

Geodetic Data Modelling System: Honours Thesis

An online interface

Michael Still
964076

Geodetic Data Modelling System: Honours Thesis: An online interface
by Michael Still

Abstract

TODO

Table of Contents

1. Introduction	1
Motivation	1
Outcome.....	1
2. Implementation rationale.....	3
Selection of the implementation technique	3
PHP	3
ASP.....	3
C or C++ CGI.....	3
Perl CGI.....	4
Conclusion	4
Presentation of the user interface.....	4
Selection of an image format for graph display.....	4
TIFF.....	4
GIF.....	5
JPEG	5
PNG.....	7
Conclusion	7
3. Implementation	9
Welcome screen.....	9
4. Conclusion.....	11
5. References.....	13
References	13
6. Appendix One: GDMS Internet interface scripting elements	15
Introduction.....	15
commands.....	15
dataset.....	15
datasets	15
eastplot	16
lparams	16
motd	16
northplot.....	16
upplot	16
winparams	17
7. Appendix Two: Source code.....	19
8. Appendix Three: Installation.....	27
9. Appendix Four: A sample configuration file	29

List of Figures

2-1. A picture before JPEG compression.....	5
2-2. A picture after it has been JPEG compressed	5
2-3. The sample text before JPEG compression	6
2-4. The text after it has been JPEG compressed.....	6
3-1. GDMS Internet interface welcome screen.....	9

Chapter 1. Introduction

Motivation

At the time that the **Geodetic Data Modeling System** (GDMS) implementation project was initiated, several of the potential users expressed a desire for the application to be available over the Internet. There are a variety of reasons that this type of functionality is enticing -- the main ones for GDMS are that it allows casual users to ability to analyse data, whilst not having to maintain their own copies of the datasets, and it allows users who would normally use the X windows interface to the application to access data whilst "in the field", or otherwise physically separated from their normal research location.

Initially, it was thought that a Internet interface to GDMS was outside the achievable scope of the project for 2002. However, as the year progressed, it became clear to me that implementing a Internet interface was indeed achievable, and would add genuinely useful functionality to the application. Hence this honours extension was undertaken.

Outcome

This document describes the design process and implementation of the GDMS Internet interface, and perl application intended to operate under the Common Gateway Interface (CGI) offered by most modern web servers. This application allows the online use of almost all of the functionality available under the X windows GDMS system.

Chapter 2. Implementation rationale

Selection of the implementation technique

A variety of implementation techniques were considered for the GDMS Internet interface before a final selection was made. This section documents the various alternatives that were considered, and then justifies the decision that was made.

PHP

PHP stands for *PHP: Hypertext Preprocessor*, although there is some evidence that the name originally stood for *Personal Home Page tools* (Lerdorf 1995). PHP is an in page scripting language which was originally developed by Rasmus Lerdorf (Lerdorf 1995). It is now developed by a online team of developers (AIMS Group, 2002).

Because PHP is an in-page scripting language, the actual code to generate the HTML page viewed by the user of the web browser is actually stored within the HTML page on the servers secondary storage. A trivial example is:

```
<html>
<head>
<title>PHP Test</title>
</head>
<body>
<?php echo "Hello World<p>"; ?>
</body>
</html>
```

In this example, the tag starting with `?php` is the script element. This example was sourced from PHP Documentation Group, 2002. The output of this script will be a simple HTML page saying "Hello World".

The main advantage of in-page scripting is that the source code of the application is very tightly tied to the HTML presentation of the application's user interface. This means that when a user is editing a page, it is apparent what the given code does at the time that the page is created.

The disadvantage of such a system is that it then requires that the users would be editing the HTML appearance of an application must also understand the scripting language which the application is implemented in. This means that graphics designers and layout consultants require further training -- the application is no longer a "black box" which the HTML designer can simply be a user of. The other main disadvantage is that the inclusion of the scripting within the HTML pages is that it clutters the HTML representation of the user interface, making it much harder to read. This problem is compounded by poor in-page scripting support in many HTML editors, including Mozilla. These editors discard tags they don't comprehend at the time of editing the page, and this unfortunately includes tags such as PHP.

Finally, developing the GDMS Internet interface in PHP would necessitate the users of the application running web servers with PHP modules installed (PHP Documentation Group, 2002). This is an extra level of configuration for systems administrators for little additional benefit.

ASP

Active Server Pages (ASP) is Microsoft's equivalent of PHP. Discussion of this alternative is deliberately brief, because ASP effectively has all the advantages and disadvantages of PHP, with the additional constraint that it only operates on Microsoft web server products running on Microsoft Operating Systems. This is not an acceptable constraint for the main target user of the application -- the University of Canberra Survey Laboratory.

