

This article describes **advanced installation topics** for GBrowse 2.0, such as accelerating GBrowse performance by running it under a persistent environment, configuring the user login and custom track upload system, and restricting access to certain databases and tracks via user authentication.

- For basic GBrowse 2.0 install instructions, see [GBrowse 2.0 Install HOWTO](#)
- For the main GBrowse 2.0 HOWTO article, see: [GBrowse 2.0 HOWTO](#).
- See also: [GBrowse Install HOWTO](#).

Contents

- [1 Running GBrowse under FastCGI](#)
- [2 Running GBrowse under modperl](#)
- [3 Configuring the User Account Database](#)
 - ◆ [3.1 Creating and configuring the user account database](#)
 - ◆ [3.2 Configuring outgoing email](#)
 - ◆ [3.3 The Admin Interface](#)
 - ◇ [3.3.1 WebGBrowse](#)
- [4 Displaying Next Generation Sequencing Data](#)
- [5 Configuring the Uploaded Track Database](#)
 - ◆ [5.1 Using the DBI::mysql backend](#)
- [6 Authentication and Authorization](#)

Running GBrowse under FastCGI

[FastCGI](#) is a web protocol for generating dynamic that replaces the usual CGI mechanism. The first time a script is requested, FastCGI launches that script as a long-running process. Subsequent requests are forwarded to that process. Because startup time is eliminated, the responsiveness of a FastCGI script is greatly improved.

GBrowse is FastCGI ready. To take advantage of this facility, you will need a version of Apache equipped with FastCGI support, and a short Apache configuration stanza for GBrowse. If your Apache was compiled from source, you will need to download and install either the older [mod_fastcgi module](#), or the newer [mod_fcgid](#) module, and install as directed. If you have a binary distribution, please install the appropriate package: *libapache2-mod-fastcgi* or *ibapache2-mod-fcgid* for APT/Debian systems, or "mod_fcgid" for RPM systems. You will also need the Perl FCGI module (*libfcgi-perl* for APT/Debian, *fcgi-perl* for RPM). The mod_fcgid module is more customizable than the older one and has better performance, but has a significant gotcha that you should be aware of if you use custom Perl library paths. See [Recompiling mod_fcgid to avoid truncated Perl library paths](#).

The next step is to create a suitable Apache configuration stanza for GBrowse, if needed. When you configured and install GBrowse, it probably created a default configuration for you, and stored it into your Apache web server's configuration directory (depending on your system, `/etc/apache/conf.d/gbrowse2.conf`, `/etc/apache2/conf.d/gbrowse2.conf`, `/etc/httpd/conf.d/gbrowse2.conf`, or similar). If this file is already installed, then you are all done. Fetch URL <http://localhost/fgb2/gbrowse> and compare the performance to GBrowse running as a CGI script.

If you declined to have the configuration script create this file and configured GBrowse manually, then you can generate a suitable config file by running `./Build apache_conf` from within the GBrowse distribution directory. The FastCGI portion of the configuration file that is generated will look something like this:

```
<IfModule mod_fastcgi.c>
Alias /fgb2 "/usr/lib/cgi-bin/gb2"
<Location /fgb2>
    SetHandler    fastcgi-script
</Location>
FastCgiConfig -initial-env GBROWSE_CONF=/etc/gbrowse2
</IfModule>
```

Copy this stanza into your main Apache configuration file, or, better, into the site configuration file directory, `/etc/apache/conf.d` (or similar). Restart Apache to pick up the configuration file changes and fetch the URL <http://localhost/fgb2/gbrowse>. You should notice a marked improvement in load and update speed. The improvement will be particularly noticeable if you combine FastCGI with a rendering slave, as described in [Running a GBrowse2 Render Farm](#).

If things don't seem to work as advertised, please check the Apache server error log file, usually `/var/log/apache/error_log`, for clues about where things are going wrong.

Running GBrowse under modperl

[ModPerl](#) is an Apache extension that embeds a Perl interpreter in the Apache process. Unfortunately signal handling is inconsistent among different combinations of Perl, `mod_perl`, and Apache, and **mod_perl is no longer supported by GBrowse2**. Please use FastCGI instead.

Configuring the User Account Database

GBrowse offers an optional feature that allows users to register themselves and establish password-protected accounts. When a user logs into her account from any machine, her custom tracks and visual settings (such as panel width) are restored. Otherwise, these settings are lost as soon as the user moves to a different machine or web browser software. In future versions of GBrowse, enabling this feature will allow users to search for public shared tracks and selectively share custom tracks with each other. The user accounts feature also allows users to login using an [OpenID](#), such as one obtained from Yahoo or Flickr, or to associate one or several OpenIDs with their account.

