and actively encouraged other teams around the BBC to contribute code - and that's pretty much what we've done. In the few weeks since the programmes beta launch JSON and YAML views have been written - due to go live next month. Integration with the BBC's centrally managed controlled vocabulary - to provide accurate term extraction and therefore programme aggregation by subject, person or place - is well underway and should be with us in the new year. And finally the iPlayer team are building the next generation iPlayer browse using the framework. All this activity is great news. With multiple teams contributing code (rather than forking it) everyone benefits from faster development cycles, less bureaucracy and enhanced functionality.

Comments

At 11:30 PM on 30 Nov 2007, *Steve* wrote:

Sounds very interesting. Is the plan for the framework to be internal to the BBC only or will it eventually be made available for external developers too?

Also does it integrate with Barley or does it go in a different direction?

At 12:26 AM on 01 Dec 2007, *schmidt349* wrote:

Why did the BBC go to all the trouble of replicating functionality that's been available on Perl for years now via Catalyst?

At 12:37 AM on 01 Dec 2007, *Anonymous Perl Lover* wrote:

Any reason U didn't use Catalyst, Maypole, Combust, CGI::Application, CGI::Prototype, or any of the dozens of other perl MVC frameworks?

Catalyst was around long before Ruby on Rails (possibly before the Ruby language for that matter), but never made the kind of headlines RoR gets. The Ruby community seems to be much better at mobilizing.

Actually, I think it's the Perl community's TMTOWTDI lifestyle. In Ruby, for small things *maybe* U use Camping, but you'll probably use Rails and for everything else you'll definitely use Rails. There are some others, but only the developers of them use them. In Perl, literally everyone writes their own.

Inferior languages like Java and C# rose up real quick and stayed there--keep getting bigger even--because they limit their users' choices. Perl stayed in the background and is now dying because it believes in offering as many choices as possible. That's why Perl 666 is going to be more limiting. As U can tell from my subtle gibe that the next version of Perl is evil, I prefer choices. But developers like me are a dying breed.

Developers now-a-days need cookie cutter, copy&paste code. When's the Perl on Rails book going to be released? Probably around the time the Catalyst one is. Or the CGI::Application one.

Bleh. I wrote way too much. U can't even put this up now, it's too long. I didn't realize I was so annoyed by the one jillion perl MVC web frameworks and how they're just one tiny example of why perl is dead.

At 01:20 AM on 01 Dec 2007, *Anon* wrote:

> The Ruby community seems to be much better at
> mobilizing.

I really think that first video demo of RoR using Textmate is what had a large effect. Before that, I don't remember seeing hardly any videos of development happening right in front of your eyes.

- BBCi Labs Blog

You watched the video thinking, "wow! it's so fast and easy! I'm gonna get in on that!". When, in reality, any good programmer using any good environment can make a software look good like that (if they practice a bit beforehand).

As an aside, anyone know of a video demo podcast for Catalyst?

At 07:14 AM on 01 Dec 2007, *Dave Cross* wrote:

Others have already commented that you seem to be reinventing the wheel here. No-one seems to have mentioned the Perl MVC framework which (in my opinion) is most like Rails - it's called Jifty (https://jifty.org/).

But there already parts of the BBC who are using Catalyst with great success. Why didn't you ask around a bit before embarking on what was presumably not a trivial task?

At 09:52 AM on 01 Dec 2007, *Raips* wrote:

How about BBC doing same as New York Times did?

https://www.linux.com/feature/120359
https://open.nytimes.com/

At 10:30 AM on 01 Dec 2007, *Mark* wrote:

To put the question a lot of people have been asking a slightly different way, what were the special constraints of the BBC servers that made it impossible to use existing frameworks?

And what features did you find were missing from the existing frameworks?

Thanks for sharing your experience.

At 10:30 AM on 01 Dec 2007, *Josh* wrote:

Developers now-a-days need cookie cutter, copy&paste code..
That's not the case at all. If you think you're going to copy and paste code to make a real application using Rails, you're very, very wrong.

