Home Coop:

Tiny Local Consumer Cooperatives Web Ordering System

Installation Guide

Prerequisites:

- Linux apache2 web server
- MySQL 5.X and up server
- PHP 5.3 and up. Packages: libxml2-dev, php5-mysql, php5-xsl, libxslt1.1

Installation:

1) Create a database schema by running **DB\00_Schema.sql** in MySQL server. In case the database was already created, you will need to omit the first CREATE DATABASE statement and possibly also the USE HomeCoop statement (due to permission restrictions on hosting sites).
2) Run in MySQL the initial data script **DB\01_Base.sql.** Note: this script creates a first user for the website. Login name: **admin**. Password: **123456**
3) Run in MySQL a language script for each language you wish your website to support (only 2 languages strings are provided – English and Hebrew, but you can easily enough add other languages). If you choose to include English, start with **DB\02_English.sql** so other languages will "fall" to English when a string is missing.
4) Run **DB\11_ProduceConstants.sql** in a MySQL client. Export the result to a spreadsheet file, and copy the entire column content (excluding the first header row) to **Website\$keys\class\consts.php**, replacing the constants under the comment DATABASE-MATCHING CONSTANTS.
5) Configure settings set in **Website\strings.xml** and each **Website\strings.[language folder name].xml** file (at the beginnings of the files only). Language folder name is usually composed of two letters, such as "en" for English.
6) Configure settings set in **Website\$keys\settings.php**. Note: each string in the format <$!SOMETHING$!> is a placeholder that will be used by **Website\create-language-folders.py** to create a folder for each language the website supports. The string values for this placeholders are found in the **Website\strings.[language folder name].xml** files

and in the file **Website\strings.xml**, which includes some non-language-specific strings and functions as a cross-development-language "constants file". The file **Website\$keys\coopbrief.htm** points to strings that describe organizational information (100% Vegan, etc.), presented to anyone on the website login page. Edit those strings in the xml files mentioned here and modify the file **Website\$keys\coopbrief.htm** to your liking.

7) For a one-language deployment other than English (which the website is already set for by default), set DEFALUT LANGUAGE in **Website\index.php** to the supported language folder name, and **$g_aSupportedLanguages** in **Website\$keys\settings.php** to a two-dimensional array with one child array – for the supported language.

The following code, when included in **Website\$keys\settings.php,** sets English as the only supported language (regardless of how many SQL language scripts were run and provided that **DB\02_English.sql** was one of them):

```
//language folder (array key),
//language name,
//is a required language? (false=optional),
//rtl/ltr align
//language id (from db)
//falling language id
$g_aSupportedLanguages = array(
                    'en' => array('English', true, 'ltr', 1, 0)
 );
```

8) For a two or more languages deployment, set the value for the system's default language folder in the root directory file **Website\index.php** below the comment DEFALUT LANGUAGE. The website's root directory should then be the folder **Website.**

9) Follow the instructions for using **Website\create-language-folders.py** in the document **Multi-language Website Generating Python Script.pdf** and run the script to generate language-specific/one-language deployment folder(s)

10) Copy, perhaps through an FTP connection, the contents of the directory **Website,** without the **create-language-folders.py** script related files and $keys folder, to your website's designated root directory path. This will leave only **Website\index.php**, **Website\uploadimg (or the name it was changed to in Website\$keys\settings.php)** and a language folder for each language your website will support.

11) Set write permissions for the following folders:

- **Website\[langauge folder name]\cache** - to enable caching of catalog.php when accessed publicly: sudo chmod 777 [cache path]. Caching can be deactivated in **Website\$keys\settings.php**.

- **Website\uploadimg (or the name it was changed to in Website\$keys\settings.php) -** to enable uploading product images: sudo chmod 777 [uploadimg path]
12) At this point you should be able to browse to the site's index.php page with username:admin, password:123456 and start looking around. To learn more about specific features – see the user guide: **userguide.pdf**.

Bulk Insert of Members

Two files were added to the installation package to facilitate easy bulk insert of all members.
The file **DB\Tools\MembersBulkInsert.ods** is a sample spreadsheet that can generate a script for each member row included in it. Copy your members information (name, emails and balance) to the file.
Use "paste special..." to copy values (and not formulas) from the Suggested User Name column to the User Name column and from the Password Generator column to the Password column.
NOTE: login name (user name) is not changeable from HomeCoop user interface (to allow identification of the member, even after hir name was changed). Suggested User Name contains an extraction of the first email address' first part. Consider setting the User Name manually instead of copying it from the Suggested User Name column.

After User Name and Password columns were set correctly, the Script column will contains a MySQL insert query for each member. Copy the first Script value to the file **DB\Tools\MembersBulkInsert.sql** instead of the placeholder "[Insert here first member script]". Copy the rest of the members scripts instead of the placeholder "[Insert here all other members script]".

Run the updated **DB\Tools\MembersBulkInsert.sql** in MySQL to bulk-insert all the members with default member permissions. Sign in to HomeCoop to set any higher permissions for specific members.

Roles Configuration:

If you are only interested in simple member permissions allocation just enter the system (as an administrator), go to Members, and set roles for those who should have coordinating rights – see **userguide.pdf** for details. The following section is written for the situation when you find this method not good enough.

Member roles are, in fact, pretty easy to re-configure, for a MySQL developer.
The file **DB\Tools\RolePermission.sql** includes a MySQL query that returns all the roles and the permissions included in them.
The file **DB\Tools\Permissions.sql** includes all the system's supported permissions.
Notice that when nAllowedScopeCodes equals 1 it means that there's only a cooperative-wide permission. When nAllowedScopeCodes equals 3 there's also a coordinating-group permission level, in which the member will only have access to specific records, as set in their or a related configuration page.

For example, the permission to update producers is defined with nAllowedScopeCodes = 3. This means that if a member has only group level permission to modify producers (as members with the role "Producer Coordinator" have – see the result of **DB\Tools\RolePermission.sql**), ze will be able to update only those producers that are set for her in their coordination page, or, in this case, those that ze hirself created.
The producer coordination also cascades to its products, so products too have nAllowedScopeCodes = 3 setting. The permission to update orders (Orders – Update) is only in use for updating a member's own order, so there's no group level for it. The permission to delete your own order is included in the permission to update it, so the Orders – Delete permission is again with nAllowedScopeCodes = 3 setting, to differentiate between coordinators who can delete any member order, and those who can delete only those that they coordinate, as set in the cooperative order coordination page. The rest of the permissions are pretty straight-forward to follow according to their names.

A script to add a permission to a role: **DB\Tools\AddPermissionToRole.sql**
A script to remove a permission from a role: **DB\Tools\RemovePermissionFromRole.sql**
A script to change group-level/coop-wide setting (scope): **DB\Tools\SetScope.sql**