XSLT
as template engine

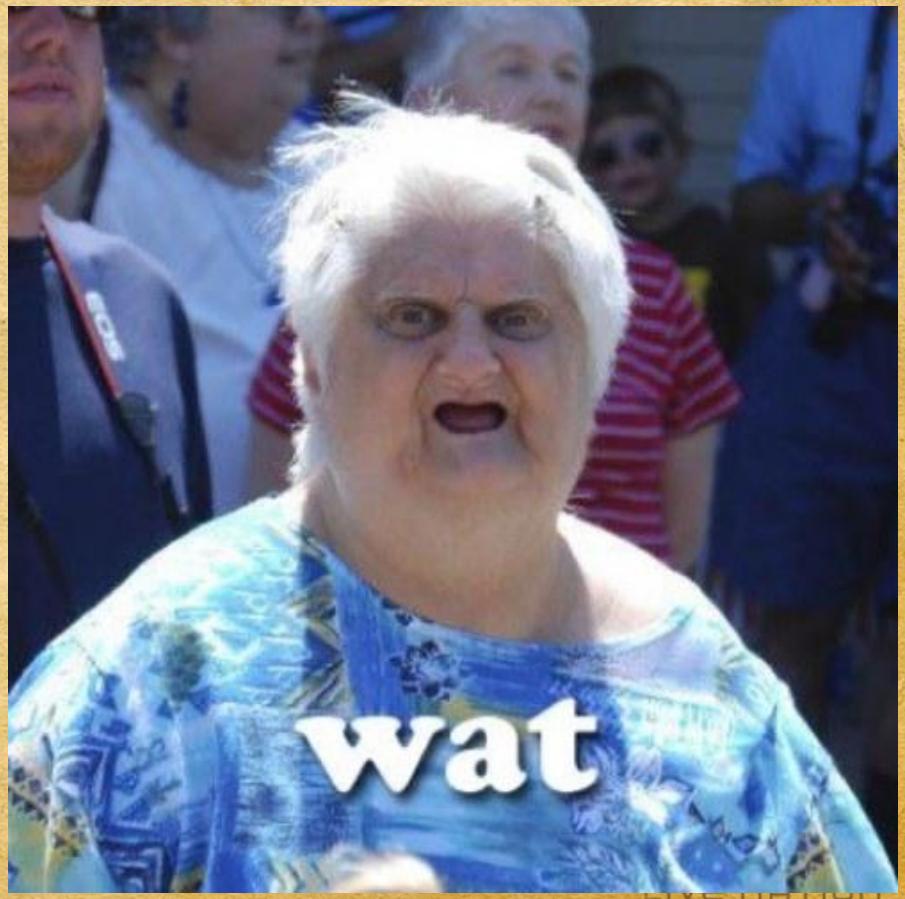
...or why do I love XSLT

LIVE NATION

LIVE NATION

Who am I?

cono



Perl TL developer

Let's Go!

LIVE NATION

XSLT is slow ?

yes :)

p.s. who cares?

DO WE HAVE "+" ?

standard

output as xhtml / xml / x*

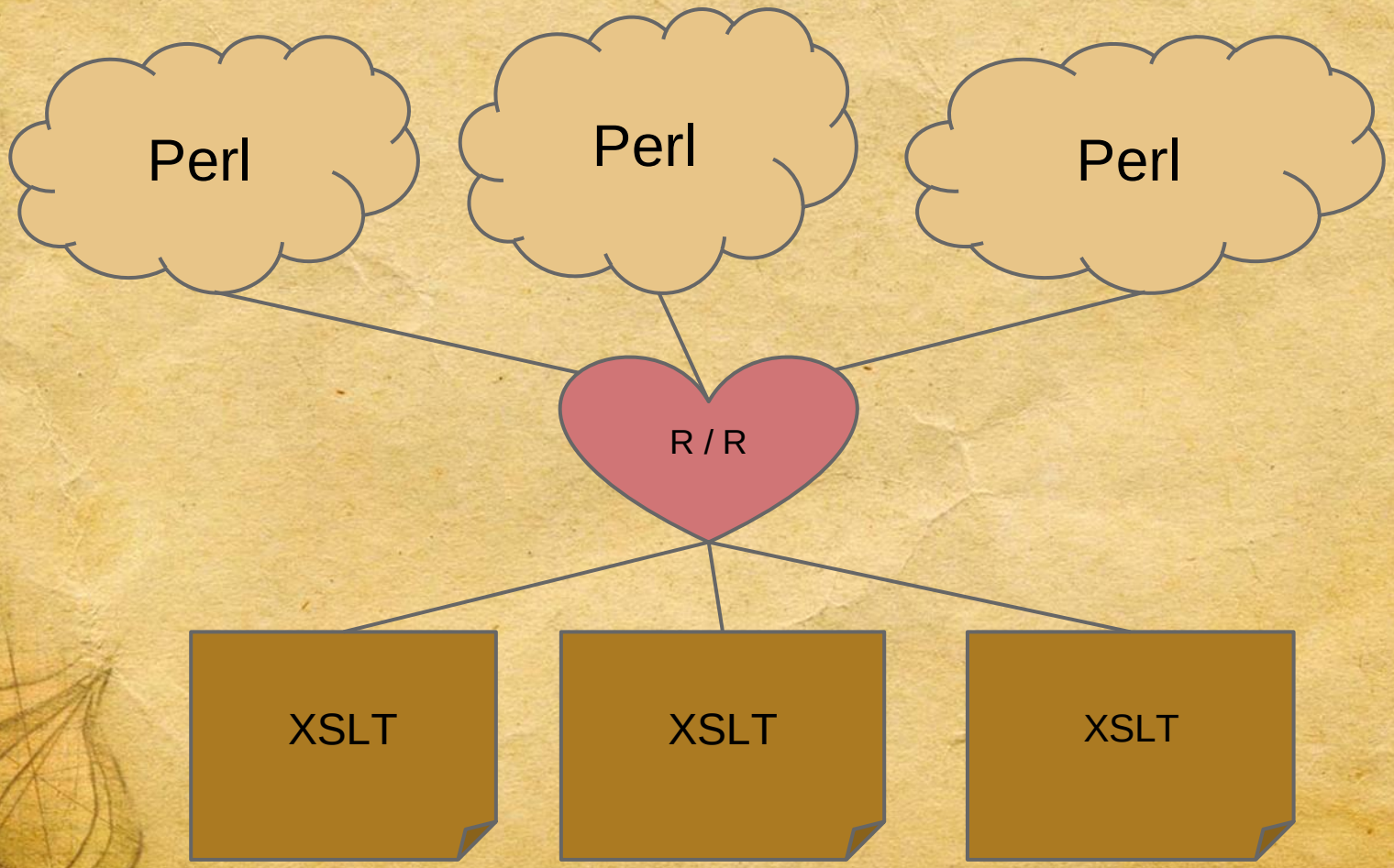
functional paradigm (apply-templates)