Having a minimal skeleton that lets you poke the database is a looooooong way from something you'd actually use, unless your application is very, very simple.

But having the skeleton is still great, because it saves you from doing the tedious set-up work. You can start coding more interesting things right away, and it's much faster to remove the bits of a skeleton that you don't want than it is to write the parts you do want from scratch.

I love perl, and since it's the highest common denominator on the systems I administer, I won't be leaving it any time soon. But I hope to apply some of what RoR has taught me MVC and automation to my work in Perl.

At 12:32 PM on 01 Dec 2007, *Duncan Robertson* wrote:

Just thought I should clarify a few things, that maybe answer some of the questions above.

1. We actually started using Ruby on Rails a few years ago.
2. As Tom said, we use it for most of our internal projects (we've also used camping), as well as using standalone Ruby, Python and Perl apps for other stuff.
3. We like Ruby on Rails and although there were other frameworks to choose from (we tried many) Rails was the one that stood out, this is likely because Ruby is such a nice language to hack with.
3. On the live environment we were told at the time we had Perl 5.6, and a few BBC approved perl modules. Nothing more! So that meant that catalyst a other solutions were out.
4. The current framework attempts to draw all the bits from RoR we like, then extended to be more suitable to the BBC environment. For example, we needed to expose any SQL queries we would make so they could be vetted by DBA's for optimization. This made using an existing ORM more difficult so we had to craft slightly different solution.

Steve: It does integrate with barley and also baresque (the new version)

At 12:43 PM on 01 Dec 2007, *Duncan Robertson* wrote:

Just thought I should clarify a few things, that maybe answer some of the questions above.

1. We actually started using Ruby on Rails a few years ago.
2. As Tom said, we use it for most of our internal projects (we've also used camping), as well as using standalone Ruby, Python and Perl apps for other stuff.
3. We like Ruby on Rails and although there were other frameworks to choose from (we tried many) Rails was the one that stood out, this is likely because Ruby is such a nice language to hack with.
3. :Dave: and others, On the live environment we were told at the time we had Perl 5.6, and a few BBC approved perl modules. Nothing more! So that meant that catalyst a other solutions were out.
4. The current framework attempts to draw all the bits from RoR we like, then extended to be more suitable to the BBC environment. For example, we needed to expose any SQL queries we would make so they could be vetted by DBA's for optimization. This made using an existing ORM more difficult so we had to craft slightly different solution.

:Steve: It does integrate with barley and also baresque (the new version)
At 05:36 PM on 01 Dec 2007, *Tyler Gee* wrote:

Is it something you are releasing to CPAN?

At 06:13 PM on 01 Dec 2007, *4thwave* wrote:

I can understand the frustration of building another MVC. But, it needs to be understood, what are some of business constraints BBC and other organisations have in building MVCs.

My team also built a MVC based on our business requirements. But, have we reinvented the wheel. Certainly not. We had certain business requirements to fulfill, and most other MVC would not be acceptable.

What were the reasons
- multi language support (33 languages support), find any MVC can do this correctly.
- perl system (because most of the developers know Perl)
- syndicate content, rss feeds, 3rd party delivery, mobile content
- a friendly user interface for journalists/editors and producers.
- plus many other features...

Again, no MVC can have these features, so cut and paste, just doesn't cut it.

But did we build everything from scratch. Definitely not. We built a a simple, understandable framework, using CGI::Application, Template::Toolkit, DBIX::Class, Xapian, two home grown libraries and unfortunately DBI::Class.

The interface has been built using dojo, tinymce and prototype.

Yes, re-use successful libraries, but expect to be writing code if you have to fulfill your business needs.

At 06:17 PM on 01 Dec 2007, *drjonss* wrote:

so will this project ever be released to the general public or what?

At 06:32 PM on 01 Dec 2007, *Another Anon* wrote:

As they say in certain parts of the web, code or it didn't happen.

At 10:08 PM on 01 Dec 2007, *Alhazred* wrote:

Feh

MVC and front controllers are a giant boondoggle for web apps. Like pounding a square peg into a round hole. KISS.

