# XMLTV Project

Milton Keynes Perl Mongers
October 2010

Nick Morrott

## Part 1

The view from 30,000 feet

## What is XMLTV?

The XMLTV Project - a collection of Perl modules, grabbers and utilities to obtain, manipulate and search TV listings;

XMLTV.pm - creates XMLTV TV listings;

xmltv.dtd - an XML format describing TV listings;

## Project History

Initial release in 2000

Moved to sourceforge.net in 2001

Current release 0.5.58 (as of 09/2010)

## Project Structure

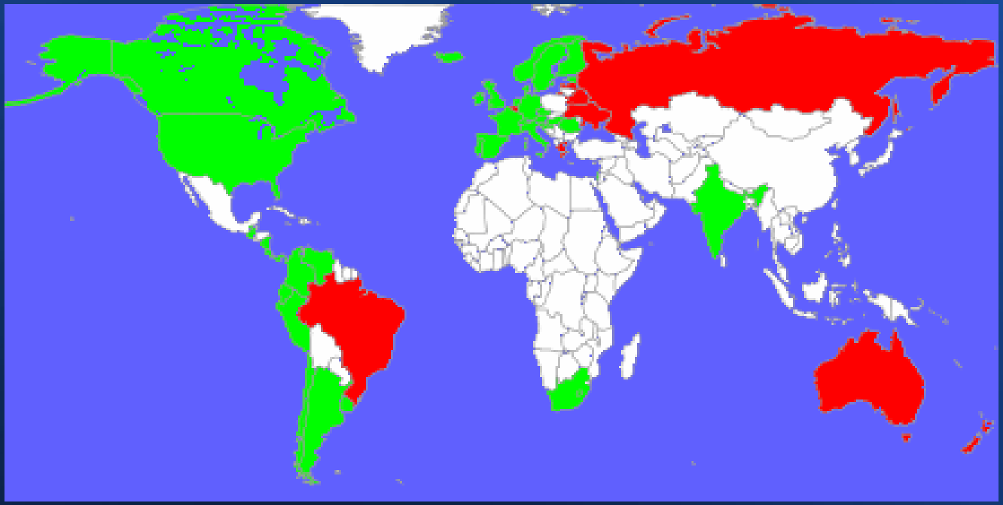
Few developers handling core modules/releases

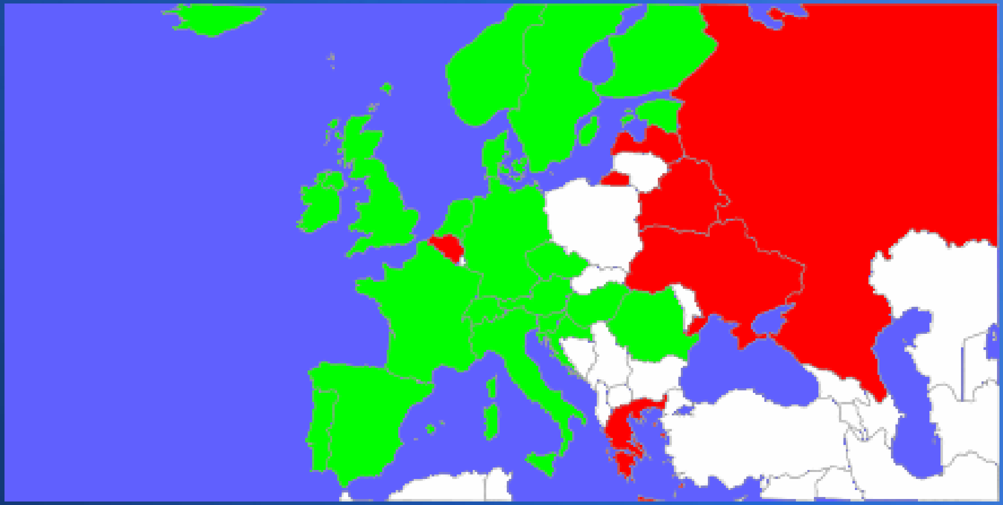
25 grabbers serving 20+ countries maintained independently

