Website installation notes

From SangerWiki

(Redirected from PfamWeb installation notes)

This page documents the installation and configuration of the Pfam website.

Contents

Catalyst basics

This is a brief description of some of the terms bandied around here...

Catalyst is a web-development framework. It's essentially a scaffold on which to build websites, taking care of the nitty-gritty details of determining which code serves which URL, sending HTTP headers, etc. Catalyst tries to enforces a clean separation of server logic, data and presentation, using the so-called **M**odel-**V**iew-**C**ontroller pattern:

Model

The model provides a store for the data that the site will serve. We use MySQL (http://www.mysql.com/) databases to store data, but the model could retrieve data from any source, from flat files to a remote server. Since we're using databases, we access the data through an object relational mapping (http://en.wikipedia.org/wiki/Object-relational_mapping) layer, DBIx::Class (http://search.cpan.org/dist/DBIx-Class/) , which allows us to treat database tables, columns and rows as objects and manipulate the date through those objects.

View

The view is responsible for generating the web pages that a user sees. We use Template Toolkit (http://www.template-toolkit.org/) (TT) to generate pages. TT is a templating system (http://en.wikipedia.org/wiki/Web_template_system) that allows dynamic web pages to be described using a mixture of HTML and code.

Controller

Controllers are the glue layer between data and presentation, responsible for implementing the logic of a website. In catalyst, controllers are perl modules with methods that map to URLs in the website. A controller method, or *action* is called in response to a hit on a particular URL. The controller is responsible for retrieving whatever data it requires from the model, manipulating it as necessary, and then handing the data to the view for presentation.

Pre-installation

CPAN configuration

It's pretty much essential to configure the perl tool `cpan` before starting to install any perl modules. There are two installation systems in use for modules in CPAN (http://www.cpan.org/) and both need to be correctly configured:

MakeMaker
: the older installation system, used by the majority of modules

Module::Build
: a supposedly improved build system, used by some newer modules

`cpan` will use both systems when installing catalyst prerequisites and the main headache in setting up `cpan` is getting them to install into the same, non-system directory. If you're installing modules into the system perl installation, things should be easier.

To get everything installed into a non-system directory, make sure that the `PERL5LIB` environment variable is set to point to that directory, so that dependencies are found as we go along:

```
csh % setenv PERL5LIB /perl/install/dir
or
bash% export PERL5LIB=/perl/install/dir
```

You'll then need to set the installation prefix and the `installdirs` parameter. To do this in an already configured `cpan` installation you need some variation on:

```
cpan> o conf makepl_arg 'PREFIX=/perl/install/dir INSTALLDIRS=site'
cpan> o conf mbuild_pl_arg '--prefix /perl/install/dir --installdirs site'
```

To make life a little easier, you can tell `cpan` to install prerequisites by default by setting

```
cpan> o conf prerequisites_policy follow
```

After changing any `cpan` configuration, you'll need to commit changes:

```
cpan> o conf commit
```

or you can set a parameter to make that happen automatically:

```
cpan> o conf auto_commit 1
cpan> o conf commit
```

Catalyst installation

The various methods of installing catalyst are listed here (http://search.cpan.org/~mramberg/Catalyst-Runtime/lib/Catalyst/Manual/Installation.pod) . The first one, the shadowcat installer (http://www.shadowcatsystems.co.uk/static/cat-install) appears to work well. It's a script that is downloaded and run using the same perl binary that will be used to run the site. It should install the prerequisites for catalyst itself and then the catalyst runtime module.

