Nov 20, 2014

CheckoutCrypto Documentation

Table of Contents

Nov 20, 2014

CheckoutCrypto Documentation

Nov 20, 2014

CheckoutCrypto Documentation

Nov 20, 2014

CheckoutCrypto Documentation

Introduction to Infrastructure

wikipedia.org

Nov 20, 2014

CheckoutCrypto Documentation

Explanation

CheckoutCrypto previously made use of a cluster of vps servers scattered across the internet, even utilizing a local, home PC. This barely, if at all, met bare minimum requirements, for what the company needed, to succeed with the release of V2, further allow growth in an advantageous environment.

Problems encountered

Each member of CheckoutCrypto needs to utilize the same environment, in order to test the functionality and design of the site. In order to accomodate this, as well as the need for open development in groups, we decided each person would need their own cloned copy of the worker, api, site and demo. The problem was, how do we accomplish this in the shortest amount of time, while extending the development capabilities of the infrastructure as whole, while being more efficient and cost effective.

Solutions Found

The solution was to pursue a dedicated host, in order to spin our own custom VPS' using our own custom ubuntu VMs prenetworked, preinstalled with all the repositories and software needed to develop with CheckoutCrypto.

Employee environment

Each employee's VM consists of:

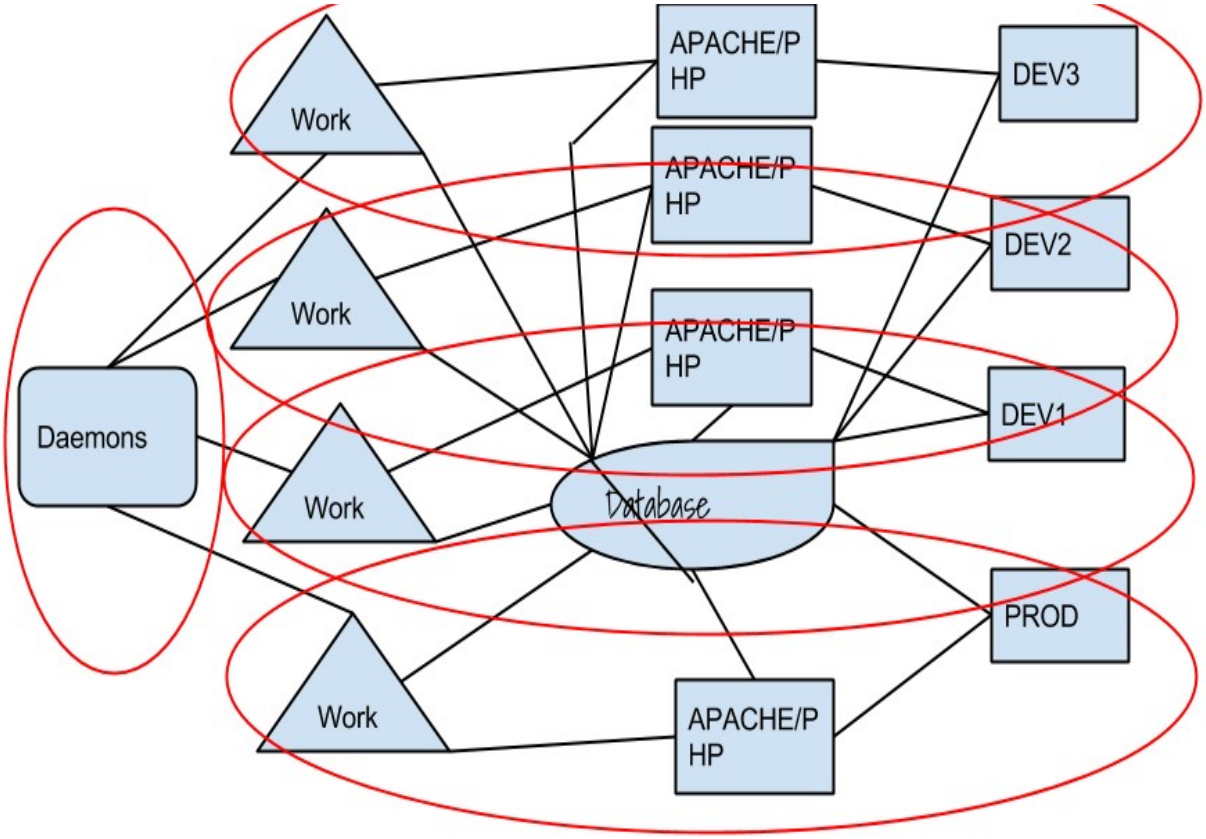
Ubuntu(Linux) 64 bit VM
1 gig of ram
1 cpu thread
Mate Desktop, Samba, qt5, qtcreator, LAMP, phpmyadmin, x2go plugins, chrome, synapse(use F3),

A CheckoutCrypto menu is accessible with the command: cgmenu
Site, API, Worker, Demo, Cron, preinstalled.