C or C++ CGI

Another implementation alternative considered was to implement the system in either C or C++ using the Common Gateway Interface (CGI) subsystem offered by most web servers. This implementation technique has the advantage of the code for the GDMS Internet interface being in the same language as the GDMS application itself. It however has the disadvantage that parsing the incoming request URLs is less trivial, as there is limited language support without the inclusion of non-standard application libraries (Kahan 2002).

C and C++ are also not idiomatic methods of implementing CGI applications, and this would therefore increase the difficulty of maintaining the system for future students and administrators.

Perl CGI

Perl was originally developed by Larry Wall in 1987 (Perl Mongers, 1999) and is now developed by a team of Open Source developers. Perl has well integrated CGI support built into the language itself, and is optimized for the parsing of complex strings (such as the requests which are returned by a web browser to the GDMS Internet interface via the CGI apparatus).

Perl is also extremely stable, well documented, and is already installed at most sites.

Conclusion

After consideration of all of the factors outlined above, Perl CGI was identified as the most suitable implementation language for the GDMS Internet interface.

Presentation of the user interface

In accordance with Internet user interface design best practise, the GDMS Internet interface presents a fully configurable user interface. This is implemented by providing extension tags the those which are available in the standard HTML specification (Raggett, Le Hors, & Jacobs 1999).

The GDMS Internet interface follows the recommendations of the most recent HTML specification at the time of development for the insertion of scripting tags into HTML source files (Raggett, Le Hors, & Jacobs 1999). Further documentation on the GDMS Internet interface tags available can be found in Appendix One of this document.

Selection of an image format for graph display

Much thought was expended on which image format to use for the plots displayed in the GDMS Internet interface. There are advantages and disadvantages to all the formats available at the time of writing this document. The following formats were considered for the GDMS Internet interface:

TIFF

TIFF (Tagged Image File Format) is a raster (bitmap) image format which was originally produced by Aldus and Microsoft. Aldus was later acquired by Adobe, who manage the TIFF specification to this day. At the time of writing, the current version of the TIFF specification is TIFF version 6.0 (Adobe 1992).

The main advantage of the TIFF image format is that it is extremely mature. It has a large selection of options to ease the development of imaging software, for example the developer can use which ever endian representation is convenient for them, as well as such fundamental parameters as photometric interpretation (whether a high value is lighter or darker than a low value for a given colour sample within a pixel) (Adobe 1992). There is also a stable, well supported Open Source application programmers interface available to manage creation and interpretation of images within the TIFF format (Warmerdam & Welles 2002). The interface to this library does not frequently change, which is also an advantage for future portability of the GDMS Internet interface.

However, some of these very advantages lead to the disadvantages that the TIFF format suffers from. Because the content of the file is so loosely specified, it is quite hard to write an application which can correctly decode all possible TIFF format images. This has resulted in there being only a few high quality TIFF viewing applications being available. The relative difficulty of implementing a TIFF decoder has also resulted in the web browser support for TIFF images being very poor (Bither 2002).

TIFF is therefore an inappropriate image format for use for the GDMS Internet interface, as good browser support in as many browsers as possible is vital.

GIF

The Graphics Interchange Format (GIF) was co-developed by CompuServe Inc. and Unisys Corporation in the late 1980s and early 1990s. GIF is dependant on LZW compression (CompuServe 1990), which is now having its patent enforced by Unisys (Anonymous 2002). The GIF format itself is quite limited, with a maximum palette size of 255 entries -- in other words 255 distinct colours are available in each image.

Vendors are also dropping support for the GIF format because of the expense of licensing the Unisys LZW patent, for example, the Open Source *libgif* is no longer supported for the creation of GIF images (Raymond, 1998).

Despite all of these largely political issues, the GIF format is still well supported in all web browser versions, which would make it an ideal candidate for the GDMS Internet interface if it wasn't for the unavailability of a stable and reliable compression library.

JPEG

JPEG is an interesting image format because whilst the JPEG compression codec has been standardized by ISO, the actual on disc format commonly called JPEG is not part of the ISO standard. The file format commonly called JPEG is actually JPEG JFIF -- created by the Independent JPEG Group (Lilly 1996).

The JPEG support in web browsers is now excellent, with support becoming much more common after 1996 (Lilly 1996). The biggest factor stopping the use of JPEG within the GDMS Internet interface is that the compression codec is lossy.

Loss

The JPEG compression algorithm is lossy -- in other words, the act of compressing the image results in image data being thrown away. This has the effect of noticeably reducing the clarity of the images -- a characteristic which is especially noticeable with images which contain text at smaller point sizes. Use of such text is a feature of the GDMS graphing subsystem.

