Embperl - How to Build Large Scale Websites/Webapplications With Perl

ApacheCon 2002

Gerald Richter

ecos gmbh

http://www.ecos.de

The Embperl Website

Documentation are written in POD, Output should be HTML and PDF

Database for storing new, articles, sites useing Embperl and other Links.

Inclusion of various document formats (HTML, binary etc.)

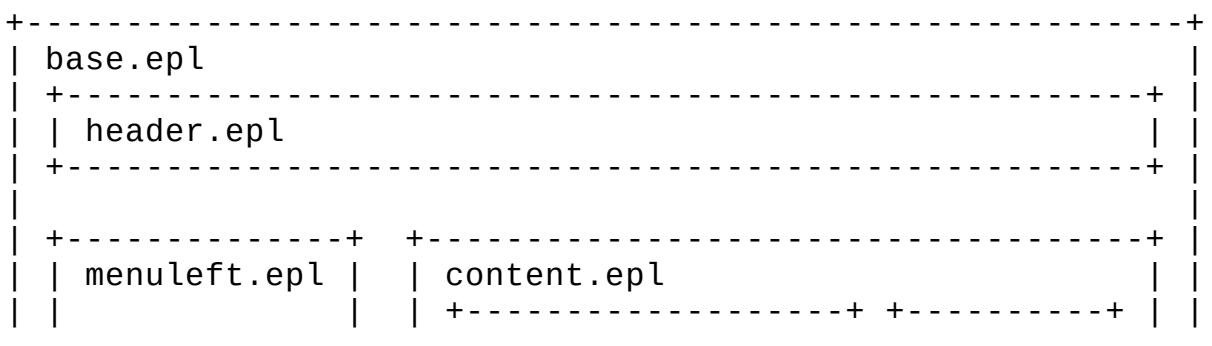
Multilanguage (german, english)

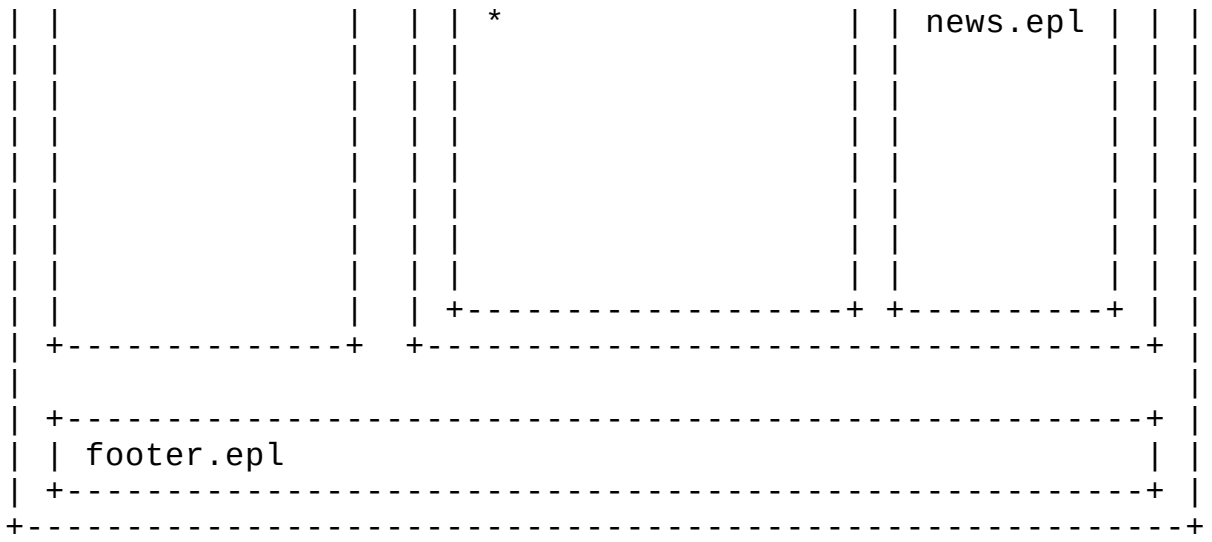
The Layout

The starting page...

http://perl.apache.org/embperl/ or http://www.ecos.de/embperl/

The components of the page...





