Written for Handlebar 2.0 - 23/04/10

Handlebar deployer guide

Tim Booth, NEBC

http://handlebar.sourceforge.net

Table of Contents

1 Deploying your own Handlebar barcoding system

1.1 Obtaining

Handlebar, the software used for the PG&P barcode initiative, is available for free download from SourceForge.net or the NEBC (http://nebc.nox.ac.uk/projects/handlebar). We recommend you first try out the test database or the latest Bio-Linux Live DVD which contains a demo of the system. You should also read the barcode user guide which describes the features of the system from the point of view of the end user.

1.2 Licence

The barcode manager software may be used and distributed under the same terms as Perl itself.

1.3 Prerequisites

To run your own Handlebar database you will need a web server capable of serving Perl CGI scripts. The recommended system is Debian GNU/Linux with Apache, for which a .deb package is available from the Bio-Linux package repository (http://envgen.nox.ac.uk/envgen/software/archives/000508.html), but the software should run happily on any modern Unix and even Windows (untested) if installed manually. Full instructions for installation onto Mac OS-X are also included in the package.

The software requires a database to store the barcode definitions and metadata. It has been developed for PostgreSQL 7.3/7.4 and will also work with 8.x. It will not currently run on other databases such as MySQL or Oracle.

For non-Debian systems, you will need to make sure you have the following:

- Perl 5.8
- A PostgreSQL server (need not be on the same machine as the web server)
- A working web server with CGI enabled
- These non-standard Perl modules:
    - CGI
    - DBD::Pg
    - Text::CSV
    - DBD::AnyData
    - Config::Simple
    - IO::String
    - Mail::Mailer (only needed for the printing module)
    - Spreadsheet::WriteExcel
    - Spreadsheet::ParseExcel

- For the report maker, the GenQuery software available from the same sources as Handlebar.

1.4 Installation

If you want to use the package for Debian Etch proceed as follows:

1. Ensure that PostgreSQL is installed and working on your system - NB. the package will allow itself to be installed without the PostgreSQL server since you may want to run this on another machine.
2. Add the Bio-Linux repository to your sources as outlined on the page mentioned above.
3. Install the bio-linux-handlebar package, confirming that you want to set up the database.
4. Check that the interface is visible at http://localhost/cgi-bin/handlebar/bc.cgi
5. Edit the file /usr/lib/cgi-bin/handlebar/barcodes.conf to suit your needs
6. Modify the barcode types - see below

If you are not running Debian you will need to make sure that the dependencies above are met, that the CGI scripts are in place and that the stylesheets are saved in a web-accessible location. Further details are given in the INSTALL file included in the tarball.

If you are not using the deb package or you don't want to rely on the quick-and-dirty database setup script in the package then you will want to use the barcodebase.sql file included in the distribution to create the required table structure. Details of PostgreSQL server setup and troubleshooting are beyond the scope of this document.

1.5 Database Setup

You should have a basic grasp of PostgreSQL server administration in order to set up the database. Such material is outside the scope of this guide but the relevant manuals are on the PG site (http://www.postgresql.org/docs/7.4/interactive/index.html). We recommend the use of the cross-platform tool PGAdmin3 (http:// www.pgadmin.org) which gives a full GUI for database administration including a graphical table design tool, which can be used to define new barcode types. The rest of this document will assume you are using PGAdmin3.

1.6 Database Security

By default the database will named *handlebar* and be owned by the user *postgres*, which is the standard PostgreSQL administrative user. The webserver on Debian runs as user *www-data* and so a corresponding user will be created in PostgreSQL. This user has limited access to the database so you do not have to rely on the security of the CGI scripts to ensure system integrity. Handlebar always uses this public account to log into the database.

Currently Handlebar makes no attempt to authenticate users. Anyone can sign up and any user can claim to be any other, so you will want to configure your webserver to limit access to the site. This can be done by adding an .htaccess file requiring a password, or by restricting access to IP ranges, or a more complex scheme. It is assumed that anyone who can get to the site can be trusted not to do anything malicious. Handlebar is, however, designed where possible to stop them doing anything stupid :-).