For example, JPEG compressing the following image results in the following output image:



Figure 2-1. A picture before JPEG compression



Figure 2-2. A picture after it has been JPEG compressed

The effects are much worse with an image which contains text at small point sizes:

```
<sidebar>
<heading refname="" type="sidebar" toc="no">Accu
<p>Why does the loss in lossy compression algori
add say a barcode to the image, so you uncompre
et of loss is introduced. You can imagine that i
<p>Whether this is a problem depends on the type
repeatedly uncompresses and recompresses an imag
on.</p>

<figure>
<heading refname="picture-start" type="figure" to


<figure>
<heading refname="text-start" type="figure" toc=

```

Figure 2-3. The sample text before JPEG compression

```

<sidebar>
<heading refname="" type="sidebar" toc="no">Accu
<p>Why does the loss in lossy compression algori
add say a barcode to the image, so you uncompress
et of loss is introduced. You can imagine that i

<p>Whether this is a problem depends on the type
repeatedly uncompresses and recompresses an imag
on.</p>

<figure>
<heading refname="picture-start" type="figure" to


<figure>
<heading refname="text-start" type="figure" toc=


```

Figure 2-4. The text after it has been JPEG compressed

PNG

Portable Network Graphics (PNG) was initially developed because of the patent problems with the GIF format described earlier in this section. The PNG format has since grown well past the feature set provided by GIF, and now rivals TIFF for completeness (Randers-Pehrson 1999).

The PNG support in most browsers is not perfect. However, browsers above version three for both Netscape and Microsoft Internet Explorer support PNG. There is an actively maintained and well supported Open Source programmers interface to the PNG format, in fact it is the reference implementation of the specification (Roelofs 2002).

Conclusion

Based on the factors outlined above, the PNG image format was selected for the GDMS Internet interface. This is because of it's excellent browser support, as well as it's mature and stable Open Source application programmers interface.

Chapter 3. Implementation

This chapter discusses the actual application which was produced.

Welcome screen

The following page greets users when they start the GDMS Internet interface.