base.epl

```
[-
$r = shift ;
$http_headers_out{'content-type'} = 'text/html' ;
-]
<html>
    <head>
        <title>Embperl</title>
        <style type="text/css">
           body                {font-family: Geneva,Arial,H
           table               {font-family: Geneva,Arial,H
        </style>
    </head>
    <body bgcolor="#ffffff">
        [- Execute ('header.epl') -]
        <table width="100%" cellspacing="0" cellpadding="0"
            <tr>
                <td>[- Execute ('menuleft.epl') -]</td>
                <td width="2"> </td>
                <td height="100%"><img src="/eg/images/frame
                <td width="10"> </td>
                <td width="90%">[- Execute ('content.epl') -
                <td width="2"> </td>
                <td height="100%"><img src="/eg/images/frame
                <td width="10"> </td>
            </tr>
        </table>
        [- Execute ('footer.htm') -]
    </body>
</html>
```

content.epl

```
<table width="100%" cellspacing="0" cellpadding="0" border="
    <tr>
        <td valign="top">
```

```
            <br><br>
            <font size="2" face="Verdana, Arial, Helvetica, sans
                [- Execute ({'*') -]</td>
            <td width="10"> </td>
            <td height="100%"><img src="/eg/images/frame.jpg" wi
            <td width="2"> </td>
            <td align="left" width="152"><font size="2" face="Ve
                [- Execute ('news.epl') -]
            </td>
        </tr>
    </table>
```

Embperl's objects

The request-object

Makes data about the request avaiable, like URI, HTTP-header, form data

The component-object

Makes data about the component available, like filename, syntax, recipe

The application-object

Brings together the data of a set of pages that forms an application. Like session handling, logging and configuration.

Embperl::Object

Embperl::Object manages the calling and overriding of components

1. **) Createing of the request-object and populating it with informations about the request**
2. **) Loading of the base template**
   Starting at the directory that contains the file that is requested, Embperl::Object searches the directory hierachie up to the document root (or EMBPERL_OBJECT_STOPDIR) for the base template. All dierectories of this search are now part of the search path for loading all file in this request. That is not only true for other Embperl components, but also for other files like XSL sytlesheets.

3. **) Application-object searching and loading**
4. **) Setting of the inherence**

```
Application-File
    |
    v
Embperl::App
```

5. **) calling the init method of the application object**
   This allows to execute application specific code (like database access) and modify the request. In a MCV modell this is the controller.

6. **) Loading of the actual requested page and blessing of the request object into the package of this page**
7. **) Setting the inherence of the request-object**

```
Requested page
    |
    v
Base template
    |
    v
Embperl::Req
```

8. **) Executing the base template**

The application-object of the Embperl Website

Parts of the base application

```
sub init
    {
    my $self     = shift ;
    my $r        = shift ;

    my $config = Execute ({object => 'config.pl', syntax =>

    $config -> new ($r) ;

    $r -> {config} = $config  ;
    $r -> {menu}   = $config -> get_menu ($r) ;
    fill_menu ($config, $r -> {menu}, $r -> {baseuri}, $r ->
    $pf = map_file ($r) ;
    $r -> param -> filename ($pf) ;

    Execute ({inputfile => 'messages.pl', syntax => 'Perl'})

    return 0 ;
    }
```