Once user accounts are installed, you may use a special administrator's account to upload and manage public tracks remotely. This eliminates the need to log into the server each time you need to create or modify a track. This facility is described in [#The Admin Interface The Admin interface](#).

There are three steps to getting the user account database going: (1) installing the required Perl modules; (2) creating and configuring the user account database; and (3) configuring GBrowse to send outgoing email.

- Installing the necessary Perl modules

The login module needs to process OpenID transactions. It also needs to send outgoing email, which nowadays frequently requires authentication between the GBrowse web server host and the mail hub. The following additional libraries and modules are required for basic functionality:

Digest::SHA

For creating and storing passwords. Available from [CPAN](#) or as Debian package or as Debian package `libdigest-sha-perl`.

Crypt::SSLeay

For OpenID authentication. Available from [CPAN](#) or as Debian package `libcrypt-ssleay-perl`. This module in turn requires the [OpenSSL package](#), Debian package `libssl-dev`.

Math::BigInt::Pari or **Math::BigInt::GMP**

These libraries speed up `Net::OpenID::Consumer`, and in particular reduce the time needed to run the `Net::OpenID::Consumer` tests. To use the Pari module you will first need to install `libpari` (<http://pari.math.u-bordeaux.fr/>). To use GMP install `libGMP` (<http://gmplib.org/>). Debian users can simply install `libmath-bigint-gmp-perl`.

Net::OpenID::Consumer

For OpenID authentication. Available from [CPAN](#) or as Debian package `libnet-openid-consumer-perl`.

If your preferred mail server requires user authentication to forward outgoing mail, then you will also need the following two modules:

Net::SMTP::SSL

Encrypted connections to mail servers. Available from [CPAN](#) or as Debian package `libnet-smtp-ssl-perl`.

Authen::SASL

Handle the authentication between mail client and server. Available from [CPAN](#) or as Debian package `libauthen-sasl-perl`.

Creating and configuring the user account database

First, choose the desired Perl DBI backend for the user account database. Currently two DBI backends are supported: `mysql` and `SQLite`. `mysql` is appropriate for an environment in which multiple load-balancing GBrowse servers are running because the database can be hosted on a common network-connected server. `SQLite` is simpler to set up, and possibly faster.

To create and initialize a `mysql` login database run the script `gbrowse_create_useraccount_db.pl` which should have been installed when you installed GBrowse:

```
gbrowse_create_useraccount_db.pl \
  -dsn DBI:mysql:gbrowse_login;user=gbrowse;password=gbrowse \
  -admin root \
  -p
```

-dsn specifies the full Perl-style path to a `Mysql` database named `"gbrowse_login"`. GBrowse will access the newly-created database using a username of `"gbrowse"` and a password of `"gbrowse"`.

-admin specifies the `Mysql` administrator's name (this user must have the ability to add and drop databases). You may combine the admin's name and password into a single argument on the command line by following the format `"-admin username:password"`.

-p tells the script to prompt for the administrator's password on standard input. This avoids having to put the password on the command line.

To create a SQLite database you can use the same script with options like these:

```
gbrowse_create_useraccount_db.pl \
  -dsn DBI:SQLite:/var/tmp/gbrowse2/users.sqlite \
  -owner www-data:www-data
```

This creates a user database at `/var/tmp/gbrowse2/users.sqlite`. This database has to be readable and writable by the web user and/or group, and so we must set this ownership using the **-owner** option. If you get a permissions error when you try to create the database, you may need to run the script as the superuser using "sudo" or "su".

Once the database has been created, you need to tell GBrowse about it, by setting the following options in the GBrowse.conf [GENERAL] section:

```
user accounts      = 1
user_account_db    = DBI:SQLite:/var/tmp/gbrowse2/users.sqlite
```

user accounts = 1 activates the login system. Change **user_account_db** to the DBI path specified when you created the database.

Configuring outgoing email

When a user registers an account in GBrowse, the system sends a confirmation message to the email address she provided during registration. When she receives the message, she is prompted to click a link to confirm her account.

To successfully send out this email message, the web server host on which GBrowse runs needs to send a message through an email gateway. You will need to identify the appropriate gateway to use and update GBrowse.conf with its address.

Only SMTP gateways are supported at this time, but almost all organizations and home computers will have access to one. Please see your local IT group, or check your internet service provider's documentation, to find out the hostname, port, and authentication required by your local gateway. As a last resort, you can always create a Google or Yahoo email account dedicated to the use of your GBrowse instance.