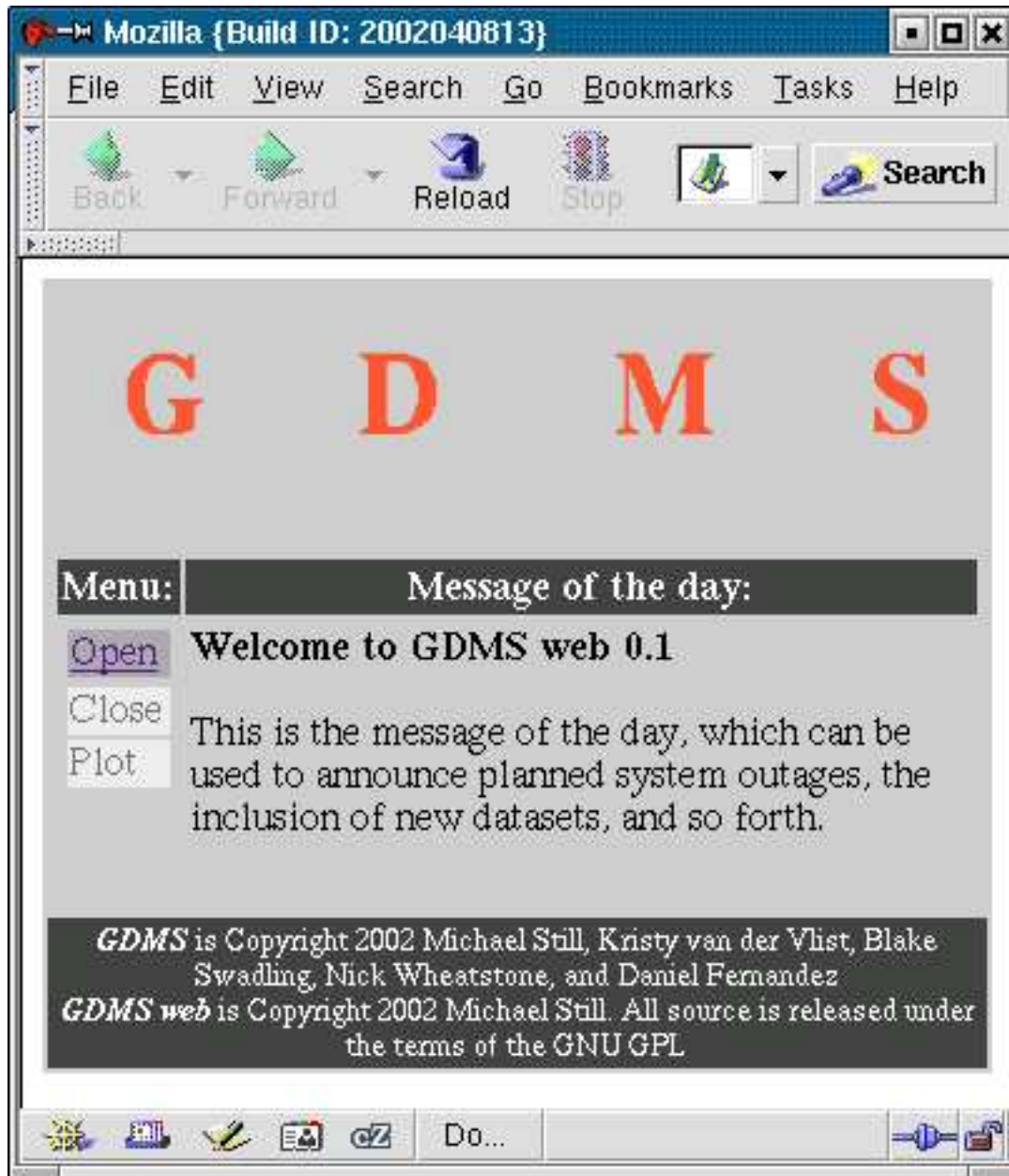


Figure 3-1. GDMS Internet interface welcome screen

Chapter 4. Conclusion

...

Chapter 5. References

References

- Adobe, 3 June 1992, *TIFF Revision 6.0*, [Online] <http://partners.adobe.com/asn/developer/pdfs/tn/TIFF6.pdf>
- AIMS Group, 2002, *PHP CVS mailing list archive*, [Online] <http://marc.theaimsgroup.com/?l=php-cvs>
- Anonymous, 2 February 2002, *The GIF Controversy: A Software Developer's Perspective*, [Online] <http://www.cloanto.com/users/mcb/19950127giflzw.html>
- Bither, B., [Last accessed] November 2002, *Benefits of the PNG Image Format*, [Online] Available: <http://www.atalasoft.com/png.asp>
- CompuServe, 1990, *Graphics Interchange Format: Version 89a* [Online] <http://www.w3.org/Graphics/GIF/spec-gif89a.txt>
- Kahan, J., 12 June 2002, *Libwww - the W3C Protocol Library*, [Online] <http://www.w3.org/Library/>
- Lerdorf, R., 8 June 1995, *Personal Home Page Tools (PHP Tools)*, [Online] [http://groups.google.com/groups?selm=3r7pgp\\$aa1@ionews.io.org](http://groups.google.com/groups?selm=3r7pgp$aa1@ionews.io.org)
- Lilly, C., 13 February 1996 [Last updated: 18 July 2001], *JPEG JFIF*, [Online] <http://www.w3.org/Graphics/JPEG/>
- Perl Mongers, 1999, *Perl History*, [Online] <http://www.perl.org/press/history.html>
- PHP Documentation Group, 2002, *PHP Manual*, [Online] <http://www.php.net/manual/en/print/>
- Raggett, D., Le Hors, A., Jacobs, I., 24 December 1999 [last update], *HTML 4.01 Specification*, [Online] Available: <http://www.w3.org/TR/html4/>
- Randers-Pehrson, G., July 1999, *PNG (Portable Network Graphics) Specification, Version 1.2*, [Online] <http://www.libpng.org/pub/png/spec/png-1.2.pdf>
- Raymond, E. S., 8 July 1998, *The GIFLIB home page*, [Online] <http://tuxedo.org/~esr/giflib/>
- Roelofs, G., 2002, *libpng*, [Online] <http://www.libpng.org/pub/png/libpng.html>
- Warmerdam, F. and Welles, M., 24 January 2002 [last update], *The libtiff home page*, [Online] <http://www.libtiff.org/>

Chapter 6. Appendix One: GDMS Internet interface scripting elements

Introduction

This appendix documents the various tags which are implemented by the GDMS Internet interface in addition to the standard HTML tags normally available. All of these tags are within the scripting namespace provided by HTML 4.01 (Raggett, Le Hors, & Jacobs 1999).

commands

This tag will display a menu of the commands available in the current context -- for example, when the Internet interface first starts, it will list the open command. Commands which are not available, but are normally available will appear in a disabled style. This results in a list of commands which is consistent, and therefore reduces the potential for user confusion.

An example use is as follows:

```
&{commands};
```

The display of the command menu is altered by the following configuration file entries:

- *commandentry*: this HTML snippet is used for enabled commands
- *discommandentry*: this HTML snippet is used for disabled commands

For example, these configuration entries ship by default with the GDMS Internet interface:

```
# This line is simple HTML used for format the command entries,
# %s is the name of the command (including link HTML)
$commandentry = "<tr><td bgcolor=\"AAAAAA\">%s</td><tr>";

# This line is used for commands which aren't available...
$discommandentry = "<tr><td bgcolor=\"EEEEEE\">
    <font color=\"777777\">%s</font></td></tr>";
```

dataset

This tag will display the name of the current dataset, if one is open. This might possibly be a descriptive string if processing has occurred on the dataset.