The configuration (config.pl)

```
BEGIN
    {
    %messages = (
        'de' =>
            {
            'Introduction'  => 'Einführung',
            'Documentation' => 'Dokumentation',
            'Examples'      => 'Beispiele',
            'Changes'       => 'Änderungen',
            'Sites using Embperl' => 'Websites mit Embperl',
            'Add info about Embperl' => 'Hinzufügen über Emb
            }
        ) ;

    @menu = (
        { menu => 'Home',                   uri => '',

        { menu => 'Features',               uri => 'pod/Feat
        { menu => 'Introduction',           uri => 'pod/intr
            [
            { menu => 'Embperl',            uri => 'Intro.ht

            { menu => 'Embperl::Object',    uri => 'IntroEmb
            ]
        },
        { menu => 'Documentation',          uri => 'pod/doc/
            [
            { menu => 'Embperl',            uri => 'Embperl.

            { menu => 'Embperl::Object',    uri => 'EmbperlO
            { menu => 'Embperl::Syntax',    uri => 'EmbperlS
                [
                { menu => 'Embperl',        uri => 'Embperl.
                { menu => 'EmbperlBlocks',  uri => 'EmbperlB
                { menu => 'EmbperlHTML',    uri => 'EmbperlH
                { menu => 'HTML',           uri => 'HTML.htm
                { menu => 'ASP',            uri => 'ASP.htm'
                { menu => 'SSI',            uri => 'SSI.htm'
                { menu => 'Perl',           uri => 'Perl.htm
                { menu => 'POD',            uri => 'POD.htm'
                { menu => 'Text',           uri => 'Text.htm
                { menu => 'RTF',            uri => 'RTF.htm'
                { menu => 'Mail',           uri => 'Mail.htm
                ],
            },
            { menu => 'Embperl::Recipe',    uri => 'EmbperlR
                [
                { menu => 'Embperl',        uri => 'Embperl.
                { menu => 'EmbperlXSLT',    uri => 'EmbperlX
                { menu => 'XSLT',           uri => 'XSLT.htm
```

```
                ],
            },
            ],
        },
        { menu => 'Installation',           uri => 'pod/INST
        { menu => 'FAQ',                    uri => 'pod/Faq.
        { menu => 'Tips & Tricks',          uri => 'pod/Tips
        { menu => 'Examples',               uri => 'examples
        { menu => 'Changes',                uri => 'pod/Chan
        { menu => 'Sites using Embperl',    uri => 'pod/Site
        { menu => 'News',                   uri => 'db/news/

        { menu => 'Sites using Embperl',    uri => 'db/sites

        { menu => 'Add info about Embperl',             uri
            [
            { menu => 'Select category',        uri => 'db/a
            { menu => 'Review added info',      uri => 'db/s
            { menu => 'Show info',              uri => 'db/d
            ],
        },
        ) ;

    } ;

sub new
    {
    my ($self, $r) = @_ ;

    # The following two values must be changed to meet your
    # Additionaly DBI and DBIx::Recordset must be installed

    $self -> {dbdsn}      = $^O eq 'MSWin32'?'dbi:ODBC:embpe
    $self -> {dbuser}     = 'www' ;
    $self -> {dbpassword} = undef ;
    }

sub get_menu
    {
    my ($self, $r) = @_ ;

    push @{$r -> messages}, $messages{$r -> param -> languag

    return \@menu ;
    }
```

The navigation

```
[$ sub menuitem $]
[*
```

```
my ($url, $txt, $state, $tablebg, $ndx) = @_ ;
*]

  <tr>
    <td [$if $tablebg $]background="[+ $r -> {imageuri} +]/h
      <table width="152" border="0" cellspacing="0" cellpadd
        <tr>
          <td nowrap align=left width=[+ $ndx * 15 + 2 +]>
                <img = src="[+ $r -> {imageuri} +]/transp.gi
          <td nowrap align=left width=15>
                <img src="[+ $r -> {imageuri} +]/i-sub-[+ $s
                            +].gif" width="11" height="11" v
          <td nowrap align=left>
                <a href="[+ $url +]"><b>[+ $r -> gettext ($t
        </tr>
      </table>
    </td>
  </tr>

[$ endsub $]

[$ sub menu $]

[*
my ($menu, $ndx, $top) = @_ ;
*]

    [$ foreach my $item (@{$menu}) $]
        [-
        if ( $r -> {menuitems}[$ndx] eq $item)
            {
            menuitem ($item -> {url}, $item -> {menu}, $r ->
            menu ($item -> {sub}, $ndx + 1, 0) if ($item ->
            }
        else
            {
            menuitem ($item -> {url}, $item -> {menu}, 0, $t
            }
        -]
        <tr>
            <td><img src="/eg/images/linie-nav.gif" width="1
        </tr>
    [$endforeach $]

[$ endsub $]

[-
$r = shift ;
-]

<table width="152" border="0" cellspacing="0" cellpadding="0
  <tr>
    <td><img src="/eg/images/h_content.gif" width="152" heig
```

```
  </tr>
  <tr>
    <td> </td>
  </tr>
  [- menu ($r -> {menu} , 0, 1) -]
  <tr>
    <td> </td>
  </tr>
  <tr>
    <td><img src="/eg/images/h_current-v.gif" width="152" he
  </tr>
  <tr>
    <td>      </td>
  </tr>
  <tr>
    <td><img src="/eg/images/linie-nav.gif" width="152" heig
  </tr>
  <tr>
    <td bgcolor="#D2E9F5"><b>     S
      1.3.4</b></td>
  </tr>
  <tr>
    <td><img src="/eg/images/linie-nav.gif" width="152" heig
  </tr>
  <tr>
    <td><img src="/eg/images/transp.gif" width="152" height=
  </tr>
  <tr>
    <td><img src="/eg/images/linie-nav.gif" width="152" heig
  </tr>
  <tr>
    <td bgcolor="#D2E9F5"><b>     
        Beta 2.0b8</b></td>
  </tr>
  <tr>
    <td><img src="/eg/images/linie-nav.gif" width="152" heig
  </tr>
</table>
```

