```
#!/usr/bin/perl -wT
#
# NMS FormMail Version 3.10c1
# Modified by Value Hosting Australia
# Web2Mail
#

use strict;
use vars qw(
  $DEBUGGING $emulate_matts_code $secure %more_config
  $allow_empty_ref $max_recipients $mailprog @referers
  @allow_mail_to @recipients %recipient_alias
  @valid_ENV $date_fmt $style $send_confirmation_mail
  $confirmation_text $locale $charset $no_content
  $double_spacing $wrap_text $wrap_style $postmaster
);

# PROGRAM INFORMATION
# -------------------
# FormMail.pl Version 3.10c1
#
# This program is licensed in the same way as Perl
# itself. You are free to choose between the GNU Public
# License <http://www.gnu.org/licenses/gpl.html>  or
# the Artistic License
# <http://www.perl.com/pub/a/language/misc/Artistic.html>
#
# For help on configuration or installation see the
# README file or the POD documentation at the end of
# this file.

# USER CONFIGURATION SECTION
# --------------------------
# Modify these to your own settings. You might have to
# contact your system administrator if you do not run
# your own web server. If the purpose of these
# parameters seems unclear, please see the README file.
#
BEGIN
{
  $DEBUGGING         = 1;
  $emulate_matts_code= 0;
  $secure            = 1;
  $allow_empty_ref   = 1;
  $max_recipients    = 5;
  $mailprog          = '/usr/sbin/sendmail -oi -t';
  $postmaster        = '';
  @referers          = qw(hawkersa.info localhost);
  @allow_mail_to     = qw(udob@internode.on.net localhost);
  @recipients        = ();
  %recipient_alias   = ();
  @valid_ENV         = qw(REMOTE_HOST REMOTE_ADDR REMOTE_USER HTTP_USER_AGENT);
  $locale            = '';
  $charset           = 'iso-8859-1';
  $date_fmt          = '%A, %B %d, %Y at %H:%M:%S';
  $style             = '/css/nms.css';
  $no_content        = 0;
  $double_spacing    = 1;
  $wrap_text         = 0;
  $wrap_style        = 1;
  $send_confirmation_mail = 0;
  $confirmation_text = <<'END_OF_CONFIRMATION';


Thank you for your form submission.

END_OF_CONFIRMATION
```

```
# You may need to uncomment the line below and adjust the path.
# use lib './lib';

# USER CUSTOMISATION SECTION
# --------------------------
# Place any custom code here



# USER CUSTOMISATION << END >>
# ----------------------------
# (no user serviceable parts beyond here)
}

#
# The code below consists of module source inlined into this
# script to make it a standalone CGI.
#
# Inlining performed by NMS inline - see /v2/buildtools/inline
# in CVS at http://sourceforge.net/projects/nms-cgi for details.
#
BEGIN {


$CGI::NMS::INLINED_SOURCE::CGI_NMS_Mailer = <<'END_INLINED_CGI_NMS_Mailer';
package CGI::NMS::Mailer;
use strict;

use POSIX qw(strftime);

=head1 NAME

CGI::NMS::Mailer - email sender base class

=head1 SYNOPSYS

  use base qw(CGI::NMS::Mailer);

  ...

=head1 DESCRIPTION

This is a base class for classes implementing low-level email
sending objects for use within CGI scripts.

=head1 METHODS

=over

=item output_trace_headers ( TRACEINFO )

Uses the print() virtual method to output email abuse tracing headers
including whatever useful information can be gleaned from the CGI
environment variables.

The TRACEINFO parameter should be a short string giving the name and
version of the CGI script.

=cut

sub output_trace_headers {
  my ($self, $traceinfo) = @_;

  $ENV{REMOTE_ADDR} =~ /^\[?([\d\.]{7,15})\]?$/ or die
     "failed to get remote address from [$ENV{REMOTE_ADDR}], so can't send
traceable email";
  $self->print("Received: from [$1]\n");
```

```
  my $me = ($ENV{SERVER_NAME} =~ /^([\w\-\.]{1,100})$/ ? $1 : 'unknown');
  $self->print("\tby $me ($traceinfo)\n");

  my $date = strftime '%a, %e %b %Y %H:%M:%S GMT', gmtime;
  $self->print("\twith HTTP; $date\n");

  if ($ENV{SCRIPT_NAME} =~ /^([\w\-\.\/]{1,100})$/) {
    $self->print("\t(script-name $1)\n");
  }

  if (defined $ENV{HTTP_HOST} and $ENV{HTTP_HOST} =~ /^([\w\-\.]{1,100})$/) {
    $self->print("\t(http-host $1)\n");
  }

  my $ff = $ENV{HTTP_X_FORWARDED_FOR};
  if (defined $ff) {
    $ff =~ /^\s*([\w\-\.\[\] ,]{1,200})\s*/ or die
      "malformed X-Forwarded-For [$ff], suspect attack, aborting";

    $self->print("\t(http-x-forwarded-for $1)\n");
  }

  my $ref = $ENV{HTTP_REFERER};
  if (defined $ref and $ref =~ /^([\w\-\.\/\:\;\%\@\#\~\=\+\?]{1,100})$/) {
    $self->print("\t(http-referer $1)\n");
  }
}

=back

=head1 VIRTUAL METHODS

Subclasses must implement the following methods:

=over

=item newmail ( TRACEINFO, SENDER, @RECIPIENTS )

Starts a new email.  TRACEINFO is the script name and version, SENDER is
the email address to use as the envelope sender and @RECIPIENTS is a list
of recipients.  Dies on error.

=item print ( @ARGS )

Concatenates the arguments and appends them to the email.  Both the
header and the body should be sent in this way, separated by a single
blank line.  Dies on error.

=item endmail ()

Finishes the email, flushing buffers and sending it.  Dies on error.

=back

=head1 SEE ALSO

L<CGI::NMS::Mailer::Sendmail>, L<CGI::NMS::Mailer::SMTP>,
L<CGI::NMS::Script>

=head1 MAINTAINERS

The NMS project, E<lt>http://nms-cgi.sourceforge.net/E<gt>

To request support or report bugs, please email
E<lt>nms-cgi-support@lists.sourceforge.netE<gt>

=head1 COPYRIGHT
```

```
=head1 LICENSE

This module is free software; you are free to redistribute it
and/or modify it under the same terms as Perl itself.

=cut

1;


END_INLINED_CGI_NMS_Mailer


$CGI::NMS::INLINED_SOURCE::CGI_NMS_Mailer_Sendmail =
<<'END_INLINED_CGI_NMS_Mailer_Sendmail';
package CGI::NMS::Mailer::Sendmail;
use strict;

use IO::File;
BEGIN {
do {
  unless (eval {local $SIG{__DIE__} ; require CGI::NMS::Mailer}) {
    eval $CGI::NMS::INLINED_SOURCE::CGI_NMS_Mailer or die $@;
    $INC{'CGI/NMS/Mailer.pm'} = 1;
  }
  undef $CGI::NMS::INLINED_SOURCE::CGI_NMS_Mailer; # to save memory
};

 import CGI::NMS::Mailer }
use base qw(CGI::NMS::Mailer);

=head1 NAME

CGI::NMS::Mailer::Sendmail - mail sender using sendmail

=head1 SYNOPSYS

  my $mailer = CGI::NMS::Mailer::Sendmail->new('/usr/lib/sendmail -oi -t');

  $mailer->newmail($from, $to);
  $mailer->print($email_header_and_body);
  $mailer->endmail;

=head1 DESCRIPTION

This implementation of the mailer object defined in L<CGI::NMS::Mailer>
uses a piped open to the UNIX sendmail program to send the email.

=head1 CONSTRUCTORS

=over

=item new ( MAILPROG )

MAILPROG msut be the shell command to which a pipe is opened, including
all nessessary switches to cause the sendmail program to read the email
recipients from the header of the email.

=cut

sub new {
  my ($pkg, $mailprog) = @_;

  return bless { Mailprog => $mailprog }, $pkg;
}
```

```
=back

=head1 METHODS

See L<CGI::NMS::Mailer> for the user interface to these methods.

=over

=item newmail ( SCRIPTNAME, POSTMASTER, @RECIPIENTS )

Opens the sendmail pipe and outputs trace headers.

=cut

sub newmail {
  my ($self, $scriptname, $postmaster, @recipients) = @_;

  my $command = $self->{Mailprog};
  $command .= qq{ -f "$postmaster"} if $postmaster;
  my $pipe;
  eval { local $SIG{__DIE__};
         $pipe = IO::File->new("| $command");
       };
  if ($@) {
    die $@ unless $@ =~ /Insecure directory/;
    delete $ENV{PATH};
    $pipe = IO::File->new("| $command");
  }

  die "Can't open mailprog [$command]\n" unless $pipe;
  $self->{Pipe} = $pipe;

  $self->output_trace_headers($scriptname);
}

=item print ( @ARGS )

Writes some email body to the sendmail pipe.

=cut

sub print {
  my ($self, @args) = @_;

  $self->{Pipe}->print(@args) or die "write to sendmail pipe: $!";
}

=item endmail ()

Closes the sendmail pipe.

=cut

sub endmail {
  my ($self) = @_;

  $self->{Pipe}->close or die "close sendmail pipe failed,
mailprog=[$self->{Mailprog}]";
  delete $self->{Pipe};
}

=back

=head1 MAINTAINERS

The NMS project, E<lt>http://nms-cgi.sourceforge.net/E<gt>

To request support or report bugs, please email
```

```
E<lt>nms-cgi-support@lists.sourceforge.netE<gt>

=head1 COPYRIGHT

Copyright 2003 London Perl Mongers, All rights reserved

=head1 LICENSE

This module is free software; you are free to redistribute it
and/or modify it under the same terms as Perl itself.

=cut

1;


END_INLINED_CGI_NMS_Mailer_Sendmail


$CGI::NMS::INLINED_SOURCE::CGI_NMS_Mailer_SMTP =
<<'END_INLINED_CGI_NMS_Mailer_SMTP';
package CGI::NMS::Mailer::SMTP;
use strict;

use IO::Socket;
BEGIN {
do {
  unless (eval {local $SIG{__DIE__} ; require CGI::NMS::Mailer}) {
    eval $CGI::NMS::INLINED_SOURCE::CGI_NMS_Mailer or die $@;
    $INC{'CGI/NMS/Mailer.pm'} = 1;
  }
  undef $CGI::NMS::INLINED_SOURCE::CGI_NMS_Mailer; # to save memory
};

 import CGI::NMS::Mailer }
use base qw(CGI::NMS::Mailer);

=head1 NAME

CGI::NMS::Mailer::SMTP - mail sender using SMTP

=head1 SYNOPSYS

  my $mailer = CGI::NMS::Mailer::SMTP->new('mailhost.bigisp.net');

  $mailer->newmail($from, $to);
  $mailer->print($email_header_and_body);
  $mailer->endmail;

=head1 DESCRIPTION

This implementation of the mailer object defined in L<CGI::NMS::Mailer>
uses an SMTP connection to a mail relay to send the email.

=head1 CONSTRUCTORS

=over

=item new ( MAILHOST )

MAILHOST must be the name or dotted decimal IP address of an SMTP
server that will relay mail for the web server.

=cut

sub new {
  my ($pkg, $mailhost) = @_;
```

```
  $mailhost .= ':25' unless $mailhost =~ /:/;
  return bless { Mailhost => $mailhost }, $pkg;
}

=back

=head1 METHODS

See L<CGI::NMS::Mailer> for the user interface to these methods.

=over

=item newmail ( SCRIPTNAME, SENDER, @RECIPIENTS )

Opens the SMTP connection and sends trace headers.

=cut

sub newmail {
  my ($self, $scriptname, $sender, @recipients) = @_;

  $self->{Sock} = IO::Socket::INET->new($self->{Mailhost});
  defined $self->{Sock} or die "connect to [$self->{Mailhost}]: $!";