Same with ORM. Every ORM framework is 10x more trouble than it is worth. There are better ways. An RDBMS is RELATIONAL, trying to make it look OO is just a terrible idea. Look at every large high performance OO DB backed app. Every one that tries to use ORM has to work around it for 50% of their functionality because by their very nature ORM solutions cripple you.

One day these fads will pass. Well maybe not since real actual system/software analysis and design is a lost skill...

At 10:09 PM on 01 Dec 2007, *Mark* wrote:

Please say more about Perl's multi-language support relative to that available in other languages (Python, Ruby).

What are you using for authentication / authorization and session management? Home grown, or some CGI::Application plugins?

At 11:48 PM on 01 Dec 2007, *railscoder* wrote:

"Why did the BBC go to all the trouble of replicating functionality that's been available on Perl for years now via Catalyst?"

Perhaps this will enlighten you:

https://readlist.com/lists/lists.rawmode.org/catalyst/0/1377.html

.

At 11:59 PM on 01 Dec 2007, *Vlad R.* wrote:

Well, I have not looked at your "onrails" thingy, it might be interesting by itself (or not).

However the problem on having too many files on a filesystem has an extremely simple solution.

you gzip them
you put them in a tar (say, 10-30 000 simple smallish ones into one tar of, say, 100-200MB)
and you serve them in 0.010 seconds or so from this tar file. (The time is around 0.050 seconds if the filename lookup before extraction included on a 7-year-old 500MHz Pentium machine, to give you an idea)

As most of web traffic should go out compressed already, you do not even un-gzip them.
Do not simply forget to index the tar. THAT's YOUR SIMPLE SOLUTION.

I.e. you drop the number of files in a directory from say each 30 000 to exactly 2, and you serve any random file in less than 0.05 of a second on an OLD server.

I do it, it simply works

At 02:08 AM on 02 Dec 2007, *Mark* wrote:

>Do not simply forget to index the tar. THAT's YOUR SIMPLE SOLUTION.
>serve any random file in less than 0.05 of a second on an OLD server.

Code or it didn't happen.

i.e. what tar options do you use to achieve this? Are you using a memory-mapped file, which I guess could make things even faster?

At 02:43 AM on 02 Dec 2007, *David Garamond* wrote:

Someone above wrote that Catalyst were born long before Ruby on Rails or even Ruby. I doubt it. A little poking around shows that Catalyst were first released around 2005. I believe it, as well as a lot of other frameworks, is inspired by RoR.

At 05:01 AM on 02 Dec 2007, *Nilson Santos F. Jr.* wrote:

Catalyst was inspired by both Rails and Maypole (an older web framework for Perl).

Nowadays, the combo Catalyst+DBIx::Class+TT (Template Toolkit) is probably a lot more powerful than RoR. But Rails was available first, in 2004.

At 11:30 AM on 02 Dec 2007, *Ed* wrote:

Just curious, but if the BBC is such a heavy user of perl why was no one from the BBC at the London Perl Workshop on Saturday? As this post demonstrates you obviously have more than a few interesting tips to share. I'd love to learn from you guys.

You seem to want to encourage re-use. As someone who once worked at a large technical organisation with 'competing' departments and projects, you might consider the approach of generating noise

outside of the organisation (ie in the open source community) as a way of increasing adoption within the organisation.

Hope to see you at future London perl events.
At 01:07 PM on 02 Dec 2007, *Dave Cross* wrote:

There were at least three people from the BBC at the London Perl Workshop. And one of them gave a talk (albeit a lightning talk).

At 02:50 PM on 02 Dec 2007, *Yair Lapin* wrote:

We started to work with the MVC framework CGI::Application one year ago. The framework permits us sharing develope work between several people. Before people can't read the code someone else wrote and people did more bugs. It has a lot of plugins , it's really a great module because it's small and easy to install. You can add more features using plugins.
Catalyst is a big horse, it's hard to install, i went into troubles with it. It's very complex framework, it try to do everything then it's difficult to use for simplest applications. CGI::Application has the most catalyst features. I think it's the same team that is updating catalyst and CGI::Application.