An example usage is:

```
&{dataset};
```

The dataset name is the portion of the filename before the extensions .dat1, .dat2, and .dat3 are applied. For example *mb_AUCK_GPS* is a dataset name.

datasets

This tag will display a list of the datasets available in the dataset directory.

An example usage is:

```
&{datasets};
```

The dataset directory is configured with the following configuration file entry:

```
# This line defines where the datasets are stored
$datasets = "/home/httpd/gdms-datasets/";
```

There are also several configuration options which alter the appearance of the list of datasets available. These are:

- *selectstart*: this is used for any output which should appear at the beginning of the list. This could include HTML tags for the creation of the required list markup (for example tables).
- *selectentry*: this configuration item is used for each entry in the list. The special text %s is replaced by the HTML for the entry itself.
- *selectend*: this is used for any HTML required to finalize the list.

```
# This is used for selection lists (for instance datasets)
$selectstart = "List start";
$selectentry = "<LI>%s";
$selectend = "List end";
```

eastplot

This tag will insert the HTML and required URL for the East direction plot from the current dataset. The output of the tag will be empty if there is no currently selected dataset. The plots are 24 bit colour PNG images, as discussed in the implementation methodology section of this document.

An example usage is:

```
&{eastplot};
```

lsparams

This tag will output the HTML form required for the entry of Least Squares processing parameters. The user will then select "Ok" on this web form, and the Least Squares regression will occur.

An example usage is:

```
&{lsparams};
```

motd

This tag will display a welcome message to the user -- the Message Of The Day (MOTD). This can be used to inform users of new datasets now being available, planned system outages, and other such informational messages.

An example usage is:

```
&{motd};
```

northplot

This tag will insert the HTML and required URL for the North direction plot from the current dataset. The output of the tag will be empty if there is no currently selected dataset. The plots are 24 bit colour PNG images, as discussed in the implementation methodology section of this document.

An example usage is:

```
&{northplot};
```

upplot

This tag will insert the HTML and required URL for the Up direction plot from the current dataset. The output of the tag will be empty if there is no currently selected dataset. The plots are 24 bit colour PNG images, as discussed in the implementation methodology section of this document.

An example usage is:

```
&{upplot};
```

winparams

This tag will output the HTML form required for the entry of windowing processing parameters. The user will then select "Ok" on this web form, windowing operation will occur.

An example usage is:

```
&{winparams};
```


Chapter 7. Appendix Two: Source code

```
#!/usr/bin/perl

# GDMS web broker
# Copyright (C) Michael Still          2002
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

# This is the broker between the GDMS code and the CGI server. It's role is to
# parse templates, execute requests, and cache results for speedy processing
# next time that request is seen...
use strict;
use CGI;

my($result, $command, $TEMP);

# Variables set by the config file
my($templates, $datasets, $commandentry, $discommandentry, $selectstart, $selectentry, $selectend, $plotcache, $tmpdir, $rooturl, $ploturl, $gdms, $temp);

# Setup the CGI module
$result = new CGI();

# Read in the config file
eval 'cat gdms.config' or die "GDMSweb $$: Could not read config file: $@";
$temp = $gdms;
$temp =~ s/\/.*$//;

# Determine what page we are accessing
$command = $result->param('command');
if($command eq ""){
    $command = "main";
}

# Some pages just do some processing and then immediately redirect to another page.
# I need to deal with these before I can display the page headers
print STDERR "GDMSweb $$: Processing $command\n";
print STDERR "GDMSweb $$: Dataset is permanent\n" if($result->param('intype') eq "");
print STDERR "GDMSweb $$: Dataset is temporary\n" if($result->param('intype') ne "");

if($command eq "lsprocess"){
}
elsif($command eq "windowprocess"){
    $temp = "$rooturl?dataset=".$result->param('dataset')."-win".$result->param('wtype').
    "&command=plot";
    if($result->param('desc') ne ""){
        $temp = $temp."&desc=".$result->param('desc');
    }
    $temp = $temp."&intype=temp";

    # Do the processing here
    performWindow();

    # And now bounce of to the real page
    print STDERR "GDMSweb $$: Redirecting after window to: $temp\n";
    print $result->redirect($temp);
    exit;
}
```

```

}
elseif($command eq "interpprocess"){
    $temp = "$rooturl?dataset=". $result->param('dataset')."-interp". $result->
    param('itype').
    "&command=plot";
    if($result->param('desc') ne ""){
        $temp = $temp."&desc=". $result->param('desc');
    }
    $temp = $temp."&intype=temp";

    # Do the processing here
    performInterpolation();

    # And now bounce of to the real page
    print STDERR "GDMSweb $$: Redirecting after interpolation to: $temp\n";
    print $result->redirect($temp);
    exit;
}
elseif($command eq "fftprocess"){
    $temp = "$rooturl?dataset=". $result->param('dataset')."-fft&command=plot";
    if($result->param('desc') ne ""){
        $temp = $temp."&desc=". $result->param('desc');
    }
    $temp = $temp."&intype=temp";

    # Do the processing here
    performFFT();

    # And now bounce of to the real page
    print STDERR "GDMSweb $$: Redirecting after FFT to: $temp\n";
    print $result->redirect($temp);
    exit;
}

# Determine if the GDMS script for this page exists already
print $result->header;
print processTemplate("$templates/$command.html");
exit;

#####
# Find the template file, and then parse it
sub processTemplate(@args){
    my($file) = @_;
    my($pre, $post, $cmd, $output, $len, $line);

    # This local usage is used to make the TEMPLATE filehandle local to this
    # subroutine...
    local *TEMPLATE;
    print STDERR "GDMSweb $$: Processing template file: $file\n";

    $output = "";
    if(!open TEMPLATE, "< $file"){
        print STDERR "GDMSweb $$: $file not found\n";
        open TEMPLATE, "< $templates/commandnotfound.html";
    }
    while(<TEMPLATE>){
        # Repeatedly process a line until there are not more template commands
        $line = $_;
        $len = -1;

        while($len != length($line)){
            $len = length($_);

            # todo: there must be a better way of doing this...
            $_ = $line;
            if(/(.*)&{([^\%]*)};(.*)/){
                $pre = $1;
                $cmd = $2;
                $post = $3;
            }
        }
    }
}