strict splitting code & templates

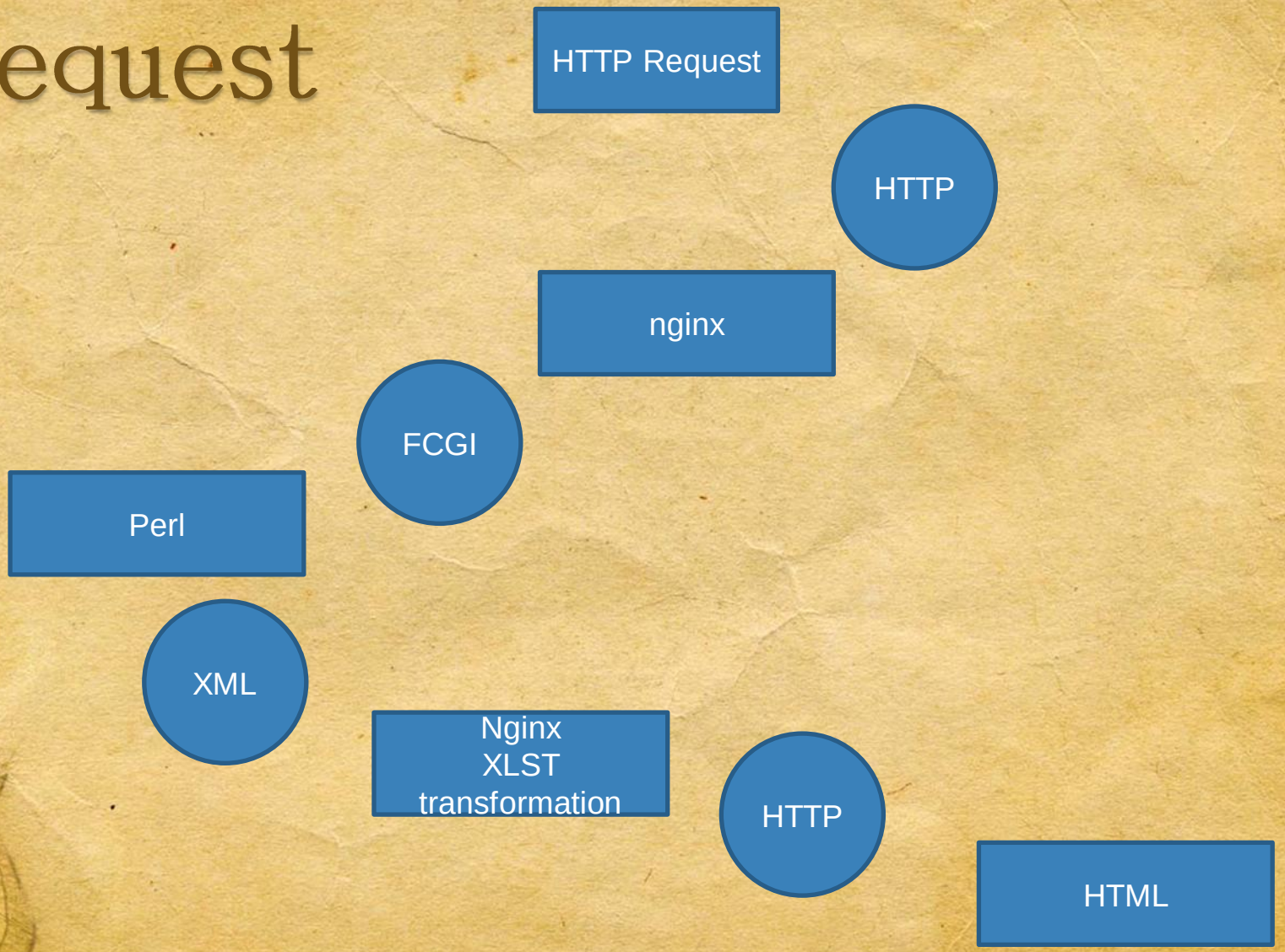
XSLT still slow

What we can do? :)

Web application structure

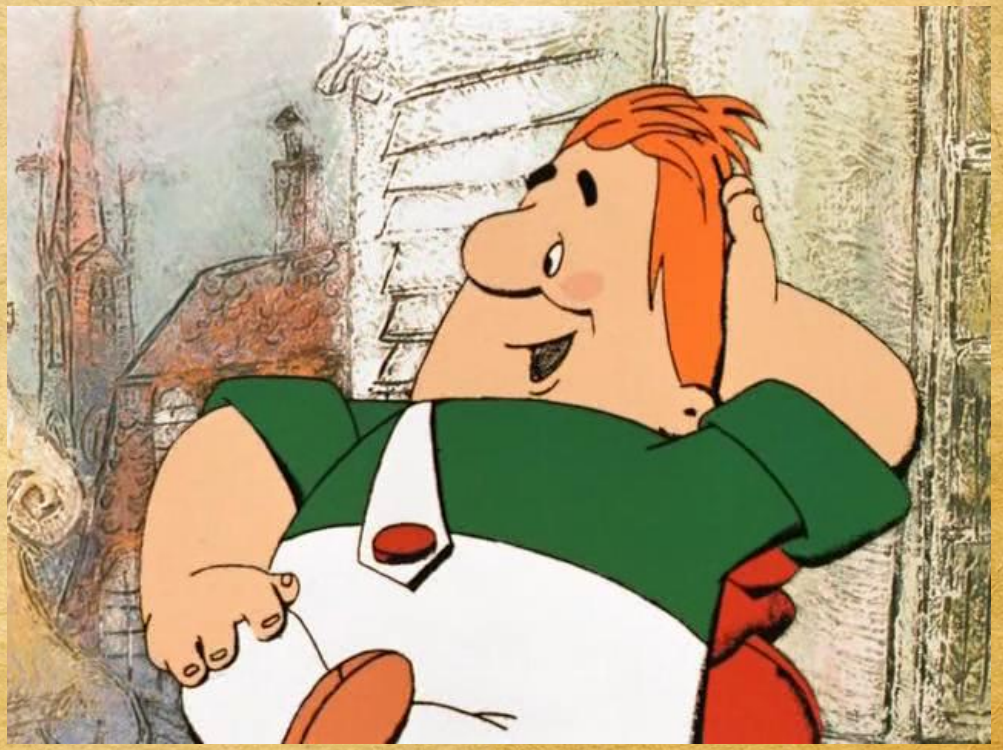


Request



I have a friend:

https://github.com/yoreek



XSLT module

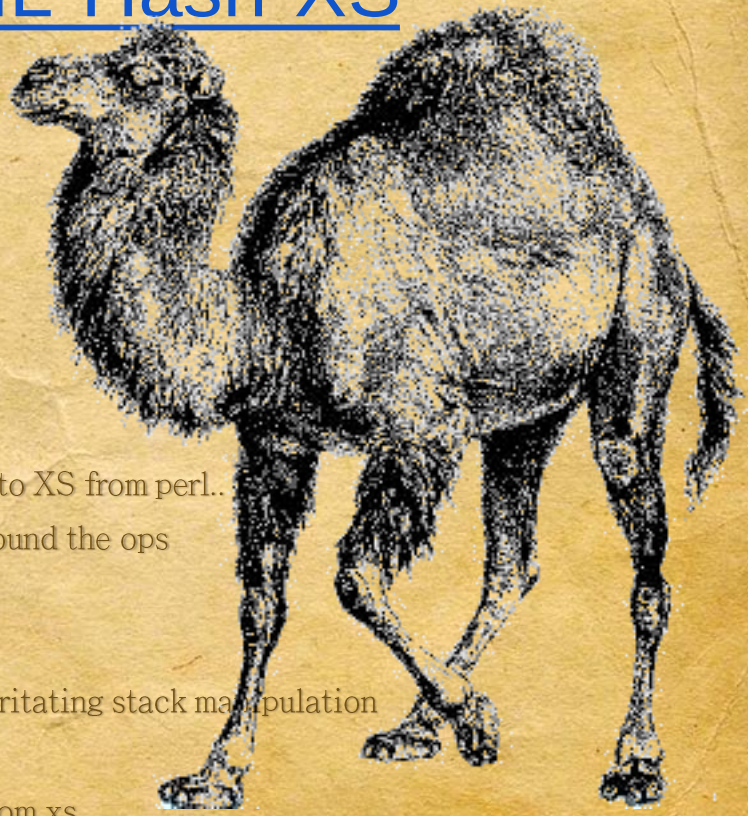
https://github.com/yoreek/nginx-xsltproc-module

X-Xslt-Stylesheet:
/xslt/test.xslt?param1=1&param2='test'

profiler / memcached

XML::Hash::XS

https://github.com/yoreek/XML-Hash-XS



PerlIO print

02:07 <@leont> Well, calling perl functions from XS is ugly too

02:08 < cono> yup

02:08 < cono> so we have not proper way to print to FH which was pass to XS from perl..

02:08 <@doy> i wonder why we don't just provide wrapper functions around the ops

02:08 < cono> y

02:09 < cono> doy: great idead

02:09 <@doy> so that they can be called from xs without having to do irritating stack manipulation

02:09 <@leont> We don't have that for a lot of things

02:09 <@leont> printing is easy, try to use regexps or smart-matching from xs

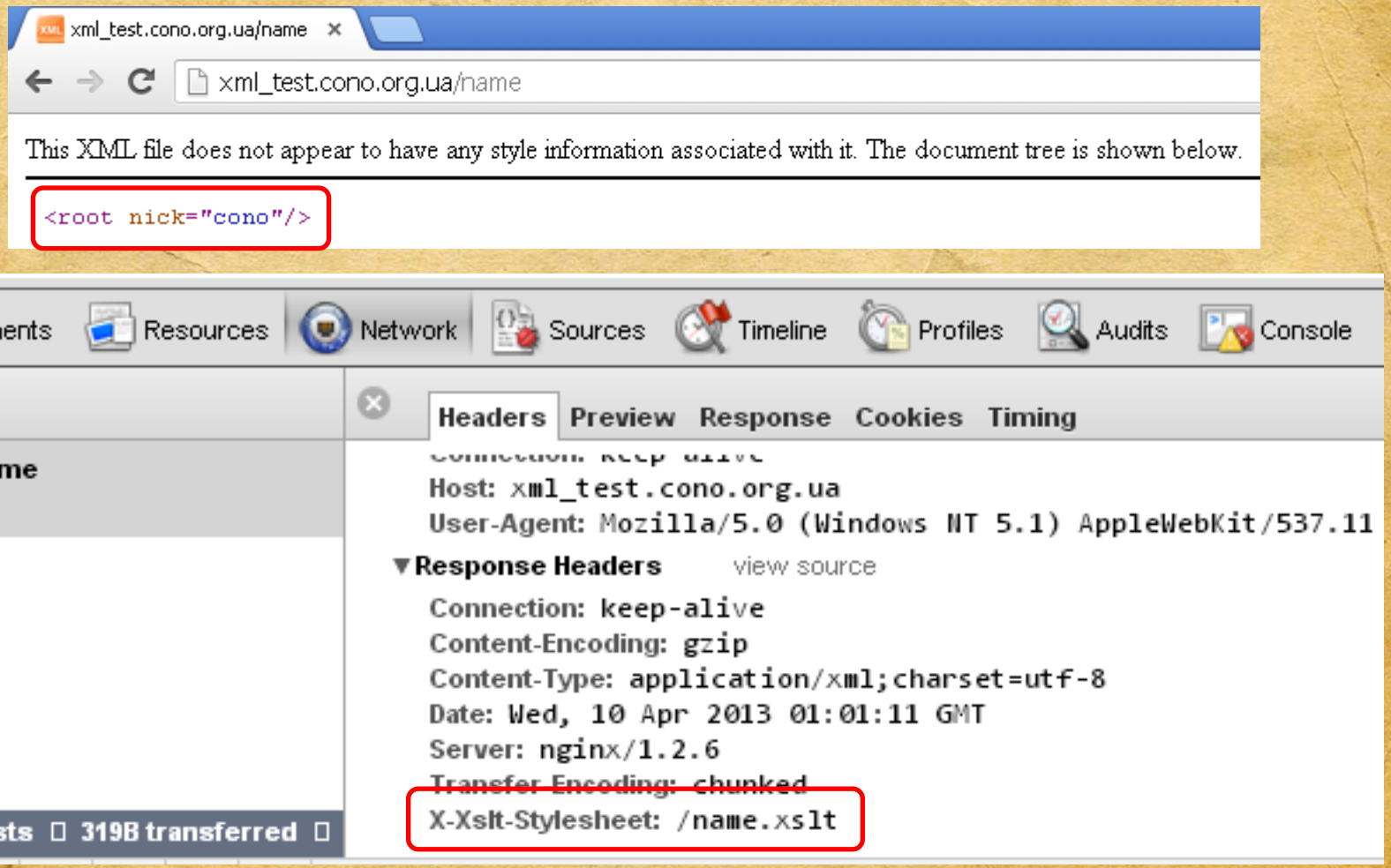
02:09 < cono> no thanks)