Released under GPL v2

Releases made 2-3 times/year

## Global Coverage


World


Europe

Note that unsupported 3rd-party grabbers may provide listings for those countries with no official XMLTV grabber coverage

(maps from chart.apis.google.com)

## Personal Involvement

- Started contributing in 2005

- Maintaining Radio Times grabber since 2007

- Rewrote French grabber after source site updated

- Working on “lineups” feature

## Who uses XMLTV Data?

PVR applications (MythTV, Freevo...)

Listings viewers (FreeGuide, OnTV...)

Scripts filtering data directly

## tv_grab_uk_rt (Radio Times)

Richest source of data for UK users

Uses Radio Times XMLTV data service

Listings for >400 channels

Location-aware setup (postcode/TV service)

Significant “data cleansing” to improve listings

## tv_grab_uk_rt - advantages

Radio Times provides 2 weeks of listings

Consistent and rich data

Sky/Virgin pay channels only supported via XMLTV

Free as in beer for home use

## tv_grab_uk_rt - disadvantages

No radio channels...

Data generated daily

Can be cumbersome to configure

New channels $\rightarrow$ reconfigure XMLTV

## Alternatives?

i) EIT (“over-the-air”) listings:

- supported in several PVR apps
- broadcast on Freeview and Freesat
- frequent updates and easy to configure

ii) Digiguide ($$$) / BBC Backstage (BBC only)

## Building XMLTV

XMLTV binaries available for most distros

but

Typical build process from cvs/tarball:

```
$ perl Makefile.PL PREFIX=/usr/local/
$ make
$ make test
# make install
```

## Configuring a grabber

Select desired channels:

```
$ tv_grab_uk_rt –configure (defaults to ~/.xmltv/)
```

Grab the data (daily via cron):

```
$ tv_grab_uk_rt --output listings.xml
```

## apiconfig – XML-based config

Supported by some grabbers

Stage-based configuration using XML

Allows for easier configuration via GUI

Not really implemented in end-user apps though...

# XMLTV Utilities

tv_grab_combiner
– run multiple grabbers and combine listings

tv_grep
– extract programmes/channels from an XMLTV file

tv_cat
– concatenate several XMLTV files together

tv_find_grabbers
- find all installed XMLTV grabbers (core and 3rd party)

(and tv_sort / tv_split / tv_imdb / tv_to_latex...)

## Part 2

The Internals

## Sources of Listings Data

- Pre-formatted XMLTV data (tv_grab_sw_swedb)
- Machine-readable data (tv_grab_uk_rt)
- Screen-scraping listings site (tv_grab_fr)
- EIT broadcast data (tv_grab_it_dvb, via Linux::DVB)

## XMLTV DTD

- Developed by XMLTV Project, also used by 3rd party applications

- Alternative to TV-Anywhere format

- Simple: <channel> and <programme> elements, sub-elements cover attributes

- Internally validated by XMLTV.pm

## XMLTV Data Structure

List of four elements:

i) character encoding used (string)

ii) attributes of the root <tv> element (hash)

iii) <channel> elements (hash)

iv) <programme> elements (list)

## XMLTV Data Structure (2)

Internal data structure will be something like:

```
[ 'UTF-8',

  { 'source-info-name' => 'Ananova', 'generator-info-name' => 'XMLTV' },

  { 'radio-4.bbc.co.uk' => { 'display-name' => [[ 'en',  'BBC Radio 4' ],
                                                [ 'en',  'Radio 4'     ],
                                                [ undef, '4'           ]],
                             'id' => 'radio-4.bbc.co.uk' },
  ... },

  [ { start => '200111121800', title => [ [ 'Simpsons', 'en' ] ],
      channel => 'radio-4.bbc.co.uk' },
  ... ]
]
```

## Grabber Capabilities

$ tv_grab_uk_rt –capabilities

- baseline (quiet, output, days, offset)
- manualconfig
- tkconfig
- apiconfig
- cache
- preferredmethod
- lineups (a work in progress...)