Once you have the gateway information, update the **smtp_gateway** option in GBrowse.conf. The full format is *smtp_server:port:encryption:user:password*, but many of the fields are optional. Here is a config that is appropriate for a simple SMTP server that does not require authentication:

```
smtp_gateway = smtp.res.oicr.on.ca
```

Here is a config that is appropriate for a server listening on an unusual port (port 75 rather than the usual 25):

```
smtp_gateway = smtp.res.oicr.on.ca:75
```

Here is a config for a server that requires SSL encryption and listens on the standard ssl-mail port 465:

```
smtp_gateway = smtp.res.oicr.on.ca:465:ssl
```

And here is a config for GMail's gateway, which requires SSL encryption as well as the name and password of a user with a GMail account:

```
smtp_gateway = smtp.gmail.com:465:ssl:joe.user:joepasswd
```

The "encryption" field is one of "plain" or "ssl", defaulting to "plain" (no encryption). The port field defaults to 25 for plain encryption, and 465 for ssl encryption.

If the email step of the registration process does not work, check the server error log to see if there are any messages about the mail being rejected due to bad authentication or inadequate authorization, which could indicate that your SMTP gateway needs SSL or user authentication to accept mail. During debugging, it may be helpful to connect to the user database and periodically empty the "users" table of aborted user registrations by issuing the SQL command "delete from users".

There are several customization options in the user accounts section that you may wish to change. These are:

application_name

When the registration confirmation email arrives, it will appear to have been sent from a user named "GBrowse". You can change this value via this option.

application_name_long

The registration confirmation also includes a longer description of the e-mail sender. You may wish to change this as well.

email_address

This value will change the reply-to address on the confirmation email.

Here is a fully worked example:

```
application_name = Arabidopsis Genome Browser
application_name_long = The Arabidopsis Information Resource
email_address = help@arabidopsis.org
```

The email will now appear to come from a user named "Arabidopsis Genome Browser" and the letter will be signed by "The Arabidopsis Information Resource."

Note that some authenticated SMTP gateways do not allow you to arbitrarily change the apparent sender in order to reduce spam. In this case, the email address will always appear to come from the user whose authentication information was provided in the smtp_gateway information.

The Admin Interface

GBrowse recognizes a special privileged user who can upload and configure tracks that will be public. These tracks become part of the standard list of tracks that everyone sees. By default this user is known as "admin", but you can change the name by editing the value of the "admin_account" option in GBrowse.conf.

Because it is privileged, you cannot use the login registration system to create the account. Instead, run the script "gbrowse_set_admin_passwd.pl" which was installed when you installed GBrowse. It does not need any arguments. Simply run it from the command line, and it will prompt you for a password to use for the admin account:

Displaying Next Generation Sequencing Data

To work with Next Generation Sequencing data in BAM or SAM format, you will need to install the following:

- Bio::DB::Sam from CPAN
 - ◆ This requires the C Samtools library. See <http://cpansearch.perl.org/src/LDS/Bio-SamTools-1.35/README> for details.
- Bio::DB::BigFile from CPAN
 - ◆ This requires the Jim Kent source tree. See <http://cpansearch.perl.org/src/LDS/Bio-BigFile-1.07/README> for details.
- The following binaries installed in your PATH
 - ◆ bedGraphToBigWig, from the Jim Kent source tree.
 - ◆ genomeCoverageBed, from BEDTools <http://code.google.com/p/bedtools/>

Once these are installed, users will be able to upload and view BAM and SAM files, as well as to link to indexed BAM files via the URL interface. See [GBrowse NGS Tutorial](#) for a tutorial on configuration.

Configuring the Uploaded Track Database

When a user uploads or imports a custom track, GBrowse creates a directory containing information about this upload. This directory is stored in the GBrowse temporary path (typically `/var/tmp/gbrowse2/userdata`) under a directory named after the user's session ID, which is a long hexadecimal number. For fast indexed access to the information in the track, GBrowse also creates a secondary database using the BioPerl Bio::DB::SeqFeature::Store genome feature storage system.

A variety of backends are supported, and GBrowse will automatically pick one that your system supports based on Perl's installed DBI drivers. GBrowse prefers the SQLite backend because it requires no additional configuration and has excellent performance. You may wish to install DBD::SQLite if it is not already there.

To force a particular backend, set the `userdb_adaptor` option in GBrowse.conf's [GENERAL] section. Valid options are "memory", "berkeleydb", "DBI::SQLite" and "DBI::mysql". Use the "memory" adaptor only as a last resort, as it has poor performance relative to the others.

Using the DBI::mysql backend

The DBI::mysql backend has excellent performance, but it requires additional configuration to give the GBrowse server the rights to create and destroy databases on the Mysql server.