Examples

nginx configuration

```
server {
        listen 80;
        server_name xml_test.cono.org.ua;

        location / {
            try_files $uri @index;
        }

        location @index {
            fastcgi_pass    127.0.0.1:6561;
            include         fastcgi_params;
        }


        access_log /var/log/nginx/xml_test.cono.org.ua.access_log main;
        error_log /var/log/nginx/xml_test.cono.org.ua.error_log info;

        root /home/cono/www/cxml/htdocs;
}
```

Result of xml virtual host



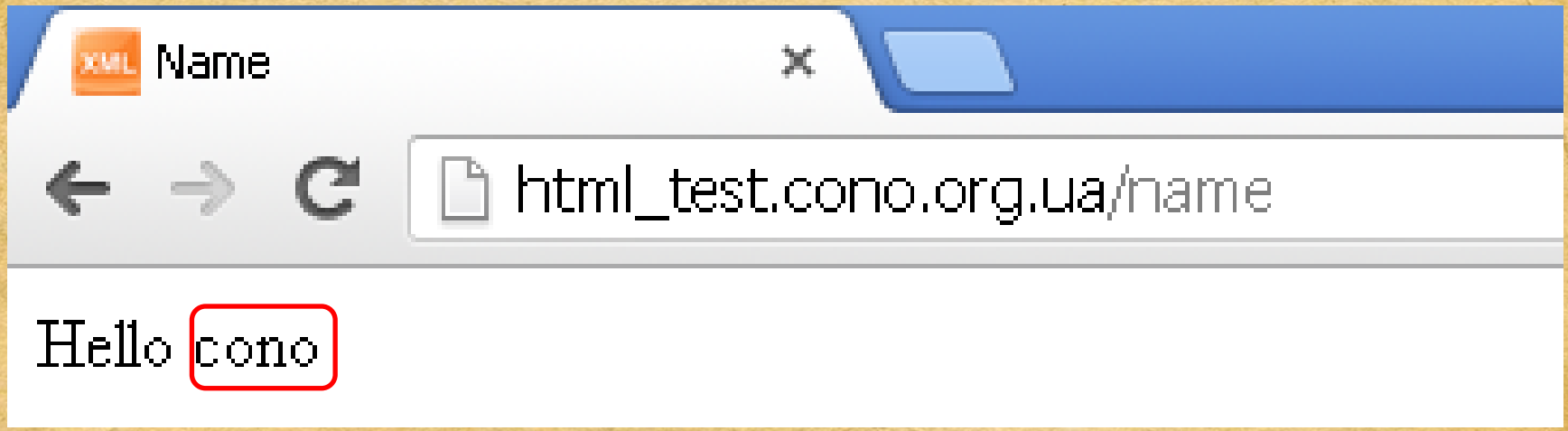
nginx html virtual host

```
server {
        listen 80;
        server_name html_test.cono.org.ua;

        location / {
            try_files $uri @index;
        }

        location @index {
            xsltproc        on;

            xsltproc_stylesheet_caching on;
            xsltproc_stylesheet_root /home/cono/www/cxml/tmpl;
            xsltproc_stylesheet_check_if_modify on;

            xsltproc_profiler on;
            xsltproc_profiler_stylesheet /home/cono/www/cxml/tmpl/profiler.xslt;
            xsltproc_profiler_repeat on;

            xsltproc_types  application/xml;

            fastcgi_pass    127.0.0.1:6561;
            include         fastcgi_params;
        }


        access_log /var/log/nginx/html_test.cono.org.ua.access_log main;
        error_log /var/log/nginx/html_test.cono.org.ua.error_log info;

        root /home/cono/www/cxml/htdocs;
}
```

XSLT template for /name

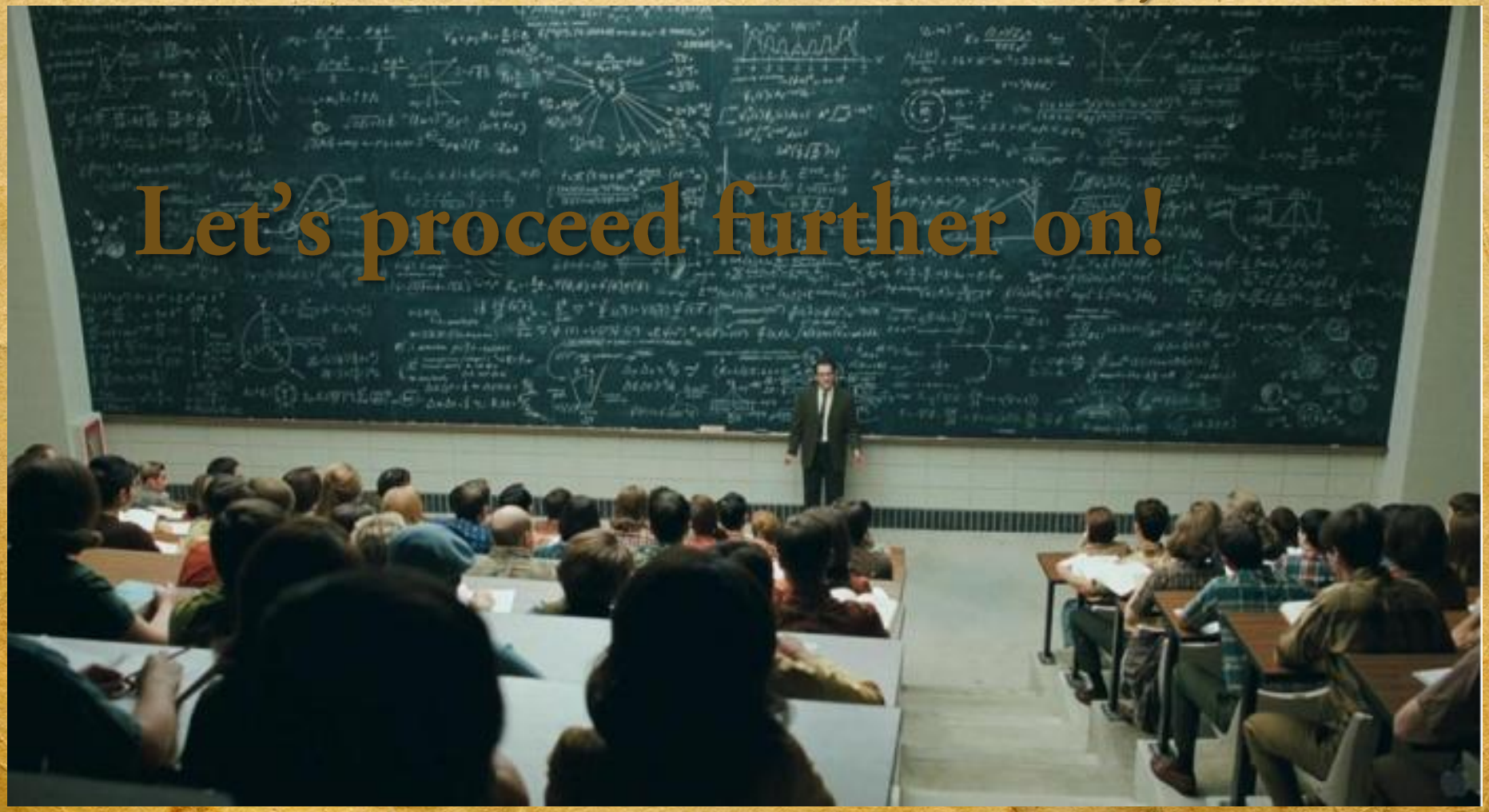
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/root">
<html>
    <head>
        <title>Name</title>
    </head>
    <body bgcolor="#fff">
        <p>Hello <xsl:value-of select="@nick" /></p>
    </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Result of html virtual host

