CMPE-272-02

Cross-Site Registration Guide

Will Maynard
2 December 2017

1. Introduction

To meet the requirements for our final project, our group must create five e-commerce websites with cross-site user registration. When we register a user on any member's domain, we must send registration information to the rest of the group.

This guide walks you through the setup and usage of the PHP functions involved. This guide will take approximately 15 minutes to complete.

2. Prerequisites

Before you set up cross-site registration, ensure that your site meets the following prerequisites:

- Your website is properly configured for PHP 7
- Your database is initialized
- Your database has a 'Users' table conforming to the group standards

For the guides related to these steps, refer to the following documents in the shared Google Drive folder:

- PHP 7 Setup Guide for IIS
- PHP 7 Setup Guide for Apache
- SQL Server Setup Guide for Windows Server 2012
- MySQL Setup Guide
- SQL 'Users' Table Creation

3. Setup

3.1. Add Provided PHP Files to Your Website

From the project library repository, copy the following PHP files to the **/api/** directory on your website:

- curlRegister.php
- dbInfo.php
- externalRegister.php
- registrationSandbox.php

3.2. Edit dbInfo.php to Reflect Your Environment

Open dbInfo.php in a text editor or IDE. In the *Connection String Variables* section, there are four variables that need to match your environment: $serverAddress, $username, $dbName, and $dbPassword.

Edit these directly for debugging purposes. Your file should look similar to this:

```
/*****************************************\
|        Connection String Variables      |
\*****************************************/
$serverAddress  = 'bfgmarket.com';
$username       = 'wmaynard';
$dbName         = 'CMPE272'
$dbPassword     = 'NotARealPassword';
```

Important note on security: After you verify that the script works, it's strongly recommended you input your database information from a file that's not included in your VCS and inaccessible to your IIS / Apache installation. Always avoid committing sensitive information to your repositories!

3.3. Verifying the Database Connection

In a web browser, navigate to **{your website}/api/registrationSandbox**. After loading the page, the PHP script will have created a randomly-generated user for each group member who has successfully completed this guide.

If the script fails at any point in this process, a detailed error message and stack trace will be displayed. Otherwise, the page will display the generated user's username, email, and timestamp of creation.

Optional: You may further verify the user was successfully created by running the following query on your database and looking for the newly-created user:

```
SELECT ID, Username, dtDateCreated
FROM Users
ORDER BY dtDateCreated DESC
```

Important note on security: After verifying the scripts work, delete registrationSandbox.php. Due to its nature, a malicious script could flood every group member's database with test users by repeatedly loading the page.

3.4. Verifying the Cross-Site Registration

To verify the user was created on other member's sites, you may log in to any group member's site with the username of the generated user and the default password for generated users: "Correct Horse Battery Staple".

3.5. Add the Cross-Site Registration Function Call to Your Site

First include the new curlRegister.php at the top of the file currently handling your user registration:

```
include_once '$_SERVER['DOCUMENT_ROOT'].'/api/curlRegister.php';
```

Next, modify the function that previously registered your users. Remove any existing database calls and replace it with the following function call, passing in the appropriate values:

```
curlRegister($email, $name, $surname, $password, $homePhone, $cellPhone);
```

That's it! Congratulations on implementing cross-site registration!

4. How It Works

1. A user clicks the 'New Account' button on your website.
2. The user fills out the registration form and clicks the 'Register' button.
3. curlRegister() uses BCRYPT to hash the user's password.
4. curlRegister() sends a POST request to each group member's externalRegister.php with user information, starting with the local server.
5. externalRegister.php verifies the request originated from whitelisted domains.
6. externalRegister.php validates the data contained in the request.
7. externalRegister.php loads database information from dbInfo.php.
8. externalRegister.php inserts a record into its associated database.

Failed registrations can be viewed in /api/failedRegistrations.log.

5. Pros & Cons of This Approach

Advantages:

- We never need to query other sites to see if a user already exists.
- Each registration costs one cURL hit per user per server.
- We can keep our databases hardened from outside connections.
- We don't need to share a central database, allowing more flexibility for other site features.
- We can all use the same registration files with minimal modifications and maintenance.

Disadvantages:

- If a group member's site is down for any reason, that site will miss the registration.
    - Can be remedied with a scheduled task to retry failed registrations. See the document "Introduction to Cron Jobs" for more information.
- Individual site registration requirements are bypassed (i.e. valid usernames, password requirements).
- Each member site must use a similar structure for their Users table.
- Users cannot be deleted without going out of sync with the group.