If you prefer, you can install catalyst using the `Task::Catalyst` module instead, but we haven't tested this method:

```
shell% cpan Task::Catalyst
...
```

PfamWeb prerequisites

Next, before installing the PfamWeb code itself, you'll need to install its prerequisites. The following is a list of "recommended" perl modules (values in brackets are the known minimum required version):

Bundle::BioPerl
: you'll need BioPerl (http://www.bioperl.org/) , but if it's already installed on your system somewhere, just make sure it's found `@INC`, probably by setting the path in `PERL5LIB`

Cache::FastMmap or Cache::Memcached (1.23)
: you will need one of these two caching mechanisms installed

Catalyst::Devel (1.0)
: required if you want to do catalyst development but not for just deploying catalyst

This is a list of **required** modules:

- Bio::Das::Lite (1.048)
- Catalyst::Action::RenderView
- Catalyst::Engine::Apache2
- Catalyst::Model::DBIC::Schema

- Catalyst::Plugin::Cache (0.03)
- Catalyst::Plugin::ConfigLoader
- Catalyst::Plugin::Email
- Catalyst::Plugin::HTML::Widget
- Catalyst::Plugin::PageCache (0.15)
- Catalyst::Plugin::Session::State::Cookie
- Catalyst::Plugin::Session::Store::FastMmap
- Catalyst::Runtime (5.7)
- Catalyst::View::TT
- Class::Data::Accessor
- Config::General (0.28)
- Data::Pageset
- Data::Validate::URI
- Data::Visitor::Callback
- Data::UUID
- DateTime
- DateTime::Format::MySQL
- DBD::mysql
- DBIx::Class
- DBIx::Class::HTMLWidget
- GD
- GD::Graph
- HTML::Tree
- HTML::Widget
- IO::All
- JSON
- LWP::Parallel::UserAgent
- SOAP::Data::Builder
- XML::Feed

You'll need to install prerequisites for these prerequisites, but if you've set `prerequisites_policy follow` in your `cpan` configuration, that should be automatic.

One problem module is `DBD::mysql`, which fails most tests unless specifically configured to be able to access a test database. It's often easier to install this module manually and add flags to the `perl Makefile.PL` line to specify a database for testing, as per these instructions (http://search.cpan.org/dist/DBD-mysql/lib/DBD/mysql/INSTALL.pod#Configuration) :

```
shell% perl Makefile.PL --testdb=test \
                        --testuser=username \
                        --testpassword=user_password \
                        --testhost=localhost \
                        --testport=3306
shell% make
shell% make test
shell% make install
```

PfamWeb installation

Installing the Pfam website itself involves three steps:

1. unpack the code
2. configure the application itself
3. configure the apache that will run the application

Installing the code

All of our code will be retrieved directly from the public CVS server at Sanger. There are some general instructions on checking out projects from Sanger CVS here (http://cvs.sanger.ac.uk/cvs.users.shtml) . You'll need four CVS modules in all:

PfamWeb
: the website code

PfamLib
: perl modules that are used by the website

PfamSchemata
: `DBIx::Class` schema definitions for the databases

PfamConfig
: sample configuration files

The modules are checked out directly into the directory where you want them to reside. For example, if you want the website code to be in the apache directory, parallel with the apache `htdocs` directory:

```
shell% mkdir /path/to/httpd/pfam-site
shell% cd /path/to/httpd/pfam-site
```

To checkout modules, first set the `CVSROOT` environment variable to point to the Sanger repository:

```
bash% export CVSROOT=:pserver:cvsuser@cvs.sanger.ac.uk:/cvsroot/pfamweb-public
or
csh%  setenv CVSROOT :pserver:cvsuser@cvs.sanger.ac.uk:/cvsroot/pfamweb-public
```

Login to the CVS server using the guest account. The password is "CVSUSER":

```
shell% cvs login
Logging in to :pserver:cvsuser@cvs.sanger.ac.uk:2401/cvsroot/pfamweb-public
CVS password: CVSUSER
shell%
```

Check out the modules:

```
shell% cvs co PfamWeb PfamLib PfamSchemata PfamConfig
cvs checkout: Updating PfamWeb
...
```

You should end up with something like the following directory structure:

- httpd
  - conf
  - htdocs
  - logs
  - pfam-site
    - PfamWeb
    - PfamLib
    - PfamSchemata
    - PfamConfig

Configuration

The website is configured through three configuration files:

the main apache configuration
: sets up the name of the server, the port on which it listens, etc.

website configuration
: an apache-style file that sets up the application itself, including, for example, the contents and arrangement of tabs within the tab-layout pages, the database connection parameters, etc

website-specific apache configuration
: handles the application-specific settings such as the `mod_perl` configuration, locations of the components of the website in the file system, etc. Importantly, this "local" apache configuration points to the location of the previous file, which configures the website itself

The main apache configuration is very site dependent and outside of the scope of this document. We'll describe the configuration of the website code and the apache configuration that applies to the Pfam site.