At 07:00 PM on 02 Dec 2007, *Aaron Trevena* wrote:

Catalyst is newer than Rails and, frankly, it shows.

Maypole is older than Rails, and was built by a single developer in a fraction of the time.

Rails is still pretty much the new kid on the block when it comes to MVC : Struts and other frameworks have been around for ages.

Anyway - as others have said - open source it please : I've paid my license, releasing the code would deliver far better value than much of the television output.

At 07:04 PM on 02 Dec 2007, *Peter Cooper* wrote:

On the live environment we were told at the time we had Perl 5.6, and a few BBC approved perl modules. Nothing more! So that meant that catalyst a other solutions were out.

It sounds like the BBC, despite the claims of it keeping up with the times, gives its developers few choices and drives them to hack around ye olde technology (not even Perl 5.8?). It's surely a drag to be within an organization that imposes such restrictions on people eager to be bringing cutting edge technology to the people. In a big blue-chip, this is a good thing.. in a progressive, publicly funded organization, definitely not.

It's understandable for the BBC to want non-progressive stalwarts in most of its departments, but in technology? How can the BBC recruit, keep, and, most importantly, *nurture* cutting edge talent in the long term? If they can't, as licence payers we all lose.

At 08:54 PM on 02 Dec 2007, *Vlad R.* wrote:

Reply to "Mark" -- HOW TO index a tar file

"Mark" wrote "code it or it won't work".

Indeed. Ordinary "tar" files are a linear concatenation of members of the archive, and so are very simple to index. Basically, it's "filename" - byte offset position.

I use a tiny utility I picked up from some unrelated software bundle (devoted to biology research).
It works as described - less than 0.05 seconds to exctract a random small text file from a 100-200 MB tar archive. I gzip them before putting into the archive, which is less efficient that gzipping the whole, obviously, but is more than tolerable as far as space efficiency is concerned. Byte streams, as obviously, can be put into archives withour compression.

This simple utility allows one to turn tar files into a sort of read-only storage filesystem with random access, and I use it, as one example, in web projects when one has to periodically archive messages/postings by the users that accumulate with time, and still let readers get the old data quickly and randomly at will.

It's surprising that most sysadmins are simply not aware of this simple, obvious solution.

To index tar files one can either pick up one of the existing utilities, or write one himself.
I have mine, use it, ane live happily.

At 11:10 PM on 02 Dec 2007, *Matt.S* wrote:

"It sounds like the BBC, despite the claims of it keeping up with the times, gives its developers few choices and drives them to hack around ye olde technology (not even Perl 5.8?). It's surely a drag to be within an organization that imposes such restrictions on people eager to be bringing cutting edge technology to the people. In a big blue-chip, this is a good thing.. in a progressive, publicly funded organization, definitely not"

You've got a very good point there. But the BBC faces a number of problems... One is that so much of the BBC's technology infrastructure has been sold off to Siemens. Want them to do something new? You'll need to wave some money under their nose.

Another is the scale of the BBC's live website infrastructure. Chances are (and I don't actually know this) it isn't as substantial as you might expect, and indeed it probably should be. As a result the guys running it have to be really careful they don't break it. They have to be careful anyway. They're running one of the most popular website domains in the world.

And don't underestimate how long it takes the Beeb's own technology ecosystem to react to change. OK I'll admit that I work for the BBC. Not in technology though. I'm seeing things happen now that should have happened at least five years ago. It's strange. Everyone you speak to seems to realize change is vital, yet nobody seems to think it will happen. Then suddenly it does... late.

Personally, as someone who writes for bbc.co.uk, I'm over the moon than dynamic elements are starting to appear on the sites. And I'm looking forward to English Regions content appearing on /programmes

At 10:20 AM on 03 Dec 2007, *Poncho* wrote:

Is 'Ruby on Rail' the monorail version? :P

At 02:12 PM on 03 Dec 2007, *john* wrote:

I would love to see this open sourced.

At 03:30 PM on 03 Dec 2007, *mofino* wrote:

So where is it? Less talk, more code!