```

```

if($cmd eq "commands"){
    # List the available commands
    $line = $pre.getCommands().$post;
}
elseif($cmd eq "dataset"){
    # The name of the current dataset
    if($result->param('desc') eq ""){
        $line = $pre.$result->param('dataset').$post;
    }
    else{
        $line = $pre.$result->param('desc').$post;
    }
}
elseif($cmd eq "datasets"){
    # List the datasets in the dataset directory
    my($temptotal, $tempfile);
    $temptotal = "";

    open TEMP, "find $datasets -type f -name \"*.dat1\" |";
    while(<TEMP>){
        my($linecount);
        $tempfile = $_;
        $tempfile =~ s/$datasets\/.*//;
        $tempfile =~ s/.dat1\n$//;

        $linecount = `cat $datasets/$tempfile.dat1 | wc -l`;
        $temptotal = $temptotal.
            substHTML($selectentry,
                "<a href=\"\$rooturl?command=plot&dataset=$tempfile\">$tempfile</a> ".
                "($linecount lines)");
    }
    close TEMP;

    $line = $pre.$temptotal.$post;
}
elseif($cmd eq "motd"){
    # Output a message of the day
    $line = $pre.processTemplate("$templates/motd.html").$post;
}
elseif($cmd eq "northplot"){
    # A plot in the X;
    $line = $pre.generateAndLink("x").$post;
}
elseif($cmd eq "eastplot"){
    # A plot in the Y direction
    $line = $pre.generateAndLink("y").$post;
}
elseif($cmd eq "upplot"){
    # A plot in the Z direction
    $line = $pre.generateAndLink("z").$post;
}
elseif($cmd eq "lsparams"){
    # Output a HTML form for the parameters of a LS regression
    $line = $pre.generateForm("lsprocess", $result->param('dataset'),
        "$templates/lsparams.html").$post;
}
elseif($cmd eq "winparams"){
    # Output a HTML form for the parameters of a LS regression
    $line = $pre.generateForm("windowprocess", $result->param('dataset'),
        "$templates/winparams.html").$post;
}
elseif($cmd eq "interpparams"){
    # Output a HTML form for the parameters of a LS regression
    $line = $pre.generateForm("interpprocess", $result->param('dataset'),
        "$templates/interpparams.html").$post;
}
}
}

# And now we can print out the resultant line