  my $banner = $self->_smtp_response;
  $banner =~ /^2/ or die "bad SMTP banner [$banner] from [$self->{Mailhost}]";

  my $helohost = ($ENV{SERVER_NAME} =~ /^([\w\-\.]+)$/ ? $1 : '.');
  $self->_smtp_command("HELO $helohost");
  $self->_smtp_command("MAIL FROM:<$sender>");
  foreach my $r (@recipients) {
    $self->_smtp_command("RCPT TO:<$r>");
  }
  $self->_smtp_command("DATA", '3');

  $self->output_trace_headers($scriptname);
}

=item print ( @ARGS )

Writes some email body to the SMTP socket.

=cut

sub print {
  my ($self, @args) = @_;

  my $text = join '', @args;
  $text =~ s#\n#\015\012#g;
  $text =~ s#^\.#..#mg;

  $self->{Sock}->print($text) or die "write to SMTP socket: $!";
}

=item endmail ()

Finishes sending the mail and closes the SMTP connection.

=cut

sub endmail {
  my ($self) = @_;

  $self->_smtp_command(".");
  $self->_smtp_command("QUIT");
  delete $self->{Sock};
}
```

```
=back

=head1 PRIVATE METHODS

These methods should be called from within this module only.

=over

=item _smtp_getline ()

Reads a line from the SMTP socket, and returns it as a string,
including the terminating newline sequence.

=cut

sub _smtp_getline {
  my ($self) = @_;

  my $sock = $self->{Sock};
  my $line = <$sock>;
  defined $line or die "read from SMTP server: $!";

  return $line;
}

=item _smtp_response ()

Reads a command response from the SMTP socket, and returns it as
a single string.  A multiline responses is returned as a multiline
string, and the terminating newline sequence is always included.

=cut

sub _smtp_response {
  my ($self) = @_;

  my $line = $self->_smtp_getline;
  my $resp = $line;
  while ($line =~ /^\d\d\d\-/) {
    $line = $self->_smtp_getline;
    $resp .= $line;
  }
  return $resp;
}

=item _smtp_command ( COMMAND [,EXPECT] )

Sends the SMTP command COMMAND to the SMTP server, and reads a line
in response.  Dies unless the first character of the response is
the character EXPECT, which defaults to '2'.

=cut

sub _smtp_command {
  my ($self, $command, $expect) = @_;
  defined $expect or $expect = '2';

  $self->{Sock}->print("$command\015\012") or die
    "write [$command] to SMTP server: $!";

  my $resp = $self->_smtp_response;
  unless (substr($resp, 0, 1) eq $expect) {
    die "SMTP command [$command] gave response [$resp]";
  }
}

=back
```

```
=head1 MAINTAINERS

The NMS project, E<lt>http://nms-cgi.sourceforge.net/E<gt>

To request support or report bugs, please email
E<lt>nms-cgi-support@lists.sourceforge.netE<gt>

=head1 COPYRIGHT

Copyright 2003 London Perl Mongers, All rights reserved

=head1 LICENSE

This module is free software; you are free to redistribute it
and/or modify it under the same terms as Perl itself.

=cut

1;


END_INLINED_CGI_NMS_Mailer_SMTP


unless (eval {local $SIG{__DIE__} ; require CGI::NMS::Charset}) {
  eval <<'END_INLINED_CGI_NMS_Charset' or die $@;
package CGI::NMS::Charset;
use strict;

require 5.00404;

use vars qw($VERSION);
$VERSION = sprintf '%d.%.2d', (q$Revision: 1.1 $ =~ /(\d+)\.(\d+)/);

=head1 NAME

CGI::NMS::Charset - a charset-aware object for handling text strings

=head1 SYNOPSIS

  my $cs = CGI::NMS::Charset->new('iso-8859-1');

  my $safe_to_put_in_html = $cs->escape($untrusted_user_input);

  my $printable = &{ $cs->strip_nonprint_coderef }( $input );
  my $escaped = &{ $cs->escape_html_coderef }( $printable );

=head1 DESCRIPTION

Each object of class C<CGI::NMS::Charset> is bound to a particular
character set when it is created.  The object provides methods to
generate coderefs to perform a couple of character set dependent
operations on text strings.

=cut

=head1 CONSTRUCTORS

=over

=item new ( CHARSET )

Creates a new C<CGI::NMS::Charset> object, suitable for handing text
in the character set CHARSET.  The CHARSET parameter must be a
character set string, such as C<us-ascii> or C<utf-8> for example.

=cut
```

```
sub new
{
   my ($pkg, $charset) = @_;

   my $self = { CHARSET => $charset };

   if ($charset =~ /^utf-8$/i)
   {
      $self->{SN} = \&_strip_nonprint_utf8;
      $self->{EH} = \&_escape_html_utf8;
   }
   elsif ($charset =~ /^iso-8859/i)
   {
      $self->{SN} = \&_strip_nonprint_8859;
      if ($charset =~ /^iso-8859-1$/i)
      {
         $self->{EH} = \&_escape_html_8859_1;
      }
      else
      {
         $self->{EH} = \&_escape_html_8859;
      }
   }
   elsif ($charset =~ /^us-ascii$/i)
   {
      $self->{SN} = \&_strip_nonprint_ascii;
      $self->{EH} = \&_escape_html_8859_1;
   }
   else
   {
      $self->{SN} = \&_strip_nonprint_weak;
      $self->{EH} = \&_escape_html_weak;
   }

   return bless $self, $pkg;
}

=back

=head1 METHODS

=over

=item charset ()

Returns the CHARSET string that was passed to the constructor.

=cut

sub charset
{
   my ($self) = @_;

   return $self->{CHARSET};
}

=item escape ( STRING )

Returns a copy of STRING with runs of non-printable characters
replaced with spaces and HTML metacharacters replaced with the
equivalent entities.

If STRING is undef then the empty string will be returned.

=cut

sub escape
{
```

```
   my ($self, $string) = @_;

   return &{ $self->{EH} }(  &{ $self->{SN} }($string)  );
}

=item strip_nonprint_coderef ()

Returns a reference to a sub to replace runs of non-printable
characters with spaces, in a manner suited to the charset in
use.

The returned coderef points to a sub that takes a single readonly
string argument and returns a modified version of the string.  If
undef is passed to the function then the empty string will be
returned.

=cut

sub strip_nonprint_coderef
{
   my ($self) = @_;

   return $self->{SN};
}

=item escape_html_coderef ()

Returns a reference to a sub to escape HTML metacharacters in
a manner suited to the charset in use.

The returned coderef points to a sub that takes a single readonly
string argument and returns a modified version of the string.

=cut

sub escape_html_coderef
{
   my ($self) = @_;

   return $self->{EH};
}

=back

=head1 DATA TABLES

=over

=item C<%eschtml_map>

The C<%eschtml_map> hash maps C<iso-8859-1> characters to the
equivalent HTML entities.

=cut

use vars qw(%eschtml_map);
%eschtml_map = (
                  ( map {chr($_) => "&#$_;"} (0..255) ),
                  '<' => '&lt;',
                  '>' => '&gt;',
                  '&' => '&amp;',
                  '"' => '&quot;',
               );

=back

=head1 PRIVATE FUNCTIONS
```

```
These functions are returned by the strip_nonprint_coderef() and
escape_html_coderef() methods and invoked by the escape() method.
The function most appropriate to the character set in use will be
chosen.

=over

=item _strip_nonprint_utf8

Returns a copy of STRING with everything but printable C<us-ascii>
characters and valid C<utf-8> multibyte sequences replaced with
space characters.

=cut

sub _strip_nonprint_utf8
{
   my ($string) = @_;
   return '' unless defined $string;

   $string =~
   s%
    ( [\t\n\040-\176]               # printable us-ascii
    | [\xC2-\xDF][\x80-\xBF]        # U+00000080 to U+000007FF
    | \xE0[\xA0-\xBF][\x80-\xBF]    # U+00000800 to U+00000FFF
    | [\xE1-\xEF][\x80-\xBF]{2}     # U+00001000 to U+0000FFFF
    | \xF0[\x90-\xBF][\x80-\xBF]{2} # U+00010000 to U+0003FFFF
    | [\xF1-\xF7][\x80-\xBF]{3}     # U+00040000 to U+001FFFFF
    | \xF8[\x88-\xBF][\x80-\xBF]{3} # U+00200000 to U+00FFFFFF
    | [\xF9-\xFB][\x80-\xBF]{4}     # U+01000000 to U+03FFFFFF
    | \xFC[\x84-\xBF][\x80-\xBF]{4} # U+04000000 to U+3FFFFFFF
    | \xFD[\x80-\xBF]{5}            # U+40000000 to U+7FFFFFFF
    ) | .
   %
    defined $1 ? $1 : ' '
   %gexs;

   #
   # U+FFFE, U+FFFF and U+D800 to U+DFFF are dangerous and
   # should be treated as invalid combinations, according to
   # http://www.cl.cam.ac.uk/~mgk25/unicode.html
   #
   $string =~ s%\xEF\xBF[\xBE-\xBF]% %g;
   $string =~ s%\xED[\xA0-\xBF][\x80-\xBF]% %g;

   return $string;
}

=item _escape_html_utf8 ( STRING )

Returns a copy of STRING with any HTML metacharacters
escaped.  Escapes all but the most commonly occurring C<us-ascii>
characters and bytes that might form part of valid C<utf-8>
multibyte sequences.

=cut

sub _escape_html_utf8
{
   my ($string) = @_;

   $string =~ s|([^\w \t\r\n\-\.\,\x80-\xFD])| $eschtml_map{$1} |ge;
   return $string;
}

=item _strip_nonprint_weak ( STRING )

Returns a copy of STRING with sequences of NULL characters
```

```
replaced with space characters.

=cut

sub _strip_nonprint_weak
{
   my ($string) = @_;
   return '' unless defined $string;

   $string =~ s/\0+/ /g;
   return $string;
}

=item _escape_html_weak ( STRING )

Returns a copy of STRING with any HTML metacharacters escaped.
In order to work in any charset, escapes only E<lt>, E<gt>, C<">
and C<&> characters.

=cut

sub _escape_html_weak
{
   my ($string) = @_;

   $string =~ s/[<>"&]/$eschtml_map{$1}/eg;
   return $string;
}

=item _escape_html_8859_1 ( STRING )

Returns a copy of STRING with all but the most commonly
occurring printable characters replaced with HTML entities.
Only suitable for C<us-ascii> or C<iso-8859-1> input.

=cut

sub _escape_html_8859_1
{
   my ($string) = @_;

   $string =~ s|([^\w \t\r\n\-\.\,\/\:])| $eschtml_map{$1} |ge;
   return $string;
}

=item _escape_html_8859 ( STRING )

Returns a copy of STRING with all but the most commonly
occurring printable C<us-ascii> characters and characters
that might be printable in some C<iso-8859-*> charset
replaced with HTML entities.

=cut

sub _escape_html_8859
{
   my ($string) = @_;

   $string =~ s|([^\w \t\r\n\-\.\,\/\:\240-\377])| $eschtml_map{$1} |ge;
   return $string;
}

=item _strip_nonprint_8859 ( STRING )

Returns a copy of STRING with runs of characters that are not
printable in any C<iso-8859-*> charset replaced with spaces.

=cut
```

```
sub _strip_nonprint_8859
{
   my ($string) = @_;