The news column of the home page

```
[-
use DBIx::Recordset ;

$r = shift ;

*set = DBIx::Recordset -> Search ({'!DataSource'    => $r ->
                                  '!Username'      => $r ->
                                  '!Password'      => $r ->
                                  '!DBIAttr'       => { Rai
                                                        Lon
```

```
                                        '!Table'         => 'item
                                        '!TabRelation'   => 'item
                                        '!Order'         => 'crea
                                        'language_id'    => $r ->
                                        'category_id'    => 1,
                                        '$max'           => 15})

-]

    <table width="252" border="0" cellspacing="0" cellpaddin
      <tr>
        <td><img src="[+ $r -> {imageuri} +]/h_news.gif" wid
      </tr>
      <tr>
        <td> </td>
      </tr>
      <tr>
        <td>

            [$while ($rec = $set -> Next)$]
                <table width="100%" border="0" cellspacing="
                    <tr>
                      <td bgcolor="#327EA7"><font size="1" f
                        <font color="#FFFFFF"> 
                            [+ $rec -> {heading} +]
                        </font></b></font></td>
                    </tr>
                    <tr>
                      <td bgcolor="#C2D9E5"><img src="[+ $r
                    </tr>
                    <tr>
                      <td bgcolor="#D2E9F5">
                        <table width="100%" border="0" cells
                          <tr>
                            <td><font size="1" face="Verdana
                                [-
                                $txt = $rec -> {description}
                                $txt =~ s#<#&lt;#g ;
                                $txt =~ s#>#&gt;#g ;
                                $txt =~ s#B&lt;(.*?)&gt;#<B>
                                $txt =~ s#(http://[-a-zA-Z.]
                                -]
                                [+ do { local $escmode = 0 ;
                            </font></td>
                          </tr>
                        </table>
                      </td>
                    </tr>
                  </table>
            [$endwhile $]

        </td>
      </tr>
    </table>
```

Syntaxes, Recipes and Provider

The execution of a component is divied in multiple steps. Every step is done by a separate provider.

Recipes defines in which order providers are executed.

This could be a simple linear structure or even a complex tree structure.

The Defaultrecipe: 'Embperl'

Read the source (File, Memory)
Parse
Compile
Execute
Output

The syntax tells the parser and compiler what input format they should expect

Examples: Embperl, ASP, SSI, Perl, Text, POD, RTF

You can create your own syntax by writing a new syntax class.

You can extented an existing syntax by inherenting from an existing class

Every intermediate step and the result can be cached

Rendering POD to HTML via XML and XSLT

The syntax POD transforms POD to XML

```
=head1 NAME
```

```
Embperl

=head1 Description

Here we have some text
```

will become

```
<pod>
    <head>
        <title>Embperl</title>
    </head>
    <sect1>
        <para>
            Here we have some text
        </para>
    </sect1>
</pod>
```

This basicly generates the same XML as Pod::XML

The Recipe EmbperlXSLT

Read the source (File, Memory)
Parse
Compile
Execute
- **Read the stylesheet**
- **``Compile'' the stylesheet**

``Compile'' the result of the Executing of the primary source
XSLT Transformation
Output

The usage of the XSLT Transformation allows the creation of different layouts from the same source.

By transformation into XSL-FO and appending of the XSL-FO Provider it's easly possible create PDF's