## Grabber Internals - Overview

Grabbers must allow for configuration, listing channels and grabbing data

Encouraged to use ParseOptions() from XMLTV::Options to simplify development

ParseOptions() provides direct access to runtime options and grabber configuration

## ParseOptions()

Implements all required functionality except configuration, listing channels and grabbing data

```
my( $opt, $conf ) = ParseOptions( {
    grabber_name => "tv_grab_uk_rt",
    capabilities => [qw/baseline manualconfig apiconfig/],
    stage_sub => \&config_stage,
    listchannels_sub => \&list_channels,
    version => 'v 1.301 2010/10/10 17:38:45',
    description => "Radio Times (UK)",
} );
```

## Grabber Internals - Skeleton

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

=pod

Your documentation here...

=cut

use strict;
use XMLTV::Options qw/ParseOptions/;

my( $opt, $conf ) = ParseOptions( {...} );

# Get the actual data and print it to stdout.

if( $is_success ) {
    exit 0;
}
else {
    exit 1;
}

sub config_stage {...}

sub list_channels {...}
```

## XMLTV.pm

Cornerstone of the project

Handles all XMLTV data I/O

Uses specific handlers to validate content

Handlers include with-lang, episode-num, video, audio, rating, credits, scalar, length, icon

# Reading XMLTV data

```
use XMLTV;

my $data = XMLTV::parsefile('tv.xml');
my ($encoding, $credits, $ch, $progs) = @$data;
```

# Writing XMLTV data

```
use XMLTV;

my $w = new XMLTV::Writer(encoding => 'UTF-8');
$w->comment("Hello");
$w->start({ ’generator-info-name’ => ’test-gen’ });

# write a single channel
my %ch = (id => ’test-channel’,
          ’display-name’ => [ [ ’Test’, ’en’ ] ]);
$w->write_channel(\%ch);

# write a single programme
my %prog = (channel => ’test-channel’,
            start => ’200203161500’,
            title => [ [ ’News’, ’en’ ] ]);
$w->write_programme(\%prog);

$w->end();
```

## Useful XMLTV modules

XMLTV::Supplement
- retrieve files such as channel lists from XMLTV server

XMLTV::DST
- handling for daylight savings timings

XMLTV::Get_nice
- inject random delays in successive HTTP retrievals

## Useful core/3rd party modules

Encode
POSIX

LWP::UserAgent (and other LWP modules)
HTML::Entities
HTML::TreeBuilder
HTTP::Cache::Transparent
Date::Manip

## HTML::TreeBuilder

```
use HTML::TreeBuilder;
use XMLTV::Get_nice qw(get_nice);

my $content = get_nice($url);
$content = decode_utf8($content);
my $tree = new HTML::TreeBuilder;
$tree->parse($content);
$tree->eof;

foreach my $cell ( $tree->look_down( "_tag", "td",
                                     "class", "channel" ) )
{
    my $img = $cell->look_down( "_tag", "img" );
    my $chname = trim( $img->attr('alt') );
    ...
}

$tree->delete(); undef $tree;
```

## Date::Manip

```
my $strDate = ParseDate( “20100301120000 +0000” );
my $strDelta = ParseDateDelta( “5minutes” );

my $date = DateCalc( $strDate, $strDelta );

my $unixDate = UnixDate( $date, “%Y%m%d%H%M %z” );

if ( Date_Cmp( $dateStart, $dateStop ) < 0 ) {
    print “Start date is earlier than stop date!”;
}
```

## Useful Links

Homepage
http://www.xmltv.org

Code (CVS/tarball)
http://sourceforge.net/projects/xmltv/

Mailing list
http://lists.sourceforge.net/lists/listinfo/xmltv-users

## Thanks for listening!

Questions?

Nick Morrott
knowledgejunkie at gmail dot com