1.7 The barcodes.conf file

Various parameters of the interface are controlled from the configuration file, *barcodes.conf*. You should read the comments in this file and modify it to suit your needs. This file also contains the database login parameters and options to enable/disable various features.

1.8 Other configuration tasks

Apart from settings in the barcodes.conf file, other things you may want to change are the HTML stylesheet and the page footer.

Three stylesheets are available within the package or you can make your own. You can edit the file /usr/lib/cgi-bin/handlebar/barcodes.footer.html to customise the footer which appears at the bottom of each page.

2 Setting up new types of item

2.1 Sample types provided with the package

The default database comes with several sample item types for which the user can allocate barcodes - eg a soil sample.

The handlebar system is designed to be applicable to a wide range of situations, and you will want to modify the item types so that users will be prompted for relevant information about the items they are barcoding. Each item type is represented by a table in the database living under the *handlebar_data* schema. If you define a new table then a new type will appear and you can assign codes of that type. All the sample tables are based on the *generic* type using the PostgreSQL inheritance mechanism. This is not a requirement (the CGI code is unaware of the inheritance) but it does simplify maintenance and reporting somewhat.

2.2 Defining a new type with PGAdmin3

1. Open up PGAdmin3 and connect to the database.

2. In the tree view on the left, open up the display until you see the *handlebar_data* schema and then the list of tables in this section. Right click on “Tables” and select “New Table”.
3. Give the table a name, which may not contain spaces, and in the inheritance box, have the table inherit from type *generic*.
4. Add data fields by using the “columns” tab.
5. Ensure under “privileges” that user *www-data* has permissions to view and modify the table.

Obviously, this part is where knowledge of PostgreSQL is desirable since the type description must be converted to an SQL table definition. You can use SQL constraints to enforce restrictions on the content of certain fields.

2.3 The *barcode_description* table

You can refine the way the tables are displayed by putting entries into the *barcode_description* table, which lives in the *handlebar_sys* schema. In pgAdmin3 you can right-click on the table and use the “view data” option to view and edit the contents of this table. The format is:

- typename: The name of the table in the data schema or a hyphen (-) to specify a default comment.
- columnname: The name of the column, or a hyphen to give a comment to the whole table.
- notes: Any text you put in here will be associated with the column or table and displayed when you hit *describe type* in the web interface.

You can also add flags to a column or to a whole table. These also go in the description table:

- typename: As before.
- columnname: As before but preceeded by an asterisk - eg. '*notes', or '*-' to refer to the whole table.
- notes: A comma separated list of flags.

Unlike comments, flags are not displayed directly but are taken as hints to the interface software. Also, unlike comments, the flags specific to a table will be applied in addition to the default flags rather than replacing them. Current flags you can use are:

- bc: This column contains a barcode, and the flag simply tells the system to format the number in the standard way and make hyperlinks. The data type should be *int8*.
- demote: Shunt this column to the end of the spreadsheet. Useful for getting the *notes* column in the appropriate place while keeping it in the generic table.
- print: This column is a candidate for printing onto barcode labels.
- noexport: This column will not appear in the Excel export. This is mainly

useful for timestamping, where the default value is supplied by the database. You may also find this useful if you have used table inheritence and an optional column in the parent type is irrelevant to the child type.
- hide: (whole table only) Indicates that a type is not in use and should be omitted from the dropdown list of types which may be allocated.

More flags may be added if a need arises.

2.4 The *last_update* timestamp

The generic table includes an *auto_timestamp* field which is set to default to the current time (according to the database server, not the client PC). This field will not appear in the downloaded Excel files but will be shown when the barcode is queried, indicating the time of last update.