pod/content.epl

```
[- Execute ({inputfile => '*'})  -]
```

The recipe for the Embperl web is provided by the application object

```
sub get_recipe
```

```
{
my ($class, $r, $recipe) = @_ ;

my $self ;
my $param  = $r -> component -> param  ;
my $config = $r -> component -> config  ;
my ($src)  = $param -> inputfile =~ /^.*\.(.*?)$/ ;
my ($dest) = $r -> param -> uri =~ /^.*\.(.*?)$/ ;

if ($src eq 'pl')
    {
    $config -> syntax('Perl') ;
    return Embperl::Recipe::Embperl -> get_recipe ($r, $reci
    }

if ($src eq 'pod' || $src eq 'pm')
    {
    $config -> escmode(0) ;
    if ($dest eq 'pod')
        {
        $config -> syntax('Text') ;
        return Embperl::Recipe::Embperl -> get_recipe ($r, $
        }

    $config -> syntax('POD') ;
    if ($dest eq 'xml')
        {
        return Embperl::Recipe::Embperl -> get_recipe ($r, $
        }

    $config -> xsltstylesheet('pod.xsl') ;
    $r -> param -> uri =~ /^.*\/(.*)\.(.*?)$/ ;
    $param -> xsltparam({
            page      => $fdat{page} || 0,
            basename  => "'$1'",
            extension => "'$2'",
            imageuri  => "'$r->{imageuri}'",
            baseuri   => "'$r->{baseuri}'",
            }) ;
    return Embperl::Recipe::EmbperlXSLT -> get_recipe ($r, $
    }

if ($src eq 'epd')
    {
    $config -> escmode(0) ;
    $config -> options($config -> options | &Embperl::Consta

    if ($dest eq 'pod')
        {
        $config -> syntax('EmbperlBlocks') ;
        return Embperl::Recipe::Embperl -> get_recipe ($r, $
        }
```

```
        $config -> xsltstylesheet('pod.xsl') ;
        $r -> param -> uri =~ /^.*\/(.*)\.(.*?)$/ ;
        $param -> xsltparam({
                page      => $fdat{page} || 0,
                basename  => "'$1'",
                extension => "'$2'",
                imageuri  => "'$r->{imageuri}'",
                baseuri   => "'$r->{baseuri}'",
                }) ;
        return Embperl::Recipe::EmbperlPODXSLT -> get_recipe ($r
        }

    if ($src eq 'epl' || $src eq 'htm')
        {
        $config -> syntax('Embperl') ;
        return Embperl::Recipe::Embperl -> get_recipe ($r, $reci
        }

    $config -> syntax('Text') ;
    return Embperl::Recipe::Embperl -> get_recipe ($r, $recipe)
    }
```

The database application

Separation of Code, Layout and Data

db/epwebapp.pl

```
    use DBIx::Recordset ;

    BEGIN { Execute ({isa => '../epwebapp.pl'}) ; }

    sub init
        {
        my $self     = shift ;
        my $r        = shift ;

        $self -> SUPER::init ($r) ;

        $self -> initdb ($r) ;

        my $db = $r -> {db} ;

        $r -> {language_set} = DBIx::Recordset -> Search ({'!Dat
                                                           '!Tab

        if ($fdat{-add_category})
            {
```

```
        $self -> add_category ($r) ;
        $self -> get_category($r) ;
        }
    elsif ($fdat{-add_item})
        {
        $self -> add_item ($r) ;
        $self -> get_category($r) ;
        $self -> get_item_lang($r) ;
        }
    elsif ($fdat{-show_item})
        {
        $self -> get_category($r) ;
        $self -> get_item_lang($r) ;
        }
    else
        {
        $self -> get_category($r) ;
        $self -> get_item($r) ;
        }

    return 0 ;
    }

# ------------------------------------------------------------

sub initdb
    {
    my $self     = shift ;
    my $r        = shift ;

    $DBIx::Recordset::Debug = 2 ;
    *DBIx::Recordset::LOG = \*STDERR ;
    my $db = DBIx::Database -> new ({'!DataSource' => $r ->
                                     '!Username'   => $r ->
                                     '!Password'   => $r ->
                                     '!DBIAttr'    => { Rais
                                                        Long

                                     }) ;

    $db -> TableAttr ('*', '!SeqClass', "DBIx::Recordset::Fi
    $db -> TableAttr ('*', '!Filter',
        {
        'creationtime'  => [\&current_time, undef, DBIx::Rec
        'modtime'       => [\&current_time, undef, DBIx::Rec
        }) ;

    $r -> {db} = $db ;
    }

# ------------------------------------------------------------

sub current_time
```

```
    {
    my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)
                                                localtime(time)

    $mon++ ;
    $year += 1900 ;
    return "$year-$mon-$mday $hour:$min:$sec" ;
    }

# ------------------------------------------------------------

sub add_category
    {
    my $self     = shift ;
    my $r        = shift ;

    my $set = DBIx::Recordset -> Insert ({'!DataSource' => $
                                          '!Table'      => '
                                          '!Serial'     => '
                                           state        => 0
    my $id = $$set -> LastSerial ;
    my $langset = $r -> {language_set} ;
    my $txtset = DBIx::Recordset -> Setup ({'!DataSource' =>
                                            '!Table'      =>

    $$langset -> Reset ;
    while ($rec = $$langset -> Next)
        {
        $$txtset -> Insert ({category_id => $id,
                             language_id => $rec->{id},
                             category    => $fdat{"category_
        delete $fdat{"category_$rec->{id}"} ;
        }
    }

# ------------------------------------------------------------

sub add_item
    {
    my $self     = shift ;
    my $r        = shift ;

    my $set = DBIx::Recordset -> Insert ({'!DataSource' => $
                                          '!Table'      => '
                                          '!Serial'     => '
                                           url          => $
                                           category_id  => $
                                           state        => 0
    my $id = $$set -> LastSerial ;
    my $langset = $r -> {language_set} ;
    my $txtset = DBIx::Recordset -> Setup ({'!DataSource' =>
                                            '!Table'      =>
```

```
    $$langset -> Reset ;
    while ($rec = $$langset -> Next)
        {
        $$txtset -> Insert ({item_id => $id,
                             language_id => $rec->{id},
                             description => $fdat{"descripti
                             url         => $fdat{"url_$rec-
                             heading     => $fdat{"heading_$
        }

    $fdat{item_id} = $id ;
    }

# -----------------------------------------------------------

sub get_category
    {
    my $self     = shift ;
    my $r        = shift ;

    $r -> {category_set} = DBIx::Recordset -> Search ({'!Dat
                                                       '!Tab
                                                       '!Tab
                                                       'lang
                                                       $fdat


    }

# -----------------------------------------------------------

sub get_item
    {
    my $self     = shift ;
    my $r        = shift ;

    $r -> {item_set} = DBIx::Recordset -> Search ({'!DataSou
                                                   '!Tab
                                                   '!Tab
                                                   'lang
                                                   $fdat
                                                   $fdat
    }

# -----------------------------------------------------------

sub get_item_lang
    {
    my $self     = shift ;
    my $r        = shift ;

    $r -> {item_set} = DBIx::Recordset -> Search ({'!DataSou
                                                    '!Table'
```

```
                                                                '!TabRela
                                                                $fdat{cat
                                                                $fdat{ite
    }
```