Configuring the site

There is a single, apache-style configuration file for the whole of the site, `PfamConfig/pfamweb.conf`. It's divided into four main sections:

General
: top-level configuration for the whole site

Model
: mainly the database connections that are used by the perl code

View
: the Template Toolkit configuration

Controller
: parameters that affect various aspects of the site operation, such as polling intervals for the sequence search, etc.

General configuration

The most important general setting is related to caching. We make heavy use of caching and you'll need to configure one of two mechanisms: FastMmap (http://search.cpan.org/dist/Cache-FastMmap/) or memcached (http://search.cpan.org/dist/Cache-Memcached/) .

FastMmap will only operate correctly when all of the apache servers are running on a single physical machine, because it uses shared memory. Memcached must be distributed across multiple servers, which spreads the memory load but requires more administration.

At Sanger we use memcached because our site is served by a farm of front-end web-servers, but if you will run the site on a single machine (as in Stockholm), you'll need to configure the site to use FastMmap.

Cache::FastMmap

Make sure that you've installed the `Cache::FastMmap` perl module.

Find the `<cache>` section of `pfamweb.conf`. Make sure that the `<backend>` section for Cache::FastMmap is **uncommented** and the `<backends>` section for Cache::Memcached is **commented**, as below:

```
<cache>
  # we can use two different caching mechanisms:
  #  FastMmap:         uses a memory mapped file and can only be used sensibly
  #                    on a single web server machine
  #  Cache::Memcached: uses a farm of cache machines and can be used from
  #                    multiple web server machines

  # use Cache::FastMmap as the backend
  <backend>
    class Cache::FastMmap
  </backend>

  # use Cache::Memcached as the backend
  #<backends>
  #  <default>
  #    class Cache::Memcached
  #    servers "1.2.3.1:11211"
  #    servers "1.2.3.2:11211"
  #    servers "1.2.3.3:11211"
  #    servers "1.2.3.4:11211"
  #    servers "1.2.3.5:11211"
  #    namespace production
  #  </default>
  #</backends>
</cache>
```

Cache::Memcached

Make sure that you've installed the `Cache::Memcached` perl module. You'll probably need to install memcached itself before trying to install the module.

Make sure that the `<backend>` section for Cache::FastMmap is **commented** and the `<backends>` section for Cache::Memcached is **uncommented**, as below:

```
<cache>
  # we can use two different caching mechanisms:
  #  FastMmap:         uses a memory mapped file and can only be used sensibly
  #                    on a single web server machine
  #  Cache::Memcached: uses a farm of cache machines and can be used from
  #                    multiple web server machines

  # use Cache::FastMmap as the backend
  #<backend>
  #  class Cache::FastMmap
  #</backend>

  # use Cache::Memcached as the backend
  <backends>
    <default>
      class Cache::Memcached
      servers "1.2.3.1:11211"
      servers "1.2.3.2:11211"
      servers "1.2.3.3:11211"
      servers "1.2.3.4:11211"
      servers "1.2.3.5:11211"
      namespace production
    </default>
  </backends>
</cache>
```

There should be one `servers` line for each of the machines that is running the memcached daemon. The default port is 11211 but this can be changed when starting the daemon, as can the amount of memory that each daemon is allowed to use. The `namespace` line sets up a cache namespace and can be used to distinguish between, for example, a development cache and a production cache.

Check the documentation for memcached (http://www.danga.com/memcached/) for details on running the daemons and the perldoc for Cache::Memcached (http://search.cpan.org/dist/Cache-Memcached/) and Catalyst::Plugin::Cache (http://search.cpan.org/dist/Catalyst-Plugin-Cache/) for details of namespaces.