   return '' unless defined $string;

   $string =~ tr#\t\n\040-\176\240-\377# #cs;
   return $string;
}

=item _strip_nonprint_ascii ( STRING )

Returns a copy of STRING with runs of characters that are not
printable C<us-ascii> replaced with spaces.

=cut

sub _strip_nonprint_ascii
{
   my ($string) = @_;
   return '' unless defined $string;

   $string =~ tr#\t\n\040-\176# #cs;
   return $string;
}

=back

=head1 MAINTAINERS

The NMS project, E<lt>http://nms-cgi.sourceforge.net/E<gt>

To request support or report bugs, please email
E<lt>nms-cgi-support@lists.sourceforge.netE<gt>

=head1 COPYRIGHT

Copyright 2002-2003 London Perl Mongers, All rights reserved

=head1 LICENSE

This module is free software; you are free to redistribute it
and/or modify it under the same terms as Perl itself.

=cut

1;


END_INLINED_CGI_NMS_Charset
  $INC{'CGI/NMS/Charset.pm'} = 1;
}


unless (eval {local $SIG{__DIE__} ; require CGI::NMS::Mailer::ByScheme}) {
  eval <<'END_INLINED_CGI_NMS_Mailer_ByScheme' or die $@;
package CGI::NMS::Mailer::ByScheme;
use strict;

=head1 NAME

CGI::NMS::Mailer::ByScheme - mail sending engine switch

=head1 SYNOPSYS

  my $mailer = CGI::NMS::Mailer::ByScheme->new('/usr/lib/sendmail -oi -t');

  my $mailer = CGI::NMS::Mailer::ByScheme->new('SMTP:mailhost.bigisp.net');
```

```
=head1 DESCRIPTION

This implementation of the mailer object defined in L<CGI::NMS::Mailer>
chooses between L<CGI::NMS::Mailer::SMTP> and L<CGI::NMS::Mailer::Sendmail>
based on the string passed to new().

=head1 CONSTRUCTORS

=over

=item new ( ARGUMENT )

ARGUMENT must either be the string C<SMTP:> followed by the name or
dotted decimal IP address of an SMTP server that will relay mail
for the web server, or the path to a sendmail compatible binary,
including switches.

=cut

sub new {
  my ($pkg, $argument) = @_;

  if ($argument =~ /^SMTP:([\w\-\.]+(:\d+)?)/i) {
    my $mailhost = $1;

do {
  unless (eval {local $SIG{__DIE__} ; require CGI::NMS::Mailer::SMTP}) {
    eval $CGI::NMS::INLINED_SOURCE::CGI_NMS_Mailer_SMTP or die $@;
    $INC{'CGI/NMS/Mailer/SMTP.pm'} = 1;
  }
  undef $CGI::NMS::INLINED_SOURCE::CGI_NMS_Mailer_SMTP; # to save memory
};


    return CGI::NMS::Mailer::SMTP->new($mailhost);
  }
  else {

do {
  unless (eval {local $SIG{__DIE__} ; require CGI::NMS::Mailer::Sendmail}) {
    eval $CGI::NMS::INLINED_SOURCE::CGI_NMS_Mailer_Sendmail or die $@;
    $INC{'CGI/NMS/Mailer/Sendmail.pm'} = 1;
  }
  undef $CGI::NMS::INLINED_SOURCE::CGI_NMS_Mailer_Sendmail; # to save memory
};


    return CGI::NMS::Mailer::Sendmail->new($argument);
  }
}

=back

=head1 MAINTAINERS

The NMS project, E<lt>http://nms-cgi.sourceforge.net/E<gt>

To request support or report bugs, please email
E<lt>nms-cgi-support@lists.sourceforge.netE<gt>

=head1 COPYRIGHT

Copyright 2003 London Perl Mongers, All rights reserved

=head1 LICENSE

This module is free software; you are free to redistribute it
```

```
and/or modify it under the same terms as Perl itself.

=cut

1;


END_INLINED_CGI_NMS_Mailer_ByScheme
  $INC{'CGI/NMS/Mailer/ByScheme.pm'} = 1;
}


unless (eval {local $SIG{__DIE__} ; require CGI::NMS::Script}) {
  eval <<'END_INLINED_CGI_NMS_Script' or die $@;
package CGI::NMS::Script;
use strict;

use CGI;
use POSIX qw(locale_h strftime);
use CGI::NMS::Charset;

=head1 NAME

CGI::NMS::Script - base class for NMS script modules

=head1 SYNOPSYS

  use base qw(CGI::NMS::Script);

  ...

=head1 DESCRIPTION

This module is a base class for the C<CGI::NMS::Script::*> modules,
which implement plugin replacements for Matt Wright's Perl CGI
scripts.

=head1 CONSTRUCTORS

=over

=item new ( CONFIG )

Creates a new C<CGI::NMS::Script> object and performs compile time
initialisation.

CONFIG is a key,value,key,value list, which will be stored as a hash
within the object, under the name C<CFG>.

=cut

sub new {
  my ($pkg, @cfg) = @_;

  my $self = bless {}, $pkg;

  $self->{CFG} = {
    DEBUGGING           => 0,
    emulate_matts_code  => 0,
    secure              => 1,
    locale              => '',
    charset             => 'iso-8859-1',
    style               => '',
    cgi_post_max        => 1000000,
    cgi_disable_uploads => 1,

    $self->default_configuration,
```

```
    @cfg
  };

  $self->{Charset} = CGI::NMS::Charset->new( $self->{CFG}{charset} );

  $self->init;

  return $self;
}

=back

=item CONFIGURATION SETTINGS

Values for the following configuration settings can be passed to new().

Subclasses for different NMS scripts will define their own set of
configuration settings, but they all inherit these as well.

=over

=item C<DEBUGGING>

If this is set to a true value, then the error message will be displayed
in the browser if the script suffers a fatal error.  This should be set
to 0 once the script is in service, since error messages may contain
sensitive information such as file paths which could be useful to
attackers.

Default: 0

=item C<name_and_version>

The name and version of the NMS script, as a single string.

=item C<emulate_matts_code>

When this variable is set to a true value (e.g. 1) the script will work
in exactly the same way as its counterpart at Matt's Script Archive. If
it is set to a false value (e.g. 0) then more advanced features and
security checks are switched on. We do not recommend changing this
variable to 1, as the resulting drop in security may leave your script
open to abuse.

Default: 0

=item C<secure>

When this variable is set to a true value (e.g. 1) many additional
security features are turned on.  We do not recommend changing this
variable to 0, as the resulting drop in security may leave your script
open to abuse.

Default: 1

=item C<locale>

This determines the language that is used in the format_date() method -
by default this is blank and the language will probably be English.

Default: ''

=item C<charset>

The character set to use for output documents.

Default: 'iso-8859-1'
```

```
=item C<style>

This is the URL of a CSS stylesheet which will be used for script
generated messages.  This should probably be the same as the one that
you use for all the other pages.  This should be a local absolute URI
fragment.  Set C<style> to 0 or the empty string if you don't want to
use style sheets.

Default: '';

=item C<cgi_post_max>

The variable C<$CGI::POST_MAX> is gets set to this value before the
request is handled.

Default: 1000000

=item C<cgi_disable_uploads>

The variable C<CGI::DISABLE_UPLOADS> gets set to this value before
the request is handled.

Default: 1

=item C<no_xml_doc_header>

If this is set to a true value then the output_cgi_html_header() method
will omit the XML document header that it would normally output.  This
means that the output document will not be strictly valid XHTML, but it
may work better in some older browsers.

Default: not set

=item C<no_doctype_doc_header>

If this is set to a true value then the output_cgi_html_header() method
will omit the DOCTYPE document header that it would normally output.
This means that the output document will not be strictly valid XHTML, but
it may work better in some older browsers.

Default: not set

=item C<no_xmlns_doc_header>

If this is set to a true value then the output_cgi_html_header() method
will omit the C<xmlns> attribute from the opening C<html> tag that it
outputs.

=back

=head1 METHODS

=over

=item request ()

This is the method that the CGI script invokes once for each run of the
CGI.  This implementation sets up some things that are common to all NMS
scripts and then invokes the virtual method handle_request() to do the
script specific processing.

=cut

sub request {
  my ($self) = @_;

  local ($CGI::POST_MAX, $CGI::DISABLE_UPLOADS);
  $CGI::POST_MAX        = $self->{CFG}{cgi_post_max};
```

```
  $CGI::DISABLE_UPLOADS = $self->{CFG}{cgi_disable_uploads};

  $ENV{PATH} =~ /(.*)/m or die;
  local $ENV{PATH} = $1;
  local $ENV{ENV}  = '';

  $self->{CGI} = CGI->new;
  $self->{Done_Header} = 0;

  my $old_locale;
  if ($self->{CFG}{locale}) {
    $old_locale = POSIX::setlocale( LC_TIME );
    POSIX::setlocale( LC_TIME, $self->{CFG}{locale} );
  }

  eval { local $SIG{__DIE__} ; $self->handle_request };
  my $err = $@;

  if ($self->{CFG}{locale}) {
    POSIX::setlocale( LC_TIME, $old_locale );
  }

  if ($err) {
    my $message;
    if ($self->{CFG}{DEBUGGING}) {
      $message = $self->escape_html($err);
    }
    else {
      $message = "See the web server's error log for details";
    }

    $self->output_cgi_html_header;
    print <<END;
 <head>
  <title>Error</title>
 </head>
 <body>
  <h1>Application Error</h1>
  <p>
   An error has occurred in the program
  </p>
  <p>
   $message
  </p>
 </body>
</html>
END

    if ($ENV{SCRIPT_NAME} =~ m#^([\w\-\/\.\:]{1,100})$#) {
      $err = "$1: $err";
    }
    warn $err;
  }
}

=item output_cgi_html_header ()

Prints the CGI content-type header and the standard header lines for
an XHTML document, unless the header has already been output.

=cut

sub output_cgi_html_header {
  my ($self) = @_;

  return if $self->{Done_Header};

  $self->output_cgi_header;
```

```
  unless ($self->{CFG}{no_xml_doc_header}) {
    print qq|<?xml version="1.0" encoding="$self->{CFG}{charset}"?>\n|;
  }

  unless ($self->{CFG}{no_doctype_doc_header}) {
    print <<END;
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
END
  }

  if ($self->{CFG}{no_xmlns_doc_header}) {
    print "<html>\n";
  }
  else {
    print qq|<html xmlns="http://www.w3.org/1999/xhtml">\n|;
  }

  $self->{Done_Header} = 1;
}

=item output_cgi_header ()

Outputs the CGI header for an HTML document.

=cut

sub output_cgi_header {
  my ($self) = @_;

  my $charset = $self->{CFG}{charset};
  my $cgi = $self->cgi_object;

  if ($CGI::VERSION >= 2.57) {
    # This is the correct way to set the charset