At 06:33 PM on 03 Dec 2007, *Tom Scott* wrote:

Wow! Well this certainly generated quite a lot of interest. I'm not sure that I will be able to address everyone's issues - especially all the folk over at slashdot but I'll write another post to address as much as I can, in the meantime I just wanted to pick up on a few of the major themes.

Why didn't we use Catalyst or something else that already existed? As Duncan indicated the simple answer is because we can't install Catalyst etc. on the live environment. The BBC's infrastructure is limited to Perl 5.6 with a limited set of approved modules and there are further limitation on what is allowed (making unmoderated system calls etc.)

Access to the code: I'll see what I can do. The BBC does open source some of its code at https://www.bbc.co.uk/opensource/, I'm don't know if we will be able to open source this code but I'll let you know. However, its worth bearing in mind that we ended up writing this app to work within the BBC's infrastructure (Perl 5.6, BBC approved modules etc.) so if we did release the code under an OSS license we would still need to maintain this requirement (clearly the code could be forked etc.)

Too many files on a file system: @Viad R. nice solution and yes that solves the seek time issue - unfortunately it doesn't solve the other problems we needed solving. These include building the sort of interactive products we want to build; nor how to maintain up to date links between pages - when we publish information about a programme we not only need to publish a page for that programme but also update a large number of other pages that should now link to it/ or remove those that shouldn't - trying to work out what pages to update becomes a nightmare, its much easier to render the pages dynamically.

BBC web infrastructure - not sure were to start with this one. You my find this hard to believe but the vast majority of bbc.co.uk is published statically - people write HTML and FTP those files onto the live servers. This provides a reliable service, if not a dreadfully exciting one. Its also clearly restrictive which is why we needed to build a solution like this in the first place, rather than using an existing framework. Now I'm very aware that this whole endeavor - building 'Perl on Rails' - seems bonkers to most people outside the BBC. But what isn't bonkers is the fact that we have built an elegant application (with a

small team) and deployed it within the constraints of the infrastructure and in doing so delivered a new product to our users and helped move the debate on inside the BBC.

At 11:17 PM on 03 Dec 2007, *James Richardson* wrote:

Perl and static files? - Heck

Its a shame that the BBC's insane outsourcing and/or IT security strategies have limited you to this.

While developing a cool application is great - its certainly not so great if there was no need for it. Its called reinventing the wheel, and theres a whole bunch of it going on. In fact its close to scandalous, and you admit it yourself. I'm not even going to mention the iPlayer.

It seems that the BBC is totally lacking in the skills in how to manage a modern hardware infrastructure, and short sighted enough that they give their IT teams such blunt instruments with which to work. Its certainly not about money saving... any number of people will show you that commodity hardware and open-source software will serve plenty of content - just ask facebook, last.fm and google...

What do the IT managers do all day, if they don't campaign against this?

Many organisations are finding that a lighter touch on pointless restrictions, coupled with greater responsibility given to teams actually makes for a much more secure, more responsive development environment - as people don't tire themselves out fighting against the system. They start to succeed because of the system, not in spite of it.

At 11:00 AM on 04 Dec 2007, *Firmicus04* wrote:

>As an aside, anyone know of a video demo podcast for Catalyst?

https://dev.catalyst.perl.org/wiki/Movies/

At 03:24 PM on 04 Dec 2007, *Andrew Bowden* wrote:

I suspect reading this that there are some commenters who haven't had the "fortune" to code for a large website like the BBC's! Having coded in the past on the BBC's enviroment, I know full well the restrictions you're forced to work in are pretty tough!

However some restrictions have to be. If you have too much of a free-for-all, you end up with code written in 101 different ways that no one understands, and no one can maintain. You also get scared of upgrading your servers because everything is liable to break! (Been there, done that!)

However, as I think we can see here, give someone a challenge, and they'll make it happen!

At 05:46 AM on 05 Dec 2007, *Makea* wrote:

Nothing wrong with Perl or static html.