Model configuration

This section sets up the database connections that the application needs. We use two databases:

`pfam_X_Y`
: is the latest version of the database that holds the Pfam data and should be accessed through a **read-only** user account

`web_user`
: holds various data for running the site and must be accessed through an account with **read/write** privileges.

Additionally, for the *PfamDB* section, you'll configure the DAS (http://www.biodas.org/) client that we use in the site.

PfamDB

To configure the database access you need to modify the three `connect_info` lines. The first contains the DBI-style connection string, e.g.

```
dbi:mysql:database=pfam_22_0;host=pfamdb;port=3306
```

`database` specifies the name of the Pfam database; `host` gives the name of the database server machine; `port` specifies the port for connecting to the database server.

Although we don't use perl DBI (http://search.cpan.org/dist/DBI/) directly to access the database, the connection string is formatted exactly as for DBI.

The second and third parameters are the database user account and password. This account should explicitly **not** have permissions to alter the database in any way.

The DAS configuration has three parameters:

dasDsn
: the URL for the DAS server. This should be left unchanged

dasTo
: the time (in seconds) that the DAS client should wait for a response from the DAS server. This can be increased if there are problems with the DAS-dependent features not returning results

dasProxy
: if you need to connect to external addresses through a proxy (as we do at Sanger), this is the URL of that proxy. Comment out this line if your server can connect to the web directly

Here's an example `Model` section for *PfamDB*:

```
<Model PfamDB>
  # PfamDB schema settings
  schema_class "PfamDB"
  connect_info "dbi:mysql:database=pfam_22_0;host=pfamdb;port=3306"
  connect_info pfamwebro
  connect_info password

  # DasLite connection parameters
  dasDsn    "http://das.sanger.ac.uk/das/pfam"
  dasTo     4
  dasProxy "http://wwwcache.sanger.ac.uk:3128"
</Model>
```

WebUser

As for *PfamDB*, the first `connect_info` line should contain the DBI connection string for the `web_user` database. The website must be able to write to this database, as this is the mechanism for queuing jobs for the sequence and other searches. The second and third `connect_info` lines contain the username and password for an account which should have write access to the web_user database. The last `connect_info` section sets the raw DBI parameters for the connection, such as `RaiseError` or `PrintError`, but this should be left untouched.

Here's an example configuration for *WebUser*:

```
<Model WebUser>
  # WebUser schema settings
  schema_class "WebUser"
  connect_info "dbi:mysql:database=web_user;host=pfamdb;port=3306"
  connect_info webuser
  connect_info password
  <connect_info>
    AutoCommit 1
  </connect_info>
</Model>
```

View configuration

Most of the *View* configuration section is devoted to setting up the contents of the various tab-layout pages in the site. That can and should be left untouched.

The following are the important settings, and the ones that will need to configured.

Set `INCLUDE_PATH` to the file system path for the `root` directory. This is used by Template Toolkit for finding template files. It should be an absolute path:

```
INCLUDE_PATH "/path/to/httpd/pfam-site/PfamWeb/root"
```

`COMPILE_DIR` points to a temporary directory that's used for caching compiled templates. When the server starts, each new template that it encounters is compiled into a snippet of perl, which is then stored and reused as required, rather than rebuilding the perl from the template at each use.

```
COMPILE_DIR /path/to/httpd/htdocs/tmp/pfam
```

Although it should make the server quicker, this parameter can cause problems for some configurations of apache. If in doubt, comment out this line.

The following group of parameters are set as Template Toolkit constants:

```
  <CONSTANTS>
    tmp         "/tmp"             # relative URL for temporary directory with Pfam graphics
    root        "/catalyst/pfam"   # root for site URLs
    tableLength 20                 # length of tables in the clan section (leave at default)
  </CONSTANTS>
```

The website generates lots of Pfam-style domain graphics. The path to the directory where the graphics should be written is specified by an environment variable in the apache configuration and used by the (separate) graphics generation code. The `CONSTANTS/tmp` setting tells the website where to find the domain graphics.

```
tmp "/tmp"
```

The website can be configured to run at various URLs. These are set in the apache configuration. For example, during development we run the site at a URL like:

```
http://testmachine.sanger.ac.uk:8000/catalyst/pfam
```

The last part of that url (`/catalyst/pfam`) is set in the apache configuration, but must be given in the website configuration as `CONSTANTS/root`.