    print $cgi->header('-type'=>'text/html', '-charset'=>$charset);
  }
  else {
    # However CGI.pm older than version 2.57 doesn't have the
    # -charset option so we cheat:
    print $cgi->header('-type' => "text/html; charset=$charset");
  }
}

=item output_style_element ()

Outputs the C<link rel=stylesheet> header line, if a style sheet URL is
configured.

=cut

sub output_style_element {
  my ($self) = @_;

  if ($self->{CFG}{style}) {
    print qq|<link rel="stylesheet" type="text/css" href="$self->{CFG}{style}"
/>\n|;
  }
}

=item cgi_object ()

Returns a reference to the C<CGI.pm> object for this request.

=cut

sub cgi_object {
```

```
  my ($self) = @_;

   return $self->{CGI};
}

=item escape_html ( INPUT )

Returns a copy of the string INPUT with all HTML metacharacters escaped.

=cut

sub escape_html {
  my ($self, $input) = @_;

  return $self->{Charset}->escape($input);
}

=item strip_nonprint ( INPUT )

Returns a copy of the string INPUT with runs of nonprintable characters
replaced by spaces.

=cut

sub strip_nonprint {
  my ($self, $input) = @_;

  &{ $self->{Charset}->strip_nonprint_coderef }($input);
}

=item format_date ( FORMAT_STRING [,GMT_OFFSET] )

Returns the current time and date formated by C<strftime> according
to the format string FORMAT_STRING.

If GMT_OFFSET is undefined or the empty string then local time is
used.  Otherwise GMT is used, with an offset of GMT_OFFSET hours.

=cut

sub format_date {
  my ($self, $format_string, $gmt_offset) = @_;

  if (defined $gmt_offset and length $gmt_offset) {
    return strftime $format_string, gmtime(time + 60*60*$gmt_offset);
  }
  else {
    return strftime $format_string, localtime;
  }
}

=item name_and_version ()

Returns the NMS script version string that was passed to the constructor.

=cut

sub name_and_version {
    my ($self) = @_;

    return $self->{CFG}{name_and_version};
}

=back

=head1 VIRTUAL METHODS

Subclasses for individual NMS scripts must provide the following
```

```
methods:

=over

=item default_configuration ()

Invoked from new(), this method must return the default script
configuration as a key,value,key,value list.  Configuration options
passed to new() will override those set by this method.

=item init ()

Invoked from new(), this method can be used to do any script specific
object initialisation.  There is a default implementation, which does
nothing.

=cut

sub init {}

=item handle_request ()

Invoked from request(), this method is responsible for performing the
bulk of the CGI processing.  Any fatal errors raised here will be
trapped and treated according to the C<DEBUGGING> configuration setting.

=back

=head1 SEE ALSO

L<CGI::NMS::Charset>, L<CGI::NMS::Script::FormMail>

=head1 MAINTAINERS

The NMS project, E<lt>http://nms-cgi.sourceforge.net/E<gt>

To request support or report bugs, please email
E<lt>nms-cgi-support@lists.sourceforge.netE<gt>

=head1 COPYRIGHT

Copyright 2003 London Perl Mongers, All rights reserved

=head1 LICENSE

This module is free software; you are free to redistribute it
and/or modify it under the same terms as Perl itself.

=cut

1;


END_INLINED_CGI_NMS_Script
  $INC{'CGI/NMS/Script.pm'} = 1;
}


unless (eval {local $SIG{__DIE__} ; require CGI::NMS::Validator}) {
  eval <<'END_INLINED_CGI_NMS_Validator' or die $@;
package CGI::NMS::Validator;
use strict;

=head1 NAME

CGI::NMS::Validator - validation methods

=head1 SYNOPSYS
```

```
  use base qw(CGI::NMS::Validator);

  ...

  my $validurl = $self->validate_abs_url($url);

=head1 DESCRIPTION

This module provides methods to validate some of the types of
data the occur in CGI scripts, such as URLs and email addresses.

=head1 METHODS

These C<validate_*> methods all return undef if the item passed
in is invalid, otherwise they return the valid item.

Some of these methods attempt to transform invalid input into valid
input (for example, validate_abs_url() will prepend http:// if missing)
so the returned valid item may not be the same as that passed in.

The returned value is always detainted.

=over

=item validate_abs_url ( URL )

Validates an absolute URL.

=cut

sub validate_abs_url {
  my ($self, $url) = @_;

  $url = "http://$url" unless $url =~ /:/;
  $url =~ s#^(\w+://)# lc $1 #e;

  $url =~ m< ^ ( (?:ftp|http|https):// [\w\-\.]{1,100} (?:\:\d{1,5})? ) ( /*
(?:[^\./].*)? ) $ >mx
    or return '';

  my ($prefix, $path) = ($1, $2);
  return $prefix unless length $path;

  $path = $self->validate_local_abs_uri_frag($path);
  return '' unless $path;

  return "$prefix$path";
}

=item validate_local_abs_uri_frag ( URIFRAG )

Validates a local absolute URI fragment, such as C</img/foo.png>.  Allows
a query string.  The empty string is considered to be a valid URI fragment.

=cut

sub validate_local_abs_uri_frag {
  my ($self, $frag) = @_;

  $frag =~ m< ^ ( (?: \.* /  [\w\-.!~*'(|);/\@+\$,%#&=]* )?
                  (?: \?     [\w\-.!~*'(|);/\@+\$,%#&=]* )?
                )
              $
           >x ? $1 : '';
}

=item validate_url ( URL )
```

```
Validates a URL, which can be either an absolute URL or a local absolute
URI fragment.

=cut

sub validate_url {
  my ($self, $url) = @_;

  if ($url =~ m#://#) {
    $self->validate_abs_url($url);
  }
  else {
    $self->validate_local_abs_uri_frag($url);
  }
}

=item validate_email ( EMAIL )

Validates an email address.

=cut

sub validate_email {
  my ($self, $email) = @_;

  $email =~ /^([a-z0-9_\-\.\*\+\=]{1,100})\@([^@]{2,100})$/i or return 0;
  my ($user, $host) = ($1, $2);

  return 0 if $host =~ m#^\.|\.$|\.\.#;

  if ($host =~ m#^\[\d+\.\d+\.\d+\.\d+\]$# or $host =~ /^[a-z0-9\-\.]+$/i ) {
     return "$user\@$host";
   }
   else {
     return 0;
  }
}

=item validate_name ( name )

Validates a real name, i.e. an email address comment field.

=cut

sub validate_name {
  my ($self, $name) = @_;

  $name =~ tr# a-zA-Z0-9_\-,./'\200-\377# #cs;
  $name = substr $name, 0, 128;

  $name =~ m#^([ a-zA-Z0-9_\-,./'\200-\377]*)$# or die "failed on [$name]";
  return $1;
}

=item validate_html_color ( COLOR )

Validates an HTML color, either as a named color or as RGB values in hex.

=cut

sub validate_html_color {
  my ($self, $color) = @_;

  $color =~ /^(#[0-9a-z]{6}|[\w\-]{2,50})$/i ? $1 : '';
}

=back
```

```
=head1 SEE ALSO

L<CGI::NMS::Script>

=head1 MAINTAINERS

The NMS project, E<lt>http://nms-cgi.sourceforge.net/E<gt>

To request support or report bugs, please email
E<lt>nms-cgi-support@lists.sourceforge.netE<gt>

=head1 COPYRIGHT

Copyright 2003 London Perl Mongers, All rights reserved

=head1 LICENSE

This module is free software; you are free to redistribute it
and/or modify it under the same terms as Perl itself.

=cut

1;


END_INLINED_CGI_NMS_Validator
  $INC{'CGI/NMS/Validator.pm'} = 1;
}


unless (eval {local $SIG{__DIE__} ; require CGI::NMS::Script::FormMail}) {
  eval <<'END_INLINED_CGI_NMS_Script_FormMail' or die $@;
package CGI::NMS::Script::FormMail;
use strict;

use vars qw($VERSION);
$VERSION = substr q$Revision: 1.9 $, 10, -1;

use Socket;  # for the inet_aton()

use CGI::NMS::Script;
use CGI::NMS::Validator;
use CGI::NMS::Mailer::ByScheme;
use base qw(CGI::NMS::Script CGI::NMS::Validator);

=head1 NAME

CGI::NMS::Script::FormMail - FormMail CGI script

=head1 SYNOPSIS

  #!/usr/bin/perl -wT
  use strict;

  use base qw(CGI::NMS::Script::FormMail);

  use vars qw($script);
  BEGIN {
    $script = __PACKAGE__->new(
      'DEBUGGING'     => 1,
      'postmaster'    => 'me@my.domain',
      'allow_mail_to' => 'me@my.domain',
    );
  }

  $script->request;
```

```
=head1 DESCRIPTION

This module implements the NMS plugin replacement for Matt Wright's
FormMail.pl CGI script.

=head1 CONFIGURATION SETTINGS

As well as the generic NMS script configuration settings described in
L<CGI::NMS::Script>, the FormMail constructor recognizes the following
configuration settings:

=over

=item C<allow_empty_ref>

Some web proxies and office firewalls may strip certain headers from the
HTTP request that is sent by a browser.  Among these is the HTTP_REFERER
that FormMail uses as an additional check of the requests validity - this
will cause the program to fail with a 'bad referer' message even though the
configuration seems fine.

In these cases, setting this configuration setting to 1 will stop the
program from complaining about requests where no referer header was sent
while leaving the rest of the security features intact.

Default: 1

=item C<max_recipients>

The maximum number of e-mail addresses that any single form should be
allowed to send copies of the e-mail to.  If none of your forms send
e-mail to more than one recipient, then we recommend that you improve
the security of FormMail by reducing this value to 1.  Setting this
configuration setting to 0 removes all limits on the number of recipients
of each e-mail.

Default: 5

=item C<mailprog>

The system command that the script should invoke to send an outgoing email.
This should be the full path to a program that will read a message from
STDIN and determine the list of message recipients from the message headers.
Any switches that the program requires should be provided here.

For example:

  'mailprog' => '/usr/lib/sendmail -oi -t',

An SMTP relay can be specified instead of a sendmail compatible mail program,
using the prefix C<SMTP:>, for example:

  'mailprog' => 'SMTP:mailhost.your.domain',

Default: C<'/usr/lib/sendmail -oi -t'>

=item C<postmaster>

The envelope sender address to use for all emails sent by the script.

Default: ''

=item C<referers>

This configuration setting must be an array reference, holding a list
of names and/or IP address of systems that will host forms that refer
to this FormMail.  An empty array here turns off all referer checking.
```