Nov 20, 2014

CheckoutCrypto Documentation

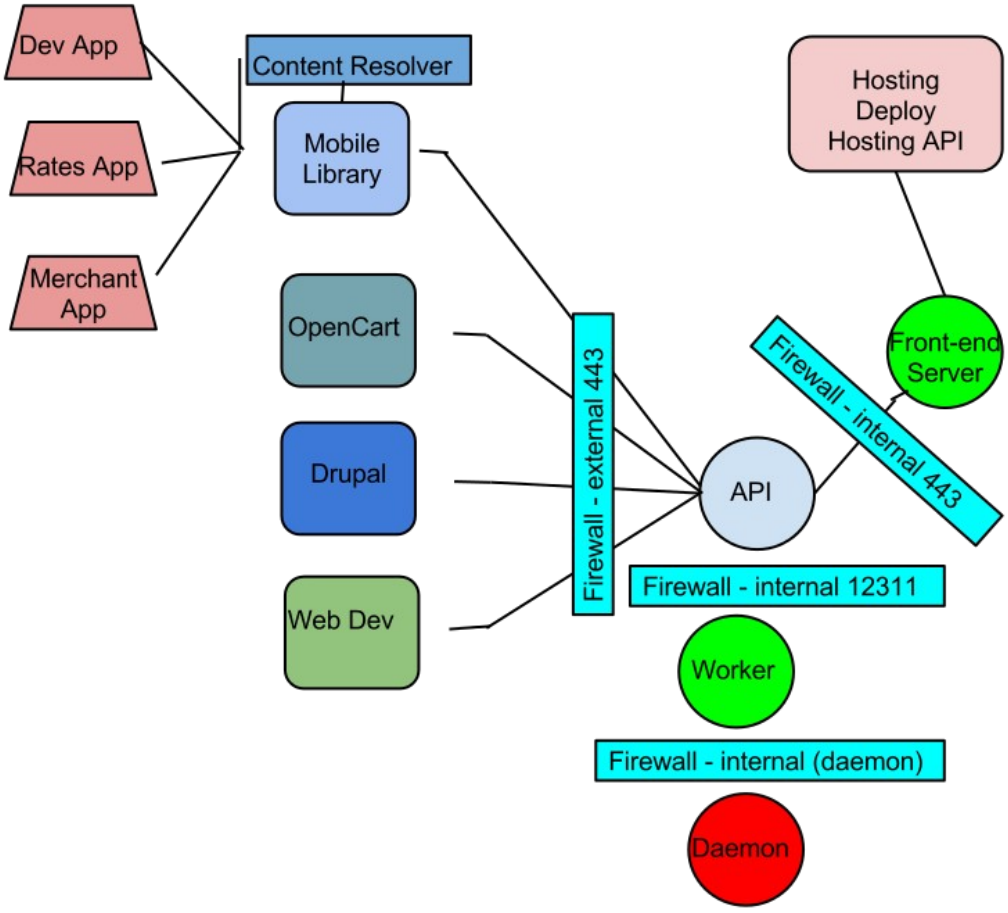
Employee Development



Nov 20, 2014

CheckoutCrypto Documentation

Merchant/Client Development



Nov 20, 2014

CheckoutCrypto Documentation

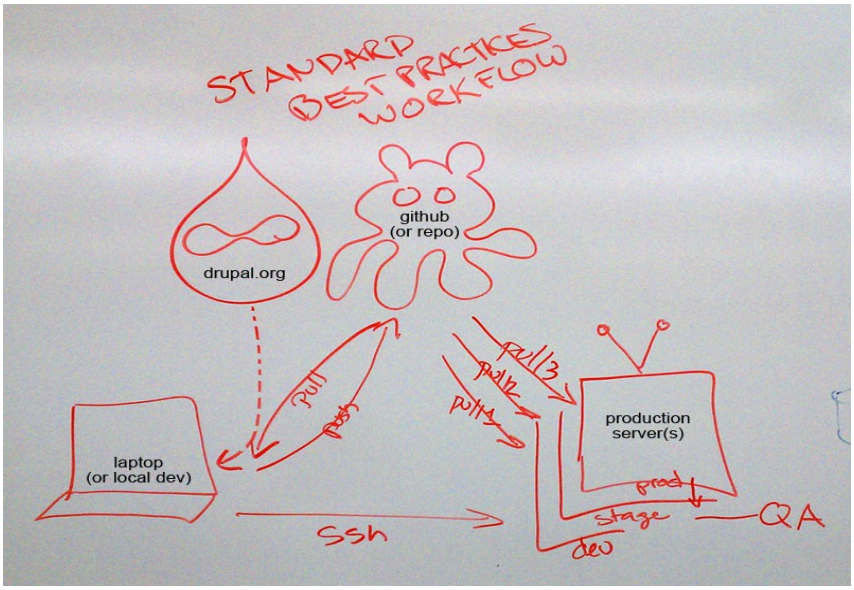
Introduction to Repositories

Github

Nov 20, 2014

CheckoutCrypto Documentation

Workflow



Explanation

Each developer of CheckoutCrypto maintains their own seperate branch, locally on their developer VPS. This way, they're free to modify the site how they please. When they're ready, their code is merged to the 'Master' branch, which is then pulled to our 'Production Server'. Hot fixes can be applied directly to Master branch, though they will be rare.

Git Hub Repositories

SITE https://github.com/CheckoutCrypto/site
API https://github.com/CheckoutCrypto/crypto-api
WORKER https://github.com/CheckoutCrypto/worker
DEMO https://github.com/CheckoutCrypto/checkoutcrypto-drupal

Nov 20, 2014

CheckoutCrypto Documentation

Commands

git clone SOME_REPO_URL (download an entire repo to the current dir)
git pulll (download changes to checked out branch)
git add somefile (stage a file(s) for commit) (add -A for all)
git commit -m “some commit message” (commit some code, give it a message)
git push origin SOMEBRANCH (push a commit to a branch)
git checkout SOMEBRANCH (switch branches)
git merge SOMEBRANCH (merge branch into the one you have checked out)

Developer VM directories where repositories are pre-installed and the suggested git add command:

site - /var/www/dev/site
demo - /var/www/dev/demo
api - /var/www/api
worker - ~/repos/worker

Introduction to Linux

Introduction

A quick, efficient, easy to use, environment, is essential for your development on any coding

related project. This ensures you accomplish the most, in the shortest amount of time, leaving you more time for actual creation!

The preferred choice by most developers, Linux is primarily used because of the open: philosophies, licensing, tools, everything required for an operating system, fully customizable for each individual project i.e. you aren't stuck with the same looking desktop, your desktop can be customized and developed on easily! The fact it comes ready, in just a few short commands (which I will get into in a moment), with all the necessary tools one would need to develop an enormous variety of software, is the primary reason for this tutorial.