If you'll run your site in a virtual host (http://httpd.apache.org/docs/2.2/vhosts/) , such as

```
http://livemachine.sanger.ac.uk/
```

you should set `CONSTANTS/root` to "/":

```
root "/"
```

Example configuration

```
<View TT>

  # set the location of the directory that will contain the template files
  INCLUDE_PATH "/path/to/httpd/pfam-site/root"

  # enable caching of the compiled templates
  #COMPILE_DIR /path/to/httpd/htdocs/tmp/pfam

  # use a single template to wrap around the content. This contains
  # all of the header and footer mark-up which was previously in
  # separate header.tt and footer.tt files
  WRAPPER "components/wrapper.tt"

  # allow perl expressions in the templates
  EVAL_PERL 1

  # allow TT to look for files along absolute paths
  ABSOLUTE 1

  # fixed values...
  <CONSTANTS>

    # the URL for the top-level tmp directory. This is where, for example,
    # we can find the auto-generated domain graphics
    tmp  "/tmp"

    # the root directory for the server, from the client's perspective. This is used
    # in the cookie handling routines for building the correct path
    root "/catalyst/pfam"

    # this controls the appearance of the table of families in the alignment tab of
    # the clans section. If there are more than this number of families, the table
    # will be split into two columns
    tableLength 20
  </CONSTANTS>

  # set up the contents of tab-layout pages. Leave untouched
  <VARIABLES>
    <blocks>
      ...
    </blocks>
    <layouts>
      ...
    </layouts>
  </VARIABLES>
</View>
```

Controller configuration

Some controllers can be configured in the website configuration file. The following sections list those controllers whose parameters you might want to change.

Search::Sequence

This controller handles the queuing of sequence searches. There are two parameters that affect the behaviour of the website code for searches:

pendingLimit
: the maximum number of jobs that are allowed in the queue. If the number of pending jobs is greater than this limit, the server will not submit the job and will show an error message to the user. A default of 100 seems reasonable

pollingInterval
: the user's web browser will poll the server at this interval (in seconds) to check on the progress of a search job. This shouldn't be set too low, otherwise the web-server will spend all its time checking on jobs, rather than serving web pages. Three seconds seems to be a reasonable default but it could be lowered to two if your backend compute is processing jobs very quickly

An example configuration:

```
<Controller Search::Sequence>
  # the maximum number of jobs that are allowed in the sequence search queue
  pendingLimit    100

  # the interval at which the user's browser should poll the search server for
  # the result of their search
  pollingInterval 3

</Controller>
```

Search::Sequence

The `Annotate` controller accepts user annotations for Pfam families. It presents a web form, accepts the submission of that form and validates the contents. If the contents are valid, it sends an email to Pfam to notify us of the annotation and returns a message to the user to tell them that their annotation has been accepted.

The most important parameter is the email address for annotation comments. Since annotations are currently processed only at Sanger, it's probably best to leave the email address as the default, `pfam-help@sanger.ac.uk`. This means that *all* annotations from *all* sites will come directly to Sanger. If you want to receive the annotations at your site, change the email address.

In order to avoid spam being submitted through the form, we perform various security checks and these can be configured to a certain extent. The annotation form is generated each time a user clicks the "add annotation" button in the site. If they don't submit the form within the time specified by `submissionTimeOut`, the server will reject the submission and return an error. This is to avoid having the form harvested and submitted repeatedly at a later date by robots. We check the timeout period by returning a token to the user when the form is generated and this token is encoded by checksumming with a secret word. This can be changed in the configuration if necessary but the default should be fine.