Default: []

=item C<allow_mail_to>

This configuration setting must be an array reference.

A list of the email addresses that FormMail can send email to. The elements of this list can be either simple email addresses (like 'you@your.domain') or domain names (like 'your.domain'). If it's a domain name then any address at that domain will be allowed.

Default: []

=item C<recipients>

This configuration setting must be an array reference.

A list of Perl regular expression patterns that determine who the script will allow mail to be sent to in addition to those set in C<allow_mail_to>. This is present only for compatibility with the original FormMail script. We strongly advise against having anything in C<recipients> as it's easy to make a mistake with the regular expression syntax and turn your FormMail into an open SPAM relay.

Default: []

=item C<recipient_alias>

This configuration setting must be a hash reference.

A hash for predefining a list of recipients in the script, and then choosing between them using the recipient form field, while keeping all the email addresses out of the HTML so that they don't get collected by address harvesters and sent junk email.

For example, suppose you have three forms on your site, and you want each to submit to a different email address and you want to keep the addresses hidden. You might set up C<recipient_alias> like this:

```
  %recipient_alias = (
    '1' => 'one@your.domain',
    '2' => 'two@your.domain',
    '3' => 'three@your.domain',
  );
```

In the HTML form that should submit to the recipient C<two@your.domain>, you would then set the recipient with:

```
  <input type="hidden" name="recipient" value="2" />
```

Default: {}

=item C<valid_ENV>

This configuration setting must be an array reference.

A list of all the environment variables that you want to be able to include in the email.

Default: ['REMOTE_HOST','REMOTE_ADDR','REMOTE_USER','HTTP_USER_AGENT']

=item C<date_fmt>

The format that the date will be displayed in, as a string suitable for passing to strftime().

Default: '%A, %B %d, %Y at %H:%M:%S'

=item C<date_offset>

The empty string to use local time for the date, or an offset from GMT in hours to fix the timezone independent of the server's locale settings.

Default: ''

=item C<no_content>

If this is set to 1 then rather than returning the HTML confirmation page or doing a redirect the script will output a header that indicates that no content will be returned and that the submitted form should not be replaced. This should be used carefully as an unwitting visitor may click the submit button several times thinking that nothing has happened.

Default: 0

=item C<double_spacing>

If this is set to 1 then a blank line is printed after each form value in the e-mail. Change this value to 0 if you want the e-mail to be more compact.

Default: 1

=item C<join_string>

If an input occurs multiple times, the values are joined to make a single string value. The value of this configuration setting is inserted between each value when they are joined.

Default: ' '

=item C<wrap_text>

If this is set to 1 then the content of any long text fields will be wrapped at around 72 columns in the e-mail which is sent. The way that this is done is controlled by the C<wrap_style> configuration setting.

Default: 0

=item C<wrap_style>

If C<wrap_text> is set to 1 then if this is set to 1 then the text will be wrapped in such a way that the left margin of the text is lined up with the beginning of the text after the description of the field - that is to say it is indented by the length of the field name plus 2.

If it is set to 2 then the subsequent lines of the text will not be indented at all and will be flush with the start of the lines. The choice of style is really a matter of taste although you might find that style 1 does not work particularly well if your e-mail client uses a proportional font where the spaces of the indent might be smaller than the characters in the field name.

Default: 1

=item C<force_config_*>

Configuration settings of this form can be used to fix configuration settings that would normally be set in hidden form fields. For example, to force the email subject to be "Foo" irrespective of what's in the C<subject> form field, you would set:

```
  'force_config_subject' => 'Foo',
```