```

```

$output = $output.$line;
}

return $output;
}

# Determine what commands should be available at this time
sub getCommands(@args){
    my($output, $temp);

    $output = "";
    # If there is no current dataset
    if($result->param('dataset') eq ""){
        # Dataset open
        $output = $output.substHTML($commandentry, "<a href=\"\$rooturl?command=open\">Open</a>");

        # Maths
        $output = $output.substHTML($discommandentry, "Least squares");
        $output = $output.substHTML($discommandentry, "Windowing");
        $output = $output.substHTML($discommandentry, "Interpolation");
        $output = $output.substHTML($discommandentry, "FFT");
    }
    else{
        # Dataset open
        $output = $output.substHTML($commandentry, "<a href=\"\$rooturl?command=open\">Open</a>");

        # Maths
        $temp = "<a href=\"\$rooturl?command=ls&dataset=\".$result->param('dataset');
        $temp = $temp."&intype=temp" if($result->param('intype') ne "");
        $temp = $temp."</a>Least squares</a>";
        $output = $output.substHTML($commandentry, $temp);

        $temp = "<a href=\"\$rooturl?command=window&dataset=\".$result->param('dataset');
        $temp = $temp."&intype=temp" if($result->param('intype') ne "");
        $temp = $temp."</a>Windowing</a>";
        $output = $output.substHTML($commandentry, $temp);

        $temp = "<a href=\"\$rooturl?command=interp&dataset=\".$result->param('dataset');
        $temp = $temp."&intype=temp" if($result->param('intype') ne "");
        $temp = $temp."</a>Interpolation</a>";
        $output = $output.substHTML($commandentry, $temp);

        $temp = "<a href=\"\$rooturl?command=fftprocess&dataset=\".$result->param('dataset');
        $temp = $temp."&intype=temp" if($result->param('intype') ne "");
        $temp = $temp."</a>FFT</a>";
        $output = $output.substHTML($commandentry, $temp);
    }

    return $output;
}

# Substitute into the HTML stub from the config file
sub substHTML(@args){
    my($html, $insert) = @_;
    my($temp);

    $temp = $html;
    $temp =~ s/%s/$insert/;
    return $temp;
}

# This subroutine deals with generating plots as required and then outputs
# the HTML needed to link to that image
sub generateAndLink(@args){
    my($dir) = @_;
    my($file, $unique);
    local *COMMANDS;

    # Generate the filename
    $unique = "$$-".time()."-".rand();
    $file = "$plotcache/".$result->param('dataset')."-$dir.png";

```

```

    if(! -f $file){
# We need to generate the image
print STDERR "GDMSweb $$: Plot cache miss for ".$result->param('dataset')." ($dir)\n";
open COMMANDS, "> $tmpdir/gdms-$unique.cmd" or
    die "Could not open temporary file $tmpdir/gdms-$unique.cmd\n";

# Open the dataset. It might be in a temporary location
if($result->param('intype') ne "temp"){
    print COMMANDS "open $datasets/".$result->param('dataset')."\n";
}
else{
    print COMMANDS "open $tmpdir/".$result->param('dataset')."\n";
}

print COMMANDS "plot $dir $file\n";
close COMMANDS;

# Execute the gdms main program with this command script
`$gdms -b $tmpdir/gdms-$unique.cmd`;
print STDERR "GDMSweb $$: GDMS execute return code is $?\n";
}

# Now link to that image
return "<img src=\"$ploturl/".$result->param('dataset').".$dir.png\">";
}

# Generate a HTML form for the required paramers
sub generateForm(@args){
    my($command, $dataset, $passedform) = @_;
    my($resultstring, $DUMPFIL);

    # Start the form
    $resultstring = "<form method=\"post\" action=\"$rooturl\" ";
    $resultstring = $resultstring."enctype=\"application/x-www-form-urlencoded\">";
    $resultstring = $resultstring."<input type=\"hidden\" name=\"command\" value=\"$command\" />";
    $resultstring = $resultstring."<input type=\"hidden\" name=\"dataset\" value=\"$dataset\" />";
    if($result->param('intype') ne ""){
        print STDERR "Continuing temporary tag\n";
        $resultstring = $resultstring."<input type=\"hidden\" name=\"intype\" value=\"temp\" />";
    }

    $resultstring = $resultstring.processTemplate("$passedform");

    # End the form
    $resultstring = $resultstring."<BR><BR><div align=\"center\">";
    $resultstring = $resultstring."<input type=\"submit\" name=\"commit\" value=\" OK \" /></div>";

    return $resultstring;
}

# Create a dataset in the temp location which is the output of a LS regression
sub performLs(){
    my($unique);
    local *COMMANDS;

    $unique = "$$-".time()."-".rand();
    open COMMANDS, "> $tmpdir/gdms-$unique.cmd" or
    die "Could not open temporary file $tmpdir/gdms-$unique.cmd\n";

    # Open the dataset. It might be in a temporary location
    if($result->param('intype') ne "temp"){
        print COMMANDS "open $datasets/".$result->param('dataset')."\n";
    }
    else{
        print COMMANDS "open $tmpdir/".$result->param('dataset')."\n";
    }

    print COMMANDS "ls \n";
    close COMMANDS;
}