An example configuration:

```
<Controller Annotate>
  # mail address that should receive user-submitted annotations
  annotationEmail pfam-help@sanger.ac.uk

  # form submission times out after 1 hour
  submissionTimeOut 3600

  # the salt for the MD5 of the timestamp. This doesn't have to be anything
  # imaginative, just difficult to guess
  salt aSecretWord
</Controller>
```

PfamGraphicsTools

There are tools in the website that allow a user to generate a Pfam domain graphic by uploading an XML description file. The XML file is validated against a schema file and this controller needs to know the location of that file. `schemaFile` is the absolute path for the XSD file, which is found in the `root` directory. A second, related parameter, `schemaURI` is relative to the server root and shouldn't need to change.

The tool requires the user to upload an XML file and the location of the temporary directory that holds the uploaded files is given by `uploadDir`.

```
<Controller PfamGraphicsTools>
  # the path to the Pfam graphics XML schema
  schemaFile /path/to/httpd/pfam-site/PfamWeb/root/static/documents/pfamDomainGraphics.xsd

  # the URL for the schema document. This will be appended to $base by tools.tt,
  # and $base always ends in a slash, so this should NOT start with a slash
  schemaURI  static/documents/pfamDomainGraphics.xsd

  # the path to the destination directory for uploaded XML files
  uploadDir  /path/to/httpd/htdocs/tmp
</Controller>
```

Structure::GetPdbFile

We serve PDB files for the structure viewers in the site, which we currently download from an external FTP site. The site can be specified as `pdbFileUrl`. This can be changed to point to a more local mirror of the PDB files, but it must point to a directory that contains all of the uncompressed PDB files. Note that the RCSB PDB does not provide such a directory. It's probably best to leave this set to the default site (EBI) for the present. We intend to look again at this mechanism.

```
<Controller Structure::GetPdbFile>
  # the URL for retrieving uncompressed PDB files
  pdbFileUrl ftp://ftp.ebi.ac.uk/pub/databases/msd/pdb_uncompressed/
</Controller>
```

Configuring apache

Prerequisites

As a minimum we require the `mod_perl` apache module to be installed in your apache server. Depending on the exact configuration, you may also need `mod_rewrite`.

Refer to the `mod_perl` documentation (http://perl.apache.org/docs/2.0/index.html) for installation instructions, or the general compilation and installation (http://httpd.apache.org/docs/2.2/install.html) documentation for apache, for information on installing modules.

Environment variables

There are several environment variables which need to be set using the `PerlSetEnv` directive:

The code that draws Pfam graphics needs to know where to put the image files, which it picks up from `PFAM_DOMAIN_IMAGES`. This should match up with the website configuration parameter `CONSTANTS/tmp`, so that the website can find the domain graphics that are placed in this location:

```
PerlSetEnv PFAM_DOMAIN_IMAGES /path/to/httpd/htdocs/tmp
```

The domain graphics code uses `PFAMWEB_ROOT` when building the URL for the links in the domain graphics. This should match up with the website parameter `CONSTANTS/root`:

```
PerlSetEnv PFAMWEB_ROOT       /catalyst/pfam
```

The location of the website configuration file is given by `PFAMWEB_CONFIG`. This will be in the `pfam-site/conf` directory by default:

```
PerlSetEnv PFAMWEB_CONFIG     /path/to/httpd/pfam-site/PfamConfig/conf/pfamweb.conf
```

There are two environment variables that affect the level of debug information that will be written to the apache server log:

DBIC_TRACE
: if true, DBIx::Class will show the SQL queries that it's building

PFAMWEB_DEBUG
: if true, catalyst prints copious debug information to the log.

You will initially want catalyst debug but not DBIx::Class logging:

```
PerlSetEnv DBIC_TRACE 0
PerlSetEnv PFAMWEB_DEBUG 1
```

If you want to see the raw SQL queries that the server is generating, enable `DBIC_TRACE` to get them printed to the apache log.