db/addsel.epl

```
[- $r = shift -]

[= addsel1 =]

<ul>
    <li>
        [- $rec = $r -> {category_set}[$row] -]
        <a href="add.epl?category_id=[+ $rec -> {category_id
    </li>
</ul>

<form action="[+ $r -> param -> uri +]">

[= addsel2 =]<br>
[= addsel3 =]<br><br>

<table>
    <tr>
        [- $rec = $r -> {language_set}[$row] -]
        <td>[+ $rec -> {name} +]:</td><td><input type="text"
    </tr>
</table>

<br><br>
    <input type="submit" name="-add_cate

</form>
```

Multi-Language-Support

messages.pl stores the different texts

```
$r = shift ;

%messages =
    (
    'de' =>
        {
        'addsel1' => 'Klicken Sie auf die Kategorie zu der S
        'addsel2' => 'oder fügen Sie eine neue Kategorie hin
        'addsel3' => 'Falls Sie die Übersetzung nicht wissen
        'addsel4' => 'Kategorie hinzufügen',
```

```
            'add1'    => 'Hinzufügen eines neuen Eintrages zu',
            'add2'    => 'Bitte geben Sie die Beschreibung in so
            'add3'    => 'Hinzufügen zu',
            'heading' => 'Überschrift',
            'url'     => 'URL',
            'description'  => 'Beschreibung',
            'show2'   => 'Folgender Eintrag wurde erfolgreich de
            },
        'en' =>
            {
            'addsel1' => 'Click on the category for wich you wan
            'addsel2' => 'or add new category. Please enter the
            'addsel3' => 'If you don\'t know the translation lea
            'addsel4' => 'Add category',
            'add1'    => 'Add a new item to',
            'add2'    => 'Please enter the description in as muc
            'add3'    => 'Add to',
            'heading' => 'Heading',
            'url'     => 'URL',
            'description'  => 'Description',
            'show2'   => 'The following entry has been sucessful
            },
        ) ;

    $lang = $r -> param -> language ;
    push @{$r -> messages}, $messages{$lang} ;
    push @{$r -> default_messages}, $messages{'en'} if ($lang ne
```

Replacement of [= foo =] through the matching text, as far as available

$r -> `gettext('foo')` to get the matching text

Future...

There are many more possibilities of Embperl, for example session-handling and form validation

2.0b8 is the last beta, which is already quite stable

Final release of 2.0 is planed for the next three month.

Main addition will be documentation improvements and threads to use the full power of mod_perl 2 in threaded mode.

More informations can be found on the Embperl Web Site

http://perl.apache.org/embperl/

http://www.ecos.de/embperl/