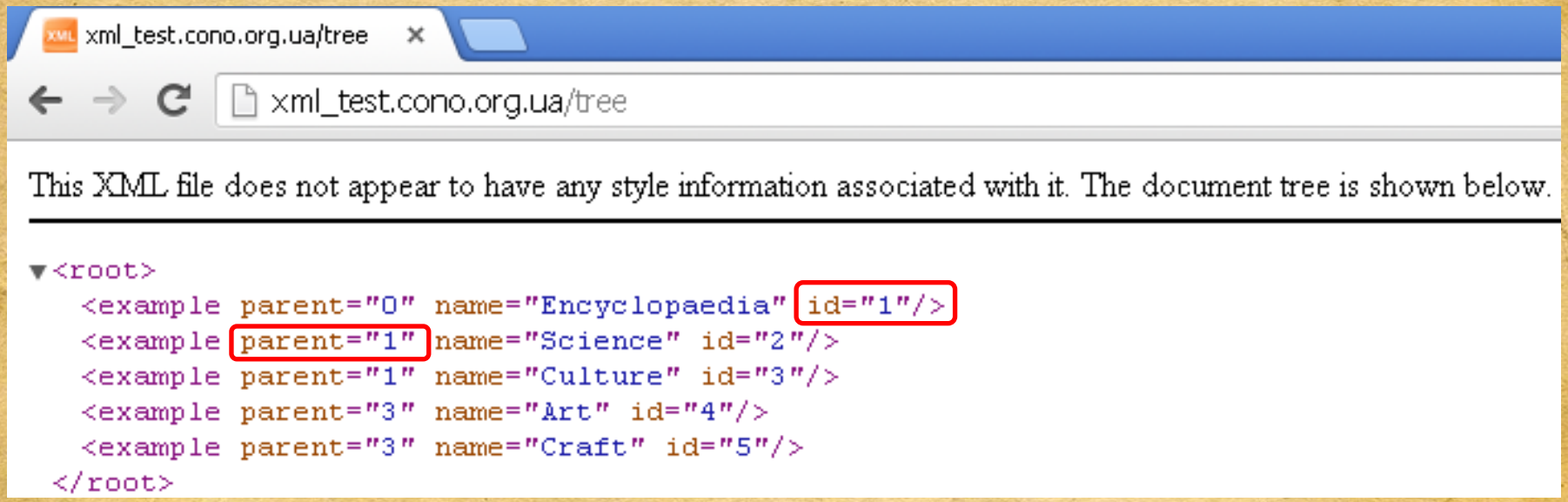


Pretty simple

Isn't it? ;)