2.5 Additional Notes

- All table and column names should be specified without spaces. Underscores will be transformed to spaces by the interface so they are hidden from the user. Likewise, barcodes are stored as simple numbers in the database but will be formatted for display according to the parameters set in the configuration file.
- If you want to remove (drop) a table then ensure that no barcodes of that type have been allocated. If barcodes are allocated the correct approach is to set the *hide* flag on the table and dispose all the allocated codes so they cannot be used.
- It is easy to append columns to an existing table using either the PGAdmin3 tool or an SQL command of the form 'ALTER TABLE ... ADD COLUMN ... NULL'. Even if data has already been entered into the table the system will handle this cleanly and the new heading will appear the next time the user retrieves a spreadsheet. If you want the new field to be mandatory then, once all existing records have been amended, you can run a command like 'ALTER TABLE ... ALTER COLUMN ... SET NOT NULL'.

  (see http://www.postgresql.org/docs/7.4/interactive/ddl-alter.html)

3 System Maintenance and Other Internals

3.1 Modifying the database directly

The operations you can perform through the web interface are limited. You may want to modify the database directly in some cases - for example to change user details, change ownership, undelete barcodes, change comments. Remember that some changes can potentially create a conflict within the database or between the database and a spreadsheet which has been exported already. As of version 2.0, Handlebar will expose some of these administrative tasks via the web interface in a “safe” manner via the “Extra Admin” menu item.

The information you will need to change is within the tables that live in the *handlebar_sys* schema. Apart from the *barcode_description* table you will find three other tables:

- *barcode_user* contains details of users who register via the website. You can add users manually and modify details. Do not rename or delete a user who has allocated codes or the database will be upset.
- *barcode_allocation* contains one line for each block of barcodes allocated. Take great care if you try to edit this table manually.
- *barcode_deletion* contains one line for every disposed barcode. You may want to un-dispose a barcode by simply deleting lines from the table, and you can also tweak the comments if need be.

3.2 User names and the checking thereof

There is a configuration option which controls if strict username checking is enabled on the site. As stated in the user guide, the system will believe that an user with access to the site is whoever they claim to be, with no authentication required. The point of prompting for a username is to stop one user unwittingly altering or disposing codes belonging to another. If strict username checking is turned off, users will be able to dispose of codes and upload data without giving a user name first, which is convenient but the verification safeguard is removed.

3.3 Code Allocations

The database currently issues 8-digit codes which are shown in the form 00-001234, the first two digits identifying the database instance to which the code belongs. The 8-digit size was chosen because this size of code fits neatly on the side of an Eppendorf tube; it is not a fundamental feature of the database. The size and format of the codes is controlled by the configuration file on the server.

The database will allocate codes according to the following rules:

1. Each new block must not be larger than the maximum size specified in the configuration.
2. If there are no codes in the database yet, the starting code will be taken from the config file. By using a different offset for each database (eg 20000000 in the default configuration) the codes are kept unique across all projects.
3. Else the highest currently allocated code will be determined, then 1 is added to this. Finally the figure is rounded up to the nearest 10 (or whatever is set in the configuration file) and this becomes the base for the next allocation block.

Codes which slip between blocks will simply go unused. Codes are allocated in strict order, and once the allocation is made the type and size of the block should not be changed (this can only be done by direct editing of the database in any case).

4 Resources and Support

4.1 Online resources

The main page for the project is http://handlebar.sf.net. Other resources are linked from this page.

Updated documentation will be added to the Wiki pages at http://darwin.nox.ac.uk/pgp-wiki/index.php/Barcode_user_guide and visitors are welcome to add to or modify these Wiki pages for the benefit of other users.

A public SubVersion repository, downloads and other features can be found on the SourceForge.net project page at http://sourceforge.net/projects/handlebar.

4.2 Support from the NEBC

We will endeavour to answer questions related to the Handlebar software and to provide updates and bug fixes. The preferred way to communicate with the developers is through the support forums and mailing lists on SourceForge.net.

For NERC-funded researchers we may be able to provide technical support or even host a Handlebar database for your group. See the NEBC data policy, available on our site, or contact us directly for more information on the NEBC remit.

4.3 Contributing

We have plans to improve the Handlebar software with various new features and improved usability. All updates will be made available for free as part of the public source code. If you are able to contribute your time and skills to Handlebar development then we would love to hear from you.