Perl configuration

The perl code from the various CVS modules needs to be added to the perl `@INC` path, which is done through the `PerlSwitches` directives. This can also be used to enable perl warnings and, if possible, taint mode. Not all cpan modules are taint-safe, so you may see errors about taint mode. If you have problems, turn off taint mode by removing the "T" from the first `PerlSwitches` line:

```
PerlSwitches -wT
PerlSwitches -I/path/to/httpd/pfam-site/PfamLib
PerlSwitches -I/path/to/httpd/pfam-site/PfamSchemata
PerlSwitches -I/path/to/httpd/pfam-site/PfamWeb/lib
```

Note the `/lib` on the end of the final line. This is required.

If you need to set `@INC` to make sure that prerequisites are found in a non-system directory, you can either set `PERL5LIB` in the environment in which apache will run, or you can add the directory using another `PerlSwitches` directive:

```
PerlSwitches -I/path/to/perl/prerequisites
```

Server location

The basic premise of running the website under apache is that we want to serve static content, such as images, stylesheets, etc, using apache, but dynamic content using our perl code, running under `mod_perl`. Setting this up can be tricky, depending on the complexity of the server arrangement at your site.

At Sanger we run a complex system of load-balancing front-end web-servers, multiple backend web-servers for static content and a farm of backend web-servers for dynamic content. We need to use `mod_rewrite` to change URLs depending on whether they involve static or dynamic content.

If you're running on a single web-server machine, which will serve both static and dynamic content, things are much simpler. The following example apache configuration sets this up.

Set the root directory to be the top-level directory for static content. This allows apache to see and serve the static files for the site:

```
DocumentRoot /path/to/httpd/pfam-site/PfamWeb/root
```

Set our perl code to be the `mod_perl` handler for all URLs in this domain:

```
<Location />
  SetHandler          perl-script
  PerlResponseHandler PfamWeb
  Order      allow,deny
  Allow from all
</Location>
```

The second `Location` directive overrides this for URLs beginning `/static` and sets up the default apache handler for these locations:

```
<Location /static>
  SetHandler default-handler
  Order      allow,deny
  Allow from all
</Location>
```

Finally, we need to make sure that our temporary directory (which will contain the Pfam domain graphics images) is also served by apache. We want to keep this tmp directory separate from the codebase, so we place it in the regular apache htdocs directory. Since we've added a `DocumentRoot` directive to change this, we need to add an `Alias` to point to the right place and then add another `Location` directive to make sure that temporary files are served by apache:

```
Alias /tmp /path/to/httpd/htdocs/tmp

<Location /tmp>
  SetHandler default-handler
  Order      allow,deny
  Allow from all
</Location>
```

Example configuration

Below is a snippet of apache configuration which sets up the PfamWeb web application. This can be added directly to your `httpd.conf`, or read in from a separate file using an `include` directive. Finally, it could also be added to a virtual host (http://httpd.apache.org/docs/2.2/vhosts/) directive, to make the server run as one of many virtual hosts under a single apache.

```
# first, load mod_perl into apache
LoadModule perl_module modules/mod_perl.so

#----------------------------------------
# environment variables

# the location of temporary files for the drawing code
PerlSetEnv PFAM_DOMAIN_IMAGES /path/to/httpd/htdocs/tmp

# the base location for the application - used only by the domain graphics
# configuration files. This specifies the root that will be added to the
# beginning of the relative URLs for families that are drawn on a domain
# graphic
#PerlSetEnv PFAMWEB_ROOT       /catalyst/pfam

# the location of the main catalyst configuration file
PerlSetEnv PFAMWEB_CONFIG     /path/to/httpd/pfam-site/PfamConfig/pfamweb.conf

# set to 1 to enable DBIx::Class debug messages; adds SQL statements to the error log
PerlSetEnv DBIC_TRACE 0

# turn on Catalyst debugging, so that we get the traceback when there's an exception
PerlSetEnv PFAMWEB_DEBUG 1

#----------------------------------------
# set up the libraries that we'll need

PerlSwitches -w
PerlSwitches -I/path/to/httpd/pfam-site/PfamLib
PerlSwitches -I/path/to/httpd/pfam-site/PfamSchemata
PerlSwitches -I/path/to/httpd/pfam-site/PfamWeb/lib

# load the application module
PerlModule PfamWeb

#----------------------------------------
# server locations
DocumentRoot /path/to/httpd/pfam-site/PfamWeb/root

# set the PfamWeb code as the handler for all URLs
<Location />
  SetHandler          perl-script
  PerlResponseHandler PfamWeb
  Order      allow,deny
  Allow from all
</Location>

# but serve static content with the default apache handler
<Location /static>
  SetHandler default-handler
  Order      allow,deny
  Allow from all
</Location>

# override the DocumentRoot setting to point at the right location for the temporary directory
Alias /tmp /path/to/httpd/htdocs/tmp

# serve the contents of /tmp with the default apache handler too
<Location /tmp>
  SetHandler default-handler
  Order      allow,deny
  Allow from all
</Location>
```