In fact, we're in the process of writing modules that generate static html files from our php based Drupal CMS.

Dynamic content is nice, but when the majority of your page content (scientific news in our case) never changes, static make more sense.

Static pages allow you to serve much more on equivalent hardware than dynamic pages.

Fascinating article. A lot of companies are in the same boat with their outdated versions of perl.

At 12:09 PM on 05 Dec 2007, *Nick Reynolds (editor)* wrote:

See also this post from James Cridland -

https://www.bbc.co.uk/blogs/bbcinternet/2007/12/external_web_infrastructure.html

At 04:47 PM on 07 Dec 2007, *jbooe* wrote:

The BBC should be thankful they weren't able to install Catalyst.

At 05:35 PM on 07 Dec 2007, *clm100* wrote:

You guys should have just gone with sqlonrails. It's probably a better idea than old perl on rails, anyways!
At 09:40 PM on 20 Dec 2007, *mw487* wrote:

Vlad, of comment 27 on indexing of tars of gzips-

(More) code or utility names or links would be appreciated. Googling things like tar, gzip, index, file system, utility, and program does not seem to help.

Wouldn't you need the stream start and end?

Anyway, links would be much appreciated.

At 01:05 AM on 31 Dec 2007, *Vlad R.* wrote:

reply to 40:

One is called tarix, and can index both GNU tar archives and compressed (zlib) GNU tar archives (i.e. first tar, then compress - and get them quickly from any place in the archive because both compression and the tar itself are indexed)

Currently it is designated by it author as being in version 1.0.2.
It can be found through www.freshmeat.net and/or the www.sourceforge.org site

TARIX IS SLOWER, it can give you response times like 0.3 seconds instead of 0.05 because of the way it treats the index (which in case of tarix is just a text file with numerical offset of the beginning of the file in the tar archive, then the length of it, if I remember correctly, adding one more parameter if overall compression was used).
I.e. getting the file is quick, but consequtive reading of the index to get the offsets may slow the extraction down. This of course can be further optimised.

THE SECOND OPTION is a much faster set of utilities, as tiny as tarix (say, 20 kb each), but they are based on a (small, 200-300 kb) library, which one needs to hack out of a huge project for biology research. When extracted to become an independent source tree, the 3 utilities become very useful indeed.
They are my preference, even though they do not allow compression of the formed "tar", and I often prefer to compress each individual file that goes into the archive as a first step instead.

At 02:22 PM on 25 Feb 2008, *Karim Ahmed* wrote:

In fact, we're in the process of writing modules that generate static html files from our php based Drupal CMS.

I'm interested to know why the BBC don't get the best of both worlds (static and dynamic) by using some intelligent caching of pages and content blocks?

Maybe I have misunderstood the technical problems but why would the mere existence of a dynamic back-end have an effect on server performance when, for instance, 90% of requests would be served from a file-system cache? Surely you could then do more exciting things with the content.

Regards,
Karim

This post is closed to new comments.

More from this blog...

Topical posts on this blog

What happens to The Proms after the Royal Albert Hall? (13)
Podcast OPML - change (5)
RealMedia - follow up (30)
RealMedia - an update (76)
BBC Radio Waves - exploring what we play (4)
Immersive audio for Planet B (8)

Archives

Past twelve months
October 2010 (1)
April 2010 (1)
March 2010 (1)
January 2010 (1)
December 2009 (3)
November 2009 (2)
October 2009 (7)

Categories

These are some of the popular topics this blog covers.

beta conferences design flash gaming links live events mobile music programmes r&d radio realmedia realplayer streaming

Latest contributors

Jem Stone
Tristan Ferne
Chris Bowley
Duncan Robertson
Michael Smethurst
Alan Ogilvie
Mark Kortekaas

Windows Media - Listen Again - Update (25)
What's your musical taste? (5)
Windows Media - Listen Again - Update 2 (11)
More Mooso (2)

September 2009 (1)
August 2009 (2)
July 2009 (2)
June 2009 (4)
May 2009 (1)
complete archive

technology vistrial visualising radio windows media

Caleb Knightley