Default: none set

```
=item C<include_config_*>

Configuration settings of this form can be used to treat particular
configuration inputs as normal data inputs as well as honoring their
special meaning.  For example, a user might use C<include_config_email>
to include the email address as a regular input as well as using it in
the email header.

Default: none set

=back

=head1 COMPILE TIME METHODS

These methods are invoked at CGI script compile time only, so long as
the new() call is placed inside a BEGIN block as shown above.

=over

=item default_configuration ()

Returns the default values for the configuration passed to the new()
method, as a key,value,key,value list.

=cut

sub default_configuration {
  return (
    allow_empty_ref        => 1,
    max_recipients         => 5,
    mailprog               => '/usr/lib/sendmail -oi -t',
    postmaster             => '',
    referers               => [],
    allow_mail_to          => [],
    recipients             => [],
    recipient_alias        => {},
    valid_ENV              => [qw(REMOTE_HOST REMOTE_ADDR REMOTE_USER
HTTP_USER_AGENT)],
    date_fmt               => '%A, %B %d, %Y at %H:%M:%S',
    date_offset            => '',
    no_content             => 0,
    double_spacing         => 1,
    join_string            => ' ',
    wrap_text              => 0,
    wrap_style             => 1,
  );
}

=item init ()

Invoked from the new() method inherited from L<CGI::NMS::Script>,
this method performs FormMail specific initialization of the script
object.

=cut

sub init {
  my ($self) = @_;

  if ($self->{CFG}{wrap_text}) {
    require Text::Wrap;
    import  Text::Wrap;
  }

  $self->{Valid_Env} = {  map {$_=>1} @{ $self->{CFG}{valid_ENV} }  };

  $self->init_allowed_address_list;
```

```
  $self->{Mailer} = CGI::NMS::Mailer::ByScheme->new($self->{CFG}{mailprog});
}

=item init_allowed_address_list ()

Invoked from init(), this method sets up a hash with a key for each
allowed recipient email address as C<Allow_Mail> and a hash with a
key for each domain at which any address is allowed as C<Allow_Domain>.

=cut

sub init_allowed_address_list {
  my ($self) = @_;

  my @allow_mail = ();
  my @allow_domain = ();

  foreach my $m (@{ $self->{CFG}{allow_mail_to} }) {
    if ($m =~ /\@/) {
      push @allow_mail, $m;
    }
    else {
      push @allow_domain, $m;
    }
  }

  my @alias_targets = split /\s*,\s*/, join ',', values %{
$self->{CFG}{recipient_alias} };
  push @allow_mail, grep /\@/, @alias_targets;

  # The username part of email addresses should be case sensitive, but the
  # domain name part should not.  Map all domain names to lower case for
  # comparison.
  my (%allow_mail, %allow_domain);
  foreach my $m (@allow_mail) {
    $m =~ /^([^@]+)\@([^@]+)$/ or die "internal failure [$m]";
    $m = $1 . '@' . lc $2;
    $allow_mail{$m} = 1;
  }
  foreach my $m (@allow_domain) {
    $m = lc $m;
    $allow_domain{$m} = 1;
  }

  $self->{Allow_Mail}   = \%allow_mail;
  $self->{Allow_Domain} = \%allow_domain;
}

=back

=head1 RUN TIME METHODS

These methods are invoked at script run time, as a result of the call
to the request() method inherited from L<CGI::NMS::Script>.

=over

=item handle_request ()

Handles the core of a single CGI request, outputting the HTML success
or error page or redirect header and sending emails.

Dies on error.

=cut

sub handle_request {
  my ($self) = @_;
```

```
  $self->{Hide_Recipient} = 0;

  my $referer = $self->cgi_object->referer;
  unless ($self->referer_is_ok($referer)) {
    $self->referer_error_page;
    return;
  }

  $self->check_method_is_post    or return;

  $self->parse_form;

  $self->check_recipients( $self->get_recipients ) or return;

  my @missing = $self->get_missing_fields;
  if (scalar @missing) {
    $self->missing_fields_output(@missing);
    return;
  }

  my $date     = $self->date_string;
  my $email    = $self->get_user_email;
  my $name = $self->get_user_name;

  $self->send_main_email($date, $email, $name);
  $self->send_conf_email($date, $email, $name);

  $self->success_page($date);
}

=item date_string ()

Returns a string giving the current date and time, in the configured
format.

=cut

sub date_string {
  my ($self) = @_;

  return $self->format_date( $self->{CFG}{date_fmt},
                             $self->{CFG}{date_offset} );
}

=item referer_is_ok ( REFERER )

Returns true if the referer is OK, false otherwise.

=cut

sub referer_is_ok {
  my ($self, $referer) = @_;

  unless ($referer) {
    return ($self->{CFG}{allow_empty_ref} ? 1 : 0);
  }

  if ($referer =~ m!^https?://([^/]*\@)?([\w\-\.]+)!i) {
    my $refhost = $2;
    return $self->refering_host_is_ok($refhost);
  }
  else {
    return 0;
  }
}

=item refering_host_is_ok ( REFERING_HOST )
```

```
Returns true if the host name REFERING_HOST is on the list of allowed
referers, or resolves to an allowed IP address.

=cut

sub refering_host_is_ok {
  my ($self, $refhost) = @_;

  my @allow = @{ $self->{CFG}{referers} };
  return 1 unless scalar @allow;

  foreach my $test_ref (@allow) {
    if ($refhost =~ m|\Q$test_ref\E$|i) {
      return 1;
    }
  }

  my $ref_ip = inet_aton($refhost) or return 0;
  foreach my $test_ref (@allow) {
    next unless $test_ref =~ /^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$/;

    my $test_ref_ip = inet_aton($test_ref) or next;
    if ($ref_ip eq $test_ref_ip) {
      return 1;
    }
  }
}

=item referer_error_page ()

Invoked if the referer is bad, this method outputs an error page
describing the problem with the referer.

=cut

sub referer_error_page {
  my ($self) = @_;

  my $referer = $self->cgi_object->referer || '';
  my $escaped_referer = $self->escape_html($referer);

  if ( $referer =~ m|^https?://([\w\.\-]+)|i) {
    my $host = $1;
    $self->error_page( 'Bad Referrer - Access Denied', <<END );
<p>
  The form attempting to use this script resides at <tt>$escaped_referer</tt>,
  which is not allowed to access this program.
</p>
<p>
  If you are attempting to configure FormMail to run with this form,
  you need to add the following to \@referers, explained in detail in the
  README file.
</p>
<p>
  Add <tt>'$host'</tt> to your <tt><b>\@referers</b></tt> array.
</p>
END
  }
  elsif (length $referer) {
    $self->error_page( 'Malformed Referrer - Access Denied', <<END );
<p>
  The referrer value <tt>$escaped_referer</tt> cannot be parsed, so
  it is not possible to check that the referring page is allowed to
  access this program.
</p>
END
  }
```

```
  else {
    $self->error_page( 'Missing Referrer - Access Denied', <<END );
<p>
  Your browser did not send a <tt>Referer</tt> header with this
  request, so it is not possible to check that the referring page
  is allowed to access this program.
</p>
END
  }
}

=item check_method_is_post ()

Unless the C<secure> configuration setting is false, this method checks
that the request method is POST.  Returns true if OK, otherwise outputs
an error page and returns false.

=cut

sub check_method_is_post {
  my ($self) = @_;

  return 1 unless $self->{CFG}{secure};

  my $method = $self->cgi_object->request_method || '';
  if ($method ne 'POST') {
    $self->error_page( 'Error: GET request', <<END );
<p>
  The HTML form fails to specify the POST method, so it would not
  be correct for this script to take any action in response to
  your request.
</p>
<p>
  If you are attempting to configure this form to run with FormMail,
  you need to set the request method to POST in the opening form tag,
  like this:
  <tt>&lt;form action=&quot;/cgi-bin/FormMail.pl&quot;
method=&quot;post&quot;&gt;</tt>
</p>
END
    return 0;
  }
  else {
    return 1;
  }
}

=item parse_form ()

Parses the HTML form, storing the results in various fields in the
C<FormMail> object, as follows:

=over

=item C<FormConfig>

A hash holding the values of the configuration inputs, such as
C<recipient> and C<subject>.

=item C<Form>

A hash holding the values of inputs other than configuration inputs.

=item C<Field_Order>

An array giving the set and order of fields to be included in the
email and on the success page.
```

```
=back

=cut

sub parse_form {
  my ($self) = @_;

  $self->{FormConfig} = { map {$_=>''} $self->configuration_form_fields };
  $self->{Field_Order} = [];
  $self->{Form} = {};

  foreach my $p ($self->cgi_object->param()) {
    if (exists $self->{FormConfig}{$p}) {
      $self->parse_config_form_input($p);
    }
    else {
      $self->parse_nonconfig_form_input($p);
    }
  }

  $self->substitute_forced_config_values;

  $self->expand_list_config_items;

  $self->sort_field_order;
  $self->remove_blank_fields;
}

=item configuration_form_fields ()

Returns a list of the names of the form fields which are used
to configure formmail rather than to provide user input, such
as C<subject> and C<recipient>.  The specially treated C<email>
and C<name> fields are included in this list.

=cut

sub configuration_form_fields {
  qw(
    recipient
    subject
    email
    name
    redirect
    bgcolor
    background
    link_color
    vlink_color
    text_color
    alink_color
    title
    sort
    print_config
    required
    env_report
    return_link_title
    return_link_url
    print_blank_fields
    missing_fields_redirect
  );
}

=item parse_config_form_input ( NAME )

Deals with the configuration form input NAME, incorporating it into
the C<FormConfig> field in the blessed hash.

=cut
```

```
sub parse_config_form_input {
  my ($self, $name) = @_;

  my $val = $self->strip_nonprint($self->cgi_object->param($name));
  if ($name =~ /return_link_url|redirect$/) {
    $val = $self->validate_url($val);
  }
  $self->{FormConfig}{$name} = $val;
  unless ($self->{CFG}{emulate_matts_code}) {
    $self->{Form}{$name} = $val;
    if ( $self->{CFG}{"include_config_$name"} ) {
      push @{ $self->{Field_Order} }, $name;
    }
  }
}

=item parse_nonconfig_form_input ( NAME )

Deals with the non-configuration form input NAME, incorporating it into
the C<Form> and C<Field_Order> fields in the blessed hash.

=cut

sub parse_nonconfig_form_input {
  my ($self, $name) = @_;

  my @vals = map {$self->strip_nonprint($_)} $self->cgi_object->param($name);
  my $key = $self->strip_nonprint($name);
  $self->{Form}{$key} = join $self->{CFG}{join_string}, @vals;
  push @{ $self->{Field_Order} }, $key;
}

=item expand_list_config_items ()

Converts the form configuration values C<required>, C<env_report> and
C<print_config> from strings of comma separated values to arrays, and
removes anything not in the C<valid_ENV> configuration setting from
C<env_report>.