Starting the server

After editing the configuration files, restart the apache server. If your configuration works, you should see something like this:

```
[debug] Debug messages enabled
[debug] Loaded plugins:
.----------------------------------------------------------------------------------------------------
| Catalyst::Plugin::Cache  0.03
| Catalyst::Plugin::ConfigLoader  0.14
| Catalyst::Plugin::Email  0.08
| Catalyst::Plugin::HTML::Widget  1.1
| Catalyst::Plugin::PageCache  0.15
| Catalyst::Plugin::Session  0.14
| Catalyst::Plugin::Session::State::Cookie  0.07
| Catalyst::Plugin::Session::Store::FastMmap  0.02
'----------------------------------------------------------------------------------------------------

[debug] Loaded dispatcher "Catalyst::Dispatcher"
[debug] Loaded engine "Catalyst::Engine::Apache2::MP20"
[debug] Found home "/path/to/httpd/pfam-site/PfamWeb"
[debug] Loaded Config "/path/to/httpd/pfam-site/PfamConfig/pfamweb.conf"
[debug] Loaded components:
.-----------------------------------------------------------------+----------.
| Class                                                           | Type     |
+-----------------------------------------------------------------+----------+
| PfamWeb::Controller::Annotate                                   | instance |
| PfamWeb::Controller::Browse                                     | instance |
| PfamWeb::Controller::Clan                                       | instance |
| PfamWeb::Controller::Clan::Relationship                         | instance |
| PfamWeb::Controller::Dead                                       | instance |
| PfamWeb::Controller::DomainGraphics                             | instance |

...

| /speciestree/text                                    | /speciestree/text
| /structure/getimage                                  | /structure/getimage/getImage
| /structure/getpdbfile                                | /structure/getpdbfile/getPdbFile
| /structure/graphics                                  | /structure/graphics/generateGraphics
| /structure/structuraldomains                         | /structure/structuraldomains/getStructuralDom
| /structure/viewer                                    | /structure/viewer/viewer
'------------------------------------------------------+---------------------------------------------

[info] PfamWeb powered by Catalyst 5.7007
[Fri Sep 07 14:58:40 2007] [notice] Apache/2.2.3 (Unix) mod_perl/2.0.3 Perl/v5.8.7 configured -- res
```

If you're using `Cache::FastMmap`, you may see the following perl warning:

```
Reference found where even-sized list expected at /path/to/perl/lib/perl5/site_perl/5.8.8/i386-linux
```

This is caused by a mismatch between caching modules but shouldn't cause any serious problems.

One of the most common error messages when starting up a catalyst site is the following:

```
[Fri Sep 07 15:52:15 2007] [error] Couldn't instantiate component "PfamWeb::Model::PfamDB", "->confi
[Fri Sep 07 15:52:15 2007] [error] Can't load Perl module PfamWeb for server myserver.org:0, exiting
```

The message suggests that there's a problem with the model but this is usually a sign that catalyst hasn't found the website configuration file. Check that you have the right paths in the website apache configuration.

Retrieved from "http://scratchy.internal.sanger.ac.uk/wiki/index.php/Website_installation_notes"

Categories: PfamGroup | PfamWeb | PfamWebNotes

- This page was last modified 10:18, 10 September 2007.