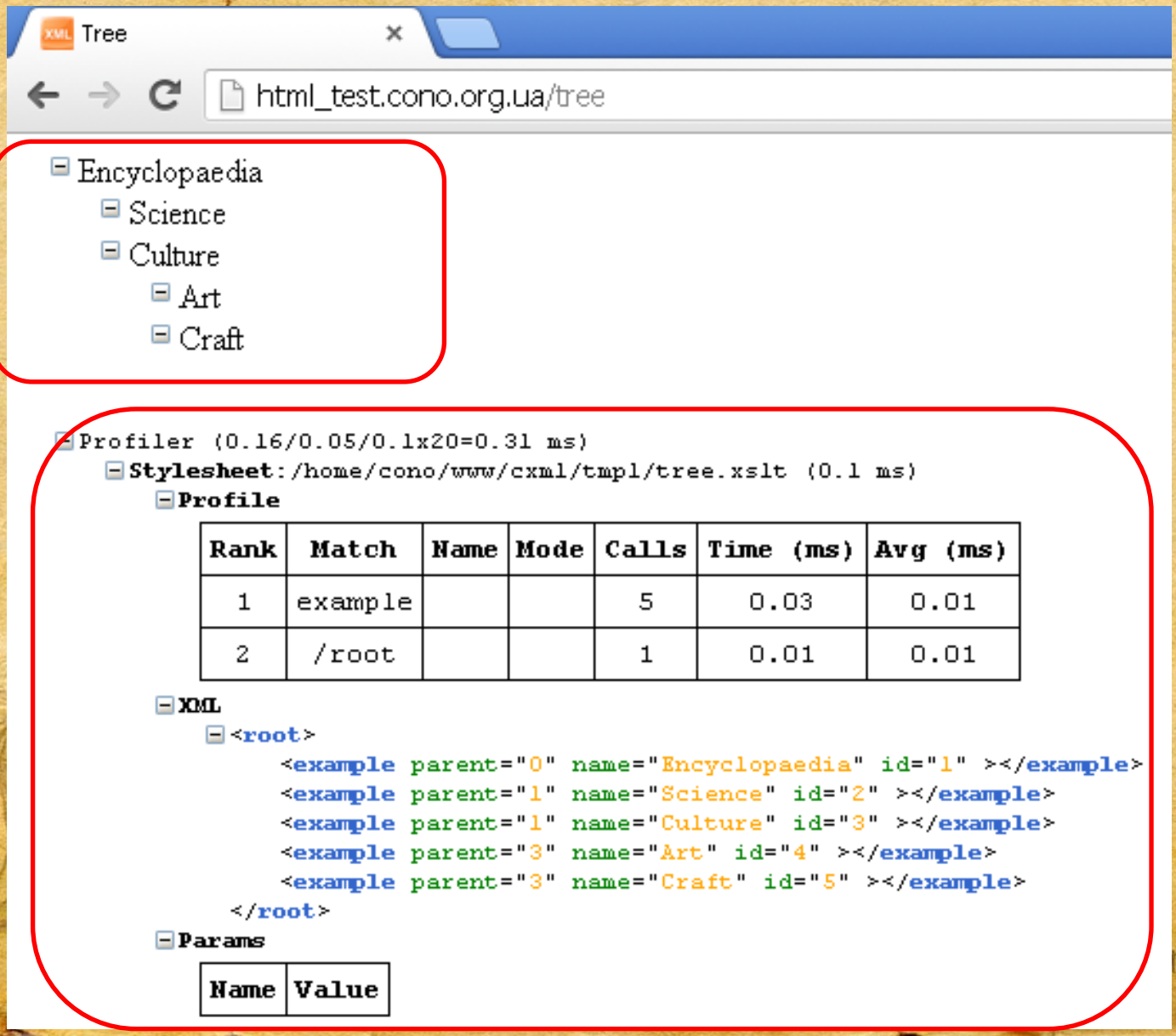
XML for /tree



XSLT for /tree

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="example">
    <li>
        <xsl:variable name="id" select="@id" />
        <label onclick="fold(this)"><xsl:value-of select="@name" /></label>
        <xsl:if test="../example[@parent=$id]">
            <ul><xsl:apply-templates select="../example[@parent=$id]" /></ul>
        </xsl:if>
    </li>
</xsl:template>

<xsl:template match="/root">
<html>
    <head>
        <title>Tree</title>
        <link rel="stylesheet" type="text/css" href="/c/index.css" />
        <script type="text/javascript" src="/j/index.js" />
    </head>
    <body bgcolor="#fff">
        <div class="treeroot">
            <ul>
                <xsl:apply-templates select="example[@parent=0]" />
            </ul>
        </div>
    </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

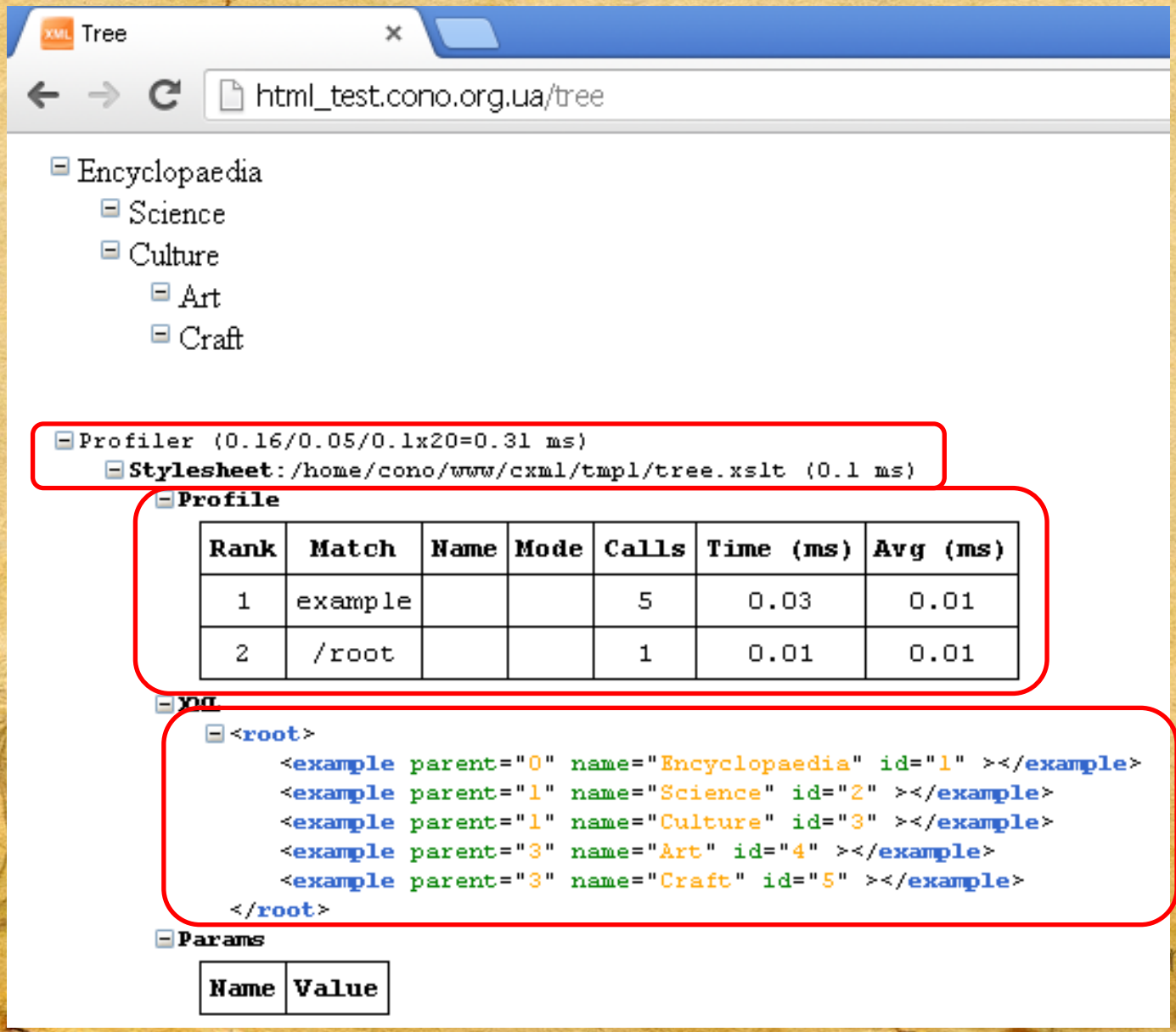
HTML for /tree



nginx html virtual host

```
server {
        listen 80;
        server_name html_test.cono.org.ua;

        location / {
            try_files $uri @index;
        }

        location @index {
            xsltproc        on;

            xsltproc_stylesheet_caching on;
            xsltproc_stylesheet_root /home/cono/www/cxml/tmpl;
            xsltproc_stylesheet_check_if_modify on;

            xsltproc_profiler on;
            xsltproc_profiler_stylesheet /home/cono/www/cxml/tmpl/profiler.xslt;
            xsltproc_profiler_repeat on;

            xsltproc_types  application/xml;

            fastcgi_pass    127.0.0.1:6561;
            include         fastcgi_params;
        }


        access_log /var/log/nginx/html_test.cono.org.ua.access_log main;
        error_log /var/log/nginx/html_test.cono.org.ua.error_log info;

        root /home/cono/www/cxml/htdocs;
}
```