=cut

sub expand_list_config_items {
  my ($self) = @_;

  foreach my $p (qw(required env_report print_config)) {
    if ($self->{FormConfig}{$p}) {
      $self->{FormConfig}{$p} = [split(/\s*,\s*/, $self->{FormConfig}{$p})];
    }
    else {
      $self->{FormConfig}{$p} = [];
    }
  }

  $self->{FormConfig}{env_report} =
     [ grep { $self->{Valid_Env}{$_} } @{ $self->{FormConfig}{env_report} } ];
}

=item substitute_forced_config_values ()

Replaces form configuration values for which there is a forced value
configuration setting with the forced value.  Sets C<Hide_Recipient>
true if the recipient config value is forced.

=cut

sub substitute_forced_config_values {
  my ($self) = @_;
```

```
  foreach my $k (keys %{ $self->{FormConfig} }) {
    if (exists $self->{CFG}{"force_config_$k"}) {
      $self->{FormConfig}{$k} = $self->{CFG}{"force_config_$k"};
      $self->{Hide_Recipient} = 1 if $k eq 'recipient';
    }
  }
}

=item sort_field_order ()

Modifies the C<Field_Order> field in the blessed hash according to
the sorting scheme set in the C<sort> form configuration, if any.

=cut

sub sort_field_order {
  my ($self) = @_;

  my $sort = $self->{FormConfig}{'sort'};
  if (defined $sort) {
    if ($sort eq 'alphabetic') {
      $self->{Field_Order} = [ sort @{ $self->{Field_Order} } ];
    }
    elsif ($sort =~ /^\s*order:\s*(.*)$/s) {
      $self->{Field_Order} = [ split /\s*,\s*/, $1 ];
    }
  }
}

=item remove_blank_fields ()

Removes the names of blank or missing fields from the C<Field_Order> array
unless the C<print_blank_fields> form configuration value is true.

=cut

sub remove_blank_fields {
  my ($self) = @_;

  return if $self->{FormConfig}{print_blank_fields};

  $self->{Field_Order} = [
    grep { defined $self->{Form}{$_} and $self->{Form}{$_} !~ /^\s*$/ }
    @{ $self->{Field_Order} }
  ];
}

=item get_recipients ()

Determines the list of configured recipients from the form inputs and the
C<recipient_alias> configuration setting, and returns them as a list.

Sets the C<Hide_Recipient> field in the blessed hash to a true value if
one or more of the recipients were aliased and so should be hidden to
foil address harvesters.

=cut

sub get_recipients {
  my ($self) = @_;

  my $recipient = $self->{FormConfig}{recipient};
  my @recipients;

  if (length $recipient) {
    foreach my $r (split /\s*,\s*/, $recipient) {
      if (exists $self->{CFG}{recipient_alias}{$r}) {
```

```
        push @recipients, split /\s*,\s*/, $self->{CFG}{recipient_alias}{$r};
        $self->{Hide_Recipient} = 1;
      }
      else {
        push @recipients, $r;
      }
    }
  }
  else {
    return $self->default_recipients;
  }

  return @recipients;
}

=item default_recipients ()

Invoked from get_recipients if no C<recipient> input is found, this method
returns the default recipient list.  The default recipient is the first email
address listed in the C<allow_mail_to> configuration setting, if any.

=cut

sub default_recipients {
  my ($self) = @_;

  my @allow = grep {/\@/} @{ $self->{CFG}{allow_mail_to} };
  if (scalar @allow > 0 and not $self->{CFG}{emulate_matts_code}) {
    $self->{Hide_Recipient} = 1;
    return ($allow[0]);
  }
  else {
    return ();
  }
}

=item check_recipients ( @RECIPIENTS )

Works through the array of recipients passed in and discards any the the script
is not configured to allow, storing the list of valid recipients in the
C<Recipients> field in the blessed hash.

Returns true if at least one (and not too many) valid recipients are found,
otherwise outputs an error page and returns false.

=cut

sub check_recipients {
  my ($self, @recipients) = @_;

  my @valid = grep { $self->recipient_is_ok($_) } @recipients;
  $self->{Recipients} = \@valid;

  if (scalar(@valid) == 0) {
    $self->bad_recipient_error_page;
    return 0;
  }
  elsif ($self->{CFG}{max_recipients} and scalar(@valid) >
$self->{CFG}{max_recipients}) {
    $self->too_many_recipients_error_page;
    return 0;
  }
  else {
    return 1;
  }
}

=item recipient_is_ok ( RECIPIENT )
```

```
Returns true if the recipient RECIPIENT should be allowed, false otherwise.

=cut

sub recipient_is_ok {
  my ($self, $recipient) = @_;

  return 0 unless $self->validate_email($recipient);

  $recipient =~ /^(.+)\@([^@]+)$/m or die "regex failure [$recipient]";
  my ($user, $host) = ($1, lc $2);
  return 1 if exists $self->{Allow_Domain}{$host};
  return 1 if exists $self->{Allow_Mail}{"$user\@$host"};

  foreach my $r (@{ $self->{CFG}{recipients} }) {
    return 1 if $recipient =~ /(?:$r)$/;
    return 1 if $self->{CFG}{emulate_matts_code} and $recipient =~ /(?:$r)$/i;
  }

  return 0;
}

=item bad_recipient_error_page ()

Outputs the error page for a bad or missing recipient.

=cut

sub bad_recipient_error_page {
  my ($self) = @_;

  my $errhtml = <<END;
<p>
  There was no recipient or an invalid recipient specified in the
  data sent to FormMail. Please make sure you have filled in the
  <tt>recipient</tt> form field with an e-mail address that has
  been configured in <tt>\@recipients</tt> or <tt>\@allow_mail_to</tt>.
  More information on filling in <tt>recipient/allow_mail_to</tt>
  form fields and variables can be found in the README file.
</p>
END

  unless ($self->{CFG}{force_config_recipient}) {
    my $esc_rec = $self->escape_html( $self->{FormConfig}{recipient} );
    $errhtml .= <<END;
<hr size="1" />
<p>
 The recipient was: [ $esc_rec ]
</p>
END
  }

  $self->error_page( 'Error: Bad or Missing Recipient', $errhtml );
}

=item too_many_recipients_error_page ()

Outputs the error page for too many recipients configured.

=cut

sub too_many_recipients_error_page {
  my ($self) = @_;

  $self->error_page( 'Error: Too many Recipients', <<END );
<p>
  The number of recipients configured in the form exceeds the
```

```
  maximum number of recipients configured in the script.  If
  you are attempting to configure FormMail to run with this form
  then you will need to increase the <tt>\$max_recipients</tt>
  configuration setting in the script.
</p>
END
}

=item get_missing_fields ()

Returns a list of the names of the required fields that have not been
filled in acceptably, each one possibly annotated with details of the
problem with the way the field was filled in.

=cut

sub get_missing_fields {
  my ($self) = @_;

  my @missing = ();

  foreach my $f (@{ $self->{FormConfig}{required} }) {
    if ($f eq 'email') {
      unless ( $self->get_user_email =~ /\@/ ) {
        push @missing, 'email (must be a valid email address)';
      }
    }
    elsif ($f eq 'name') {
      unless ( length $self->get_user_name ) {
        push @missing, 'name';
      }
    }
    else {
      my $val = $self->{Form}{$f};
      if (! defined $val or $val =~ /^\s*$/) {
        push @missing, $f;
      }
    }
  }

  return @missing;
}

=item missing_fields_output ( @MISSING )

Produces the configured output (an error page or a redirect) for the
case when there are missing fields.  Takes a list of the missing
fields as arguments.

=cut

sub missing_fields_output {
  my ($self, @missing) = @_;

  if ( $self->{FormConfig}{'missing_fields_redirect'} ) {
    print
$self->cgi_object->redirect($self->{FormConfig}{'missing_fields_redirect'});
  }
  else {
    my $missing_field_list = join '',
                             map { '<li>' . $self->escape_html($_) . "</li>\n" }
                             @missing;
    $self->error_page( 'Error: Blank Fields', <<END );
<p>
    The following fields were left blank in your submission form:
</p>
<div class="c2">
   <ul>
```

```
     $missing_field_list
    </ul>
</div>
<p>
    These fields must be filled in before you can successfully
    submit the form.
</p>
<p>
    Please use your back button to return to the form and
    try again.
</p>
END
  }
}

=item get_user_email ()

Returns the user's email address if they entered a valid one in the C<email>
form field, otherwise returns the string C<nobody>.

=cut

sub get_user_email {
  my ($self) = @_;

  my $email = $self->{FormConfig}{email};
  $email = $self->validate_email($email);
  $email = 'nobody' unless $email;

  return $email;
}

=item get_user_name ()

Returns the user's real name, as entered in the C<name> form field.

=cut

sub get_user_name {
  my ($self) = @_;

  my $name = $self->{FormConfig}{name};
  if (defined $name) {
    $name = $self->validate_name($name);
  } else {
    $name = '';
  }

  return $name;
}

=item send_main_email ( DATE, EMAIL, name )

Sends the main email.  DATE is a date string, EMAIL is the
user's email address if they entered a valid one and name
is the user's real name if entered.

=cut

sub send_main_email {
  my ($self, $date, $email, $name) = @_;

  my $mailer = $self->mailer;
  $mailer->newmail($self->name_and_version, $self->{CFG}{postmaster}, @{
$self->{Recipients} });

  $self->send_main_email_header($email, $name);
  $mailer->print("\n");
```

```
  $self->send_main_email_body_header($date);

  $self->send_main_email_print_config;

  $self->send_main_email_fields;

  $self->send_main_email_footer;

  $mailer->endmail;
}

=item send_main_email_header ( EMAIL, name )

Sends the email header for the main email, not including the terminating
blank line.

=cut

sub send_main_email_header {
  my ($self, $email, $name) = @_;

  my $subject = $self->{FormConfig}{subject} || 'WWW Form Submission';
  if ($self->{CFG}{secure}) {
    $subject = substr($subject, 0, 256);
  }
  $subject =~ s#[\r\n\t]+# #g;

  my $to = join ',', @{ $self->{Recipients} };
  my $from = (length $name ? "$email ($name)" : $email);

  $self->mailer->print(<<END);
X-Mailer: ${\( $self->name_and_version )}
To: $to
From: $from
Subject: $subject
END
}

=item send_main_email_body_header ( DATE )

Invoked after the blank line to terminate the header is sent, this method
outputs the header of the email body.

=cut

sub send_main_email_body_header {
  my ($self, $date) = @_;

  my $dashes = '-' x 75;
  $dashes .= "\n\n" if $self->{CFG}{double_spacing};

  $self->mailer->print(<<END);
$date

$self->{FormConfig}{name}
$self->{FormConfig}{email}
$dashes
END
}

=item send_main_email_print_config ()

If the C<print_config> form configuration field is set, outputs the configured
config values to the email.

=cut
```

```
sub send_main_email_print_config {
  my ($self) = @_;

  if ($self->{FormConfig}{print_config}) {
    foreach my $cfg (@{ $self->{FormConfig}{print_config} }) {
      if ($self->{FormConfig}{$cfg}) {
        $self->mailer->print("$cfg: $self->{FormConfig}{$cfg}\n");
        $self->mailer->print("\n") if $self->{CFG}{double_spacing};
      }