```

```

    # Execute the gdms main program with this command script
    '$gdms -b $tmpdir/gdms-$unique.cmd';
}

# Create a dataset in the temp location which is the output of a windowing
sub performWindow(){
    my($unique);
    local *COMMANDS;

    $unique = "$$-.time()-.rand()";
    open COMMANDS, "> $tmpdir/gdms-$unique.cmd" or
    die "Could not open temporary file $tmpdir/gdms-$unique.cmd\n";

    # Open the dataset. It might be in a temporary location
    if($result->param('intype') ne "temp"){
        print COMMANDS "open $datasets/". $result->param('dataset')." \n";
    }
    else{
        print COMMANDS "open $tmpdir/". $result->param('dataset')." \n";
    }

    print COMMANDS "window ". $result->param('wtype')." ".
    $result->param('wsize')." ". $result->param('woverlap')." $tmpdir/".
    $result->param('dataset')." -win". $result->param('wtype')." \n";
    close COMMANDS;

    # Execute the gdms main program with this command script
    '$gdms -b $tmpdir/gdms-$unique.cmd';
    print STDERR "GDMSweb $$: GDMS return code is $? \n";
}

# Create a dataset in the temp location which is the output of interpolation
sub performInterpolation(){
    my($unique);
    local *COMMANDS;

    $unique = "$$-.time()-.rand()";
    open COMMANDS, "> $tmpdir/gdms-$unique.cmd" or
    die "Could not open temporary file $tmpdir/gdms-$unique.cmd\n";

    # Open the dataset. It might be in a temporary location
    if($result->param('intype') ne "temp"){
        print COMMANDS "open $datasets/". $result->param('dataset')." \n";
    }
    else{
        print COMMANDS "open $tmpdir/". $result->param('dataset')." \n";
    }

    print COMMANDS "interp ". $result->param('itype')." ".
    $result->param('isamplerate')." $tmpdir/".
    $result->param('dataset')." -interp". $result->param('itype')." \n";
    close COMMANDS;

    # Execute the gdms main program with this command script
    '$gdms -b $tmpdir/gdms-$unique.cmd';
    print STDERR "GDMSweb $$: GDMS return code is $? \n";
}

# Create a dataset in the temp location which is the output of the FFT
sub performFFT(){
    my($unique);
    local *COMMANDS;

    $unique = "$$-.time()-.rand()";
    open COMMANDS, "> $tmpdir/gdms-$unique.cmd" or
    die "Could not open temporary file $tmpdir/gdms-$unique.cmd\n";

    # Open the dataset. It might be in a temporary location
    if($result->param('intype') ne "temp"){
        print COMMANDS "open $datasets/". $result->param('dataset')." \n";
    }
}

```

```

    else{
print COMMANDS "open $tmpdir/" . $result->param('dataset') . "\n";
    }

    print COMMANDS "fft $tmpdir/" .
$result->param('dataset') . "-fft\n";
    close COMMANDS;

    # Execute the gdms main program with this command script
    '$gdms -b $tmpdir/gdms-$unique.cmd';
    print STDERR "GDMSweb $$: GDMS return code is $?\n";
}

```

Code: eps

Chapter 8. Appendix Three: Installation

...

Chapter 9. Appendix Four: A sample configuration file

This appendix details a sample configuration file for the GDMS Internet interface. This file should be stored in the same directory as the GDMS CGI script. It should also be noted that the ".cep" configuration file that would normally reside in a user's home directory should also be in the same directory as the GDMS CGI script.

The easiest way to create a suitable ".cep" configuration file is to run GDMS as any user, configure it so that you are happy with the graph output, and then copy that .cep file to the right location. Don't forget to update the permissions so that the user the web server is running as has read and write permissions on the file.

The same gdms.config file is as follows:

```
#####
# Directories
#####

# This line defines where the templates are stored
$templates = "/home/httpd/gdms-templates/";

# This line defines where the datasets are stored
$datasets = "/home/httpd/gdms-datasets/";

#####
# HTML configuration
#####

# This line is simple HTML used for format the command entries, %s is the name
# of the command (including link HTML)
$commandentry = "<tr><td bgcolor=\"AAAAAA\">%s</td><tr>";

# This line is used for commands which aren't available...
$discommandentry = "<tr><td bgcolor=\"EEEEEE\"><font color=\"777777\">%s</font></td></tr>";

# This is used for selection lists (for instance datasets)
$selectstart = "List start";
$selectentry = "<LI>%s";
$selectend = "List end";

#####
# Caching configuration
#####

# This is the location of the plot cache -- it can get quite big
$plotcache = "/home/httpd/html/gdms-plots/";
$ploturl = "/gdms-plots/";

# Location for temporary files
$tmpdir = "/home/httpd/gdms-temp/";

#####
# Other
#####

# This is the root URL
$rooturl = "/cgi-bin/gdms.pl";

# This is the full path to the GDMS application
$gdms = "/home/httpd/gdmsbatch";
```

Code: eps