HTML for /tree



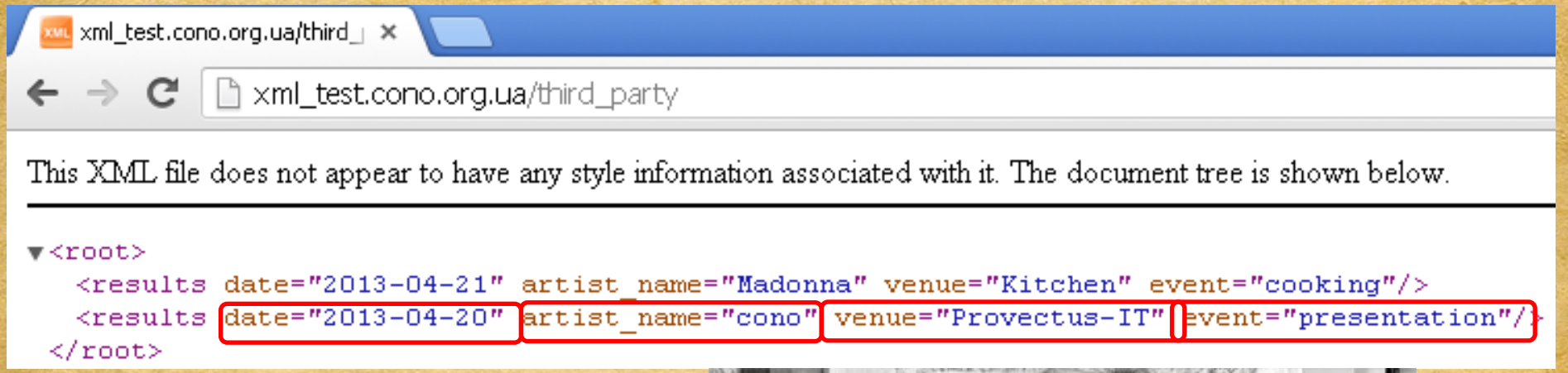
Any other reason to use XSLT?

Third parties!

nginx third party

```
server {
        listen 80;
        server_name third_party.cono.org.ua;

        location / {
            try_files $uri @index;
        }

        location @index {
            xsltproc        on;

            xsltproc_stylesheet_caching on;
            xsltproc_stylesheet_root /home/cono/www/cxml/tmpl;
            xsltproc_stylesheet_check_if_modify on;

            xsltproc_profiler on;
            xsltproc_profiler_stylesheet /home/cono/www/cxml/tmpl/profiler.xslt;
            xsltproc_profiler_repeat on;

            xsltproc_types  application/xml;


            add_header 'X-Xslt-Stylesheet' '/third_party_real.xslt';
            proxy_pass http://xml_test.cono.org.ua:80;
        }


        access_log /var/log/nginx/html_test.cono.org.ua.access_log main;
        error_log /var/log/nginx/html_test.cono.org.ua.error_log info;

        root /home/cono/www/cxml/htdocs;
}
```

XML for /third_party



XSLT for /third_party

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/root">
<html>
    <head>
        <title>Third Party</title>
    </head>
    <body bgcolor="#fff">
        <table border="1" cellpadding="1" cellspacing="1">
            <th>Event</th><th>Performer</th><th>Where</th><th>Date</th>
            <xsl:for-each select="results">
                <tr>
                    <td>
                        <xsl:value-of select="@event" />
                    </td>
                    <td>
                        <xsl:value-of select="@artist_name" />
                    </td>
                    <td>
                        <xsl:value-of select="@venue" />
                    </td>
                    <td>
                        <xsl:value-of select="@date" />
                    </td>
                </tr>
            </xsl:for-each>
        </table>
    </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

HTML for /third_party

Third Party

third_party.cono.org.ua/third_party

Event	Performer	Where	Date
cooking	Madonna	Kitchen	2013-04-21
presentation	cono	Provectus-IT	2013-04-20

Profiler (0.14/0.03/0.05x20=0.22 ms)

Stylesheet: /home/cono/www/cxml/tmpl/third_party_real.xslt (0.05 ms)

Profile

Rank	Match	Name	Mode	Calls	Time (ms)	Avg (ms)
1	/root			1	0.01	0.01

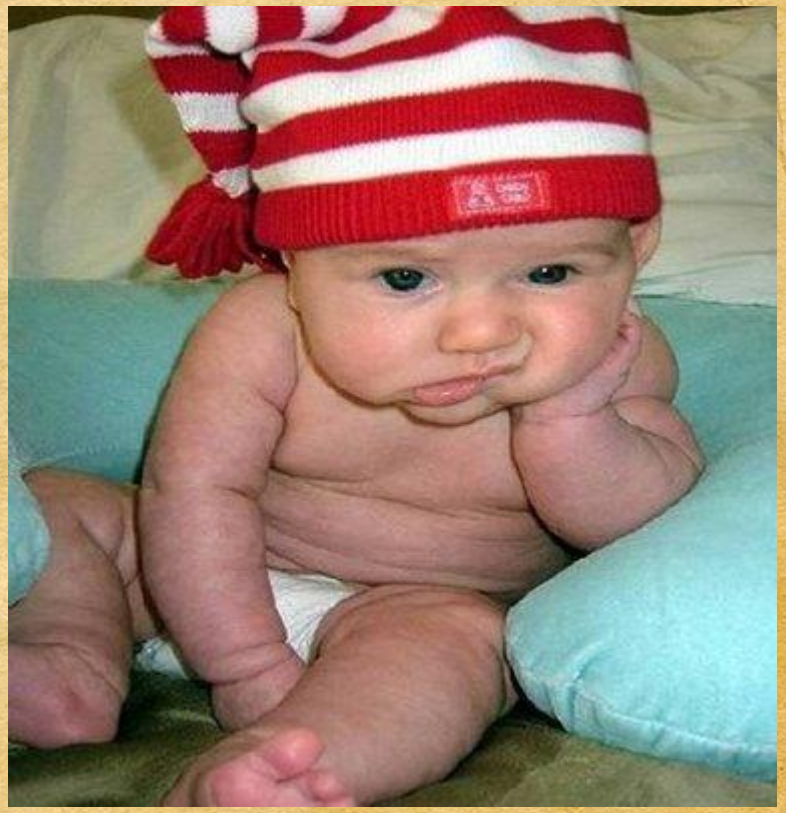
XML

```
<root>
    <results date="2013-04-21" artist_name="Madonna" venue="Kitchen" e
    <results date="2013-04-20" artist_name="cono" venue="Provectus-IT"
</root>
```

Params

Anything left?