A brief lesson on Linux history would be beneficial to hear.
https://www.youtube.com/watch?v=5ocq6_3-nEw

Preparation

1) Time, you will need to ensure you have a few hours to spare, you could potentially lose any previous data on your harddrives if you aren't careful.

2) A blank DVD and Burner + HardDrive space OR an external drive. In other words, you will need to install on your main hard drive OR utilize an additional hard drive to boot to linux.

3) The preferred linux distribution for CheckoutCrypto is Ubuntu 14.04 64 bit. If you're more familiar with another flavor of Linux you're open to utilizing it, we will only provide support for debian based distributions internally and publicly. You can download the latest version of ubuntu for your processor at ubuntu.com

4) their is a possibility your hardware may be incompatible with ubuntu, for one, make sure you have the compatible video driver (not covered) and last make sure your wireless card is compatible(most if not all usually are, this was a problem in the past). Bluetooth may require some fiddling depending on the make, model, brand. Google is your friend. A phone or another computer is handy if you run into issues.

Ubuntu 14.04 Installation

Nov 20, 2014

CheckoutCrypto Documentation

Option 1:

1.Download Ubuntu Image and burn image to disc.
2.Start up computer, boot to disc.
3.Run the installer setup (it will take a bit to start, needs to load liveCD into RAM)
4. Skip To Step 5 from Option 2 below.

Option 2:

1. Format external drive (backup contents elsewhere if need be)
2. Download image and extract it to the external drive. It will need to format and create 2 partitions (ext4 and swap).
3.Restart the computer, boot to external hard drive.
4.Run the installer setup (it will take a bit to start, needs to load liveCD into RAM)

5. If you want to install along side windows, make sure you set that option when it asks you.
6. If you want to install on a blank drive, click that option when asked.
7. When asked how much for swap, set it to double the size of the computer's RAM. So if you have 4 G of RAM set the swap size to 8 G
8. Make sure you allow proprietary codecs, and updates like mp3 etc.
9. When ready hit install, set your timezone and user account settings, as it installs and updates.

don't forget your root password, don't make it to easy or to long, you will use it constantly!