    }
  }
}

=item send_main_email_fields ()

Outputs the form fields to the email body.

=cut

sub send_main_email_fields {
  my ($self) = @_;

  foreach my $f (@{ $self->{Field_Order} }) {
    my $val = (defined $self->{Form}{$f} ? $self->{Form}{$f} : '');

    $self->send_main_email_field($f, $val);
  }
}

=item send_main_email_field ( NAME, VALUE )

Outputs a single form field to the email body.

=cut

sub send_main_email_field {
  my ($self, $name, $value) = @_;

  my ($prefix, $line) = $self->build_main_email_field($name, $value);

  my $nl = ($self->{CFG}{double_spacing} ? "\n\n" : "\n");

  if ($self->{CFG}{wrap_text} and length("$prefix$line") >
$self->email_wrap_columns) {
    $self->mailer->print( $self->wrap_field_for_email($prefix, $line) . $nl );
  }
  else {
    $self->mailer->print("$prefix$line$nl");
  }
}

=item build_main_email_field ( NAME, VALUE )

Generates the email body text for a single form input, and returns
it as a two element list of prefix and remainder of line.  The return
value is split into a prefix and remainder of line because the text
wrapping code may need to indent the wrapped line to the length of the
prefix.

=cut

sub build_main_email_field {
  my ($self, $name, $value) = @_;

  return ("$name: ", $value);
}

=item wrap_field_for_email ( PREFIX, LINE )
```

```
Takes the prefix and rest of line of a field as arguments, and returns them
as a text wrapped paragraph suitable for inclusion in the main email.

=cut

sub wrap_field_for_email {
  my ($self, $prefix, $value) = @_;

  my $subs_indent = '';
  $subs_indent = ' ' x length($prefix) if $self->{CFG}{wrap_style} == 1;

  local $Text::Wrap::columns = $self->email_wrap_columns;

  # Some early versions of Text::Wrap will die on very long words, if that
  # happens we fall back to no wrapping.
  my $wrapped;
  eval { local $SIG{__DIE__} ; $wrapped = wrap($prefix,$subs_indent,$value) };
  return ($@ ? "$prefix$value" : $wrapped);
}

=item email_wrap_columns ()

Returns the number of columns to which the email should be wrapped if the
text wrapping option is in use.

=cut

sub email_wrap_columns { 72; }

=item send_main_email_footer ()

Sends the footer of the main email body, including any environment variables
listed in the C<env_report> configuration form field.

=cut

sub send_main_email_footer {
  my ($self) = @_;

  my $dashes = '-' x 75;
  $self->mailer->print("$dashes\n\n");

  foreach my $e (@{ $self->{FormConfig}{env_report}}) {
    if ($ENV{$e}) {
      $self->mailer->print("$e: " . $self->strip_nonprint($ENV{$e}) . "\n");
    }
  }
}

=item send_conf_email ( DATE, EMAIL, name )

Sends a confirmation email back to the user, if configured to do so and the
user entered a valid email addresses.

=cut

sub send_conf_email {
  my ($self, $date, $email, $name) = @_;

  if ( $self->{CFG}{send_confirmation_mail} and $email =~ /\@/ ) {
    my $to = (length $name ? "$email ($name)" : $email);
    $self->mailer->newmail("NMS FormMail.pm v$VERSION",
$self->{CFG}{postmaster}, $email);
    $self->mailer->print("To: $to\n$self->{CFG}{confirmation_text}");
    $self->mailer->endmail;
  }
}
```

```
=item success_page ()

Outputs the HTML success page (or redirect if configured) after the email
has been successfully sent.

=cut

sub success_page {
  my ($self, $date) = @_;

  if ($self->{FormConfig}{'redirect'}) {
    print $self->cgi_object->redirect( $self->{FormConfig}{'redirect'} );
  }
  else {
    $self->output_cgi_html_header;
    $self->success_page_html_preamble($date);
    $self->success_page_fields;
    $self->success_page_footer;
  }
}

=item success_page_html_preamble ( DATE )

Outputs the start of the HTML for the success page, not including the
standard HTML headers dealt with by output_cgi_html_header().

=cut

sub success_page_html_preamble {
  my ($self, $date) = @_;

  my $title = $self->escape_html( $self->{FormConfig}{'title'} || 'Thank you for
booking with Pichi Richi Camel Tours' );
  my $torecipient = 'to ' .
$self->escape_html($self->{FormConfig}{'recipient'});
  $torecipient = '' if $self->{Hide_Recipient};
  my $attr = $self->body_attributes;

    print <<END;
  <head>
     <title>$title</title>
END

    $self->output_style_element;

    print <<END;
     <style>
       h1.title {
                  text-align : center;
                }
     </style>
  </head>
  <body $attr>
    <h1 class="title">$title</h1>
    <p><h3>Quorn South Australia</h3></p>
    <p>Thank you for your booking - we will respond within 24 hours</p>
    <p><h3>Please PRINT this page for your records</h3></p>
    <p>Below is what you submitted $torecipient on $date</p>
    <p><hr size="1" width="75%" /></p>
END
}

=item success_page_fields ()

Outputs success page HTML output for each input field.

=cut
```

```
sub success_page_fields {
  my ($self) = @_;

  foreach my $f (@{ $self->{Field_Order} }) {
    my $val = (defined $self->{Form}{$f} ? $self->{Form}{$f} : '');
    $self->success_page_field( $self->escape_html($f), $self->escape_html($val)
);
  }
}

=item success_page_field ( NAME, VALUE ) {

Outputs success page HTML for a single input field.  NAME and VALUE
are the HTML escaped field name and value.

=cut

sub success_page_field {
  my ($self, $name, $value) = @_;

  print "<p><b>$name:</b> $value</p>\n";
}

=item success_page_footer ()

Outputs the footer of the success page, including the return link if
configured.

=cut

sub success_page_footer {
  my ($self) = @_;

  print qq{<p><hr size="1" width="75%" /></p>\n};
  $self->success_page_return_link;
  print <<END;
        <p align="center">
           <font size="-1">
             <a href="http://www.hawkersa.info/booksafari.htm">Back to Enquiry
Form</a>

           </font>
        </p>
        <p align="center">
           <font size="-1">
             <a href="http://nms-cgi.sourceforge.net/">FormMail</a>
             &copy; 2001  London Perl Mongers
           </font>
        </p>
        </body>
       </html>
END
}

=item success_page_return_link ()

Outputs the success page return link if any is configured.

=cut

sub success_page_return_link {
  my ($self) = @_;

  if ($self->{FormConfig}{return_link_url} and
$self->{FormConfig}{return_link_title}) {
    print "<ul>\n";
    print '<li><a href="',
```

```
$self->escape_html($self->{FormConfig}{return_link_url}),
      '">', $self->escape_html($self->{FormConfig}{return_link_title}),
"</a>\n";
    print "</li>\n</ul>\n";
  }
}

=item body_attributes ()

Gets the body attributes for the success page from the form
configuration, and returns the string that should go inside
the C<body> tag.

=cut

sub body_attributes {
  my ($self) = @_;

  my %attrs = (bgcolor     => 'bgcolor',
               background  => 'background',
               link_color  => 'link',
               vlink_color => 'vlink',
               alink_color => 'alink',
               text_color  => 'text');

  my $attr = '';

  foreach my $at (keys %attrs) {
    my $val = $self->{FormConfig}{$at};
    next unless $val;
    if ($at =~ /color$/) {
      $val = $self->validate_html_color($val);
    }
    elsif ($at eq 'background') {
      $val = $self->validate_url($val);
    }
    else {
      die "no check defined for body attribute [$at]";
    }
    $attr .= qq( $attrs{$at}=") . $self->escape_html($val) . '"' if $val;
  }

  return $attr;
}

=item error_page( TITLE, ERROR_BODY )

Outputs a FormMail error page, giving the HTML document the title
TITLE and displaying the HTML error message ERROR_BODY.

=cut

sub error_page {
  my ($self, $title, $error_body) = @_;

  $self->output_cgi_html_header;

  my $etitle = $self->escape_html($title);
  print <<END;
  <head>
    <title>$etitle</title>
END

  $self->output_style_element;

  print <<END;
    <style type="text/css">
    <!--
```

```
       body {
              background-color: #FFFFFF;
              color: #000000;
             }
       p.c2 {
              font-size: 80%;
              text-align: center;
            }
       th.c1 {
               text-align: center;
               font-size: 143%;
             }
       p.c3 {font-size: 80%; text-align: center}
       div.c2 {margin-left: 2em}
     -->
    </style>
  </head>
  <body>
    <table border="0" width="600" bgcolor="#9C9C9C" summary="">
      <tr bgcolor="#9C9C9C">
        <th class="c1">$etitle</th>
      </tr>
      <tr bgcolor="#CFCFCF">
        <td>
          $error_body
          <hr size="1" />
          <p class="c3">
            <a href="http://nms-cgi.sourceforge.net/">FormMail</a>
            &copy; 2001-2003 London Perl Mongers
          </p>
        </td>
      </tr>
    </table>
  </body>
</html>
END
}

=item mailer ()

Returns an object satisfying the definition in L<CGI::NMS::Mailer>,
to be used for sending outgoing email.

=cut

sub mailer {
  my ($self) = @_;

  return $self->{Mailer};
}

=back

=head1 SEE ALSO

L<CGI::NMS::Script>

=head1 MAINTAINERS

The NMS project, E<lt>http://nms-cgi.sourceforge.net/E<gt>

To request support or report bugs, please email
E<lt>nms-cgi-support@lists.sourceforge.netE<gt>

=head1 COPYRIGHT

Copyright 2003 London Perl Mongers, All rights reserved
```

```
=head1 LICENSE

This module is free software; you are free to redistribute it
and/or modify it under the same terms as Perl itself.

=cut

1;


END_INLINED_CGI_NMS_Script_FormMail
  $INC{'CGI/NMS/Script/FormMail.pm'} = 1;
}

}
#
# End of inlined modules
#
use CGI::NMS::Script::FormMail;
use base qw(CGI::NMS::Script::FormMail);

use vars qw($script);
BEGIN {
  $script = __PACKAGE__->new(
     DEBUGGING              => $DEBUGGING,
     name_and_version       => 'NMS FormMail 3.10c1',
     emulate_matts_code     => $emulate_matts_code,
     secure                 => $secure,
     allow_empty_ref        => $allow_empty_ref,
     max_recipients         => $max_recipients,
     mailprog               => $mailprog,
     postmaster             => $postmaster,
     referers               => [@referers],
     allow_mail_to          => [@allow_mail_to],
     recipients             => [@recipients],
     recipient_alias        => {%recipient_alias},
     valid_ENV              => [@valid_ENV],
     charset                => $charset,
     date_fmt               => $date_fmt,
     style                  => $style,
     no_content             => $no_content,
     double_spacing         => $double_spacing,
     wrap_text              => $wrap_text,
     wrap_style             => $wrap_style,
     send_confirmation_mail => $send_confirmation_mail,
     confirmation_text      => $confirmation_text,
     %more_config
  );
}

$script->request;
```