XSD!



Test class

```
package CF::Test::XSD;

use strict;
use warnings;

use LWP::Simple;
use XML::LibXML;

use base 'Test::Builder::Module';

our @EXPORT = qw(xsd_ok);

my $CLASS = __PACKAGE__;


sub xsd_ok {
    my ($url, $xsd, $name) = @_;
    my $tb = $CLASS->builder;

    eval {
        my $xml = get($url);
        die "Could not fetch $url" unless defined $xml;

        my $parser = XML::LibXML->new(
            expand_entities => 0,
            load_ext_dtd    => 0,
            validation      => 0,
        );

        my $schema = XML::LibXML::Schema->new(
            location => $xsd
        );

        my $doc = $parser->parse_string($xml);
        $schema->validate($doc);
    };

    warn $@ if $@;

    $tb->ok(!$@, $name);
}
```

Test script

```
#! /usr/bin/perl

use strict;
use warnings;

use Test::More tests => 2;
use File::Spec;
use FindBin qw($Bin);
use lib File::Spec->catdir($Bin, qw|.. lib|);

use CF::Test::XSD;

my $host = 'http://xml_test.cono.org.ua/';

for my $uri ( qw/name tree/ ) {
    xsd_ok("$host$uri", File::Spec->catfile($Bin, 'xsd', "$uri.xsd"), "Check /$uri");
}
```

XSD for /name

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <!-- root element -->
    <xsd:element name="root" type="root-type" />

    <!-- root element type -->
    <xsd:complexType name="root-type">
        <xsd:attribute name="nick" type="xsd:string" use="required" />
    </xsd:complexType>
</xsd:schema>
```

Test successful

```
[Tue 13/04/09 23:48 UTC][pts/12][x86_64/linux-gnu/3.8.2-gentoo][5.0.2]
<cono@betacoda(4):~/www/cxml>
zsh/2 335  (git)-[master]-% prove t
t/00-xsd.t .. ok
All tests successful.
Files=1, Tests=2,  0 wallclock secs ( 0.02 usr  0.00 sys +  0.08 cusr  0.03 csys =  0.13 CPU)
Result: PASS
```

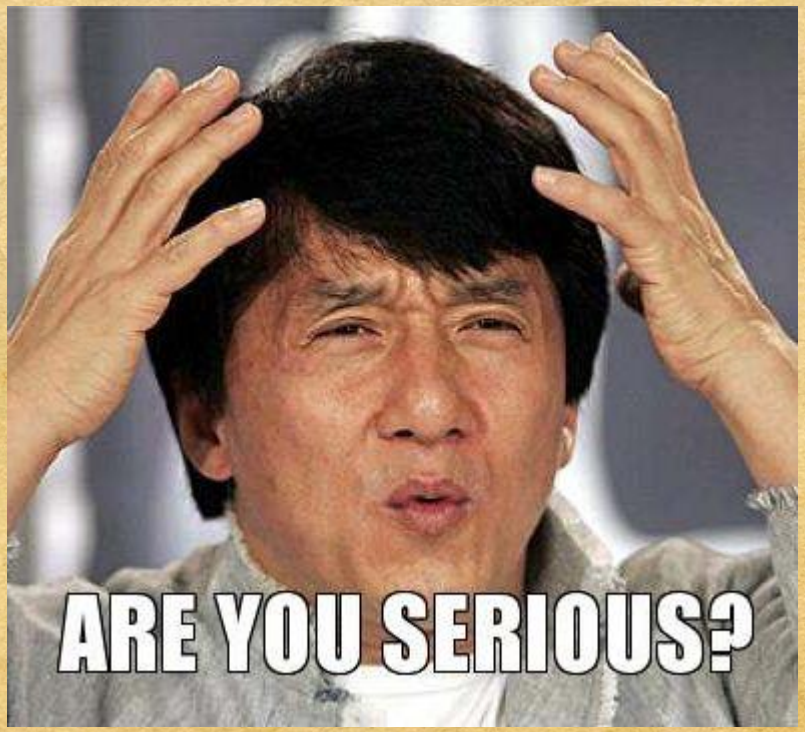
XSD for /tree w/ mistake

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <!-- root element -->
    <xsd:element name="root" type="root-type" />

    <!-- root element type -->
    <xsd:complexType name="root-type">
        <xsd:sequence>
            <xsd:element name="example" type="example-type" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>

    <!-- example element type -->
    <xsd:complexType name="example-type">
        <xsd:attribute name="parent" type="xsd:unsignedInt" use="required" />
        <xsd:attribute name="name"   type="xsd:int"         use="required" />
        <xsd:attribute name="id"     type="xsd:unsignedInt" use="required" />
    </xsd:complexType>
</xsd:schema>
```

Test fail

```
[Tue 13/04/09 23:49 UTC][pts/12][x86_64/linux-gnu/3.8.2-gentoo][5.0.2]
<cono@betacoda(4):~/www/cxml>
zsh/2 336 _(git)-[master]-% prove t
t/00-xsd.t .. 1/2 unknown-2613d60:0: Schemas validity error : Element 'example', attribute 'name': 'Encyclopaedia' is not a valid value of the atomic type 'xs:int'.
unknown-2613d60:0: Schemas validity error : Element 'example', attribute 'name': 'Science' is not a valid value of the atomic type 'xs:int'.
unknown-2613d60:0: Schemas validity error : Element 'example', attribute 'name': 'Culture' is not a valid value of the atomic type 'xs:int'.
unknown-2613d60:0: Schemas validity error : Element 'example', attribute 'name': 'Art' is not a valid value of the atomic type 'xs:int'.
unknown-2613d60:0: Schemas validity error : Element 'example', attribute 'name': 'Craft' is not a valid value of the atomic type 'xs:int'.

#   Failed test 'Check /tree'
#   at t/00-xsd.t line 16.
# Looks like you failed 1 test of 2.
t/00-xsd.t .. Dubious, test returned 1 (wstat 256, 0x100)
Failed 1/2 subtests

Test Summary Report
-------------------
t/00-xsd.t (Wstat: 256 Tests: 2 Failed: 1)
  Failed test:  2
  Non-zero exit status: 1
Files=1, Tests=2,  0 wallclock secs ( 0.01 usr  0.01 sys +  0.08 cusr  0.02 csys =  0.12 CPU)
Result: FAIL
```

Almost the end!



Joking! ☺

Thanks

bb :)

https://github.com/cono/xslt-template-engine