First, choose a username and password for the GBrowse account. We will use "gbrowse" and "secret" in this example.

Second, use the mysql client application to log into the Mysql server using the db administrator's name and password.

Now, grant the GBrowse user the ability to create and drop databases that begin with the string "userdata_":

```
mysql> grant create on `userdata\_%\`.* to 'gbrowse'@'localhost' identified by "secret";
```

If you are using a Mysql server running on a different host from the web server, replace "localhost" with the name of the web server host, or use the wildcard character "%" to allow GBrowse to connect from any host.

Last, update GBrowse.conf with the appropriate values for **userdb_adaptor**, **userdb_host**, **userdb_user** and **userdb_pass**.

Authentication and Authorization

You can restrict who has access to gbrowse by IP address, host name, domain or username and password. Restriction can apply to the datasource as a whole, or to particular annotation tracks.

To limit access to a whole datasource, you can use Apache's standard authentication and authorization. GBrowse uses a URL of this form to select which datasource it is set to:

http://your.host/cgi-bin/gb2/gbrowse/your_datasource

where "your_datasource" is the name of the currently selected database. For example, the yeast source is <http://your.host/cgi-bin/gb2/gbrowse/yeast>.

To control access to the entire database, create a <Location> section in httpd.conf. The <Location> section should look like this:

```
<Location /cgi-bin/gb2/gbrowse/your_database>
    Order deny,allow
    deny from all
    allow from localhost .cshl.edu .ebi.ac.uk
</Location>
```

This denies access to everybody except for "localhost" and browsers from the domains .cshl.edu and .ebi.ac.uk. You can also limit by IP address, by username and password or by combinations of these techniques. See <http://httpd.apache.org/docs/howto/auth.html> for the full details.

You can also limit individual tracks to certain individuals or organizations. Unless the stated requirements are met, the track will not appear on the main screen or any of the configuration screens. To set this up, add a "restrict" option to the track you wish to make off-limits:

```
[PROPRIETARY]
feature = etc
glyph   = etc
restrict = Order deny,allow
         deny from all
         allow from localhost .cshl.edu .ebi.ac.uk
```

The value of the restrict option is identical to the Apache authorization directives and can include any of the directives "Order," "Satisfy," "deny from," "allow from," "require valid-user" or "require user." The only difference is that the "require group" directive is not supported, since the location of Apache's group file is not passed to CGI scripts. Note that username/password authentication must be turned on in httpd.conf and the user must have successfully authenticated himself in order for the username to be available.

As with other `gbrowse` options, `restrict` can be a code subroutine. The subroutine will be called with three arguments consisting of the host, ip address and authenticated user. It should return a true value to allow access to the track, or a false value to forbid it. This can be used to implement group-based authorization or more complex schemes.

Here is an example that uses the `Text::GenderFromName` to allow access if the user's name sounds female and forbids access if the name sounds male. (It might be useful for an X-chromosome annotation site.)

```
restrict = sub {
    my ($host,$ip,$user) = @_ ;
    return unless defined $user;
    use Text::GenderFromName qw(gender);
    return gender($user) eq 'f';
}
```

You should be aware that the username will only be defined if username authentication is turned on and the user has successfully authenticated himself against Apache's user database using the correct password. In addition, the hostname will only be defined if `HostnameLookups` have been turned on in `httpd.conf`. In the latter case, you can convert the IP address into a hostname using this piece of code:

```
use Socket;
$host = gethostbyaddr(inet_aton($addr),AF_INET);
```

Note that this may slow down the response time of `gbrowse` noticeably if you have a slow DNS name server.

Another thing to be aware of when restricting access to an entire datasource is that that even though the datasource itself will not be accessible to unauthorized users, the name of the datasource will still be available from the popup "Data Source" menu. If you wish the name to be suppressed from view, add the option **hide=1** to the datasource configuration stanza of the `GBrowse.conf` file:

```
[example_datasource]
description = Draft annotation of v. volvulus
path        = v_volvulus_draft.conf
hide        = 1
```

The **restrict** option can also be used in a `[datasource]` section of `GBrowse.conf` to conditionally enable display of the source to certain IP addresses or users. For example, to display the `V. volvulus` source only to users in the `.ebi.ac.uk` domain or to an authenticated user named "Admin":

```
[example_datasource]
description = Draft annotation of v. volvulus
path        = v_volvulus_draft.conf
restrict    = Satisfy any
             Order deny,allow
             deny from all
             allow from .ebi.ac.uk
             require user Admin
```

To completely disable generation of the data sources popup menu, set **show sources=0** in the `[GENERAL]` section of `GBrowse.conf`.