Success! You have entered the world of Linux.

Restart, make sure you can login. Explore your new desktop. Download additional software from the Ubuntu Software Centre.

Ubuntu also offers the ability to signin with different environments such as CairoDock. You may wish to click the ubuntu logo next to your login name, when you first boot it up (before you enter password), if you want to change your desktop environment. CairoDock is what I prefer, but you may prefer the default Ubuntu Unity, to each their own.

Basic Linux Commands

Nov 20, 2014

CheckoutCrypto Documentation

Any command that requires admin permissions must start with sudo.
The syntax goes: skynet@skynet:~$ sudo anycommand -anyargument

~	- means relative to the current user's home directory
sudo	– super user(root) do something
su	– super user
cd	- change directory
chmod	- set a file or directories user/group permissions (read, write, execute)
chown	- set a file or directory user/group owner
mkdir	- create a directory
rm	- delete file
rmdir	- remove directory
touch	- create blank file
tail	- read the end of a file (-n 200 will give you last 200 lines)
ls	- list files (-li gives permission info)
cp	- copy files (cp onefile ./somedirectory/onefile)
scp	- copy files across network to a specific user (scp onefile skynet@somewhere:/home/skynet/somedirectory/onefile)

Absolute Paths
skynet@skynet:~$ cd /home/skynet/Pictures ----------|
|
Relative Paths | Same Path
skynet@skynet:~$ cd ./Pictures --------- |
skynet@skynet:~$ cd ~/Pictures --------- |

Google “Linux Cheat Sheets” for more detailed linux shell syntax.

LAMP(Linux,Apache,Mysql,PHP) Server Installation

Open a Terminal (ctrl+ alt + t)

Nov 20, 2014

CheckoutCrypto Documentation

sudo apt-get install tasksel
sudo tasksel install lamp-server

enter your current root password
sets a root password for your MySQL database

Phpmyadmin setup and usage

sudo apt-get install phpmyadmin
enter your current root MySQL database password (set in LAMP step)
connect to phpmyadmin by opening a browser and going to http://127.0.0.1/phpmyadmin
login with user: root password: (the mysql pass you entered in LAMP setup).

Additional Software – Optional

note: Most of these can be installed via terminal: sudo apt-get install SomeApplication
The rest can be found in the ubuntu software centre.

Eye Candy/Efficiency

Synapse(set to a start on boot, set activate hotkey to something you will use constantly), CairoDock, Compiz Config Settings Manager(careful, test everything you modify)

Video Drivers

Nvidia/ATI proprietary drivers - check the additional drivers section of your ubuntu software centre, if not nvidia.com or ati.com

Development IDEs (interface development environment)

Eclipse (develop for everything, install additional plugins, eclipse.org)
VIM, gedit, nano

2D/3D Image Edit/Create/Animate

Gimp, Blender, Inkscape
gimp-gap (animation plugin)
gimp paint studio (https://code.google.com/p/gps-gimp-paint-studio/)

Browser (firefox is already included)

Nov 20, 2014

CheckoutCrypto Documentation

Introduction to PHP

PHP Tutorial

Cheat Sheet

Nov 20, 2014

CheckoutCrypto Documentation

$anything = variable

echo $anything = print variable can also echo 'whatever'; with quotes

combine variables using . Or +

```
$variable = 1;
$variable2 = $variable * 2;
$variable2 = $variable + $variable;
$variable = “bob”;
$variable2 = “joe”;
$variableResult = $variable.” “.$variable2;   // result: bob joe

function whatever($someParameter){   =  a function
        echo $someParameter;
}

whatever($anyVariable);    =  a function call

foreach($skittles in $bag)  = conditioned loop

if/else  = condition

switch(condition) { case 1: }  = condition

<?php     =  beginning a php section of code
?>       =  ending a php section of code

////   =  a comment
/*   to  */  = commented section
```

Example

Write a standard html page, we're going to seperate and template our sections with php. Remember to always tab in at least i indent, that way it's easier to read.

Example.html

```
<html>
<head><title>PHP Example</title>
 <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
        <div class=”main”>  <!-- Total Outter page box -->
                <div class=”header”><p>Header/logo here</p></div>
                <div class=”navigation”><p>Navigation bar here</p></div>
                <div class=”content”><p> Some content here</p></div>
                <div class=”footer”><p>Footer here</p></div>
        </div>
</body>
</html>
```

Nov 20, 2014

CheckoutCrypto Documentation

Create a CSS style called style.css

```
/*
*  Remember, each class inherits the parameters of the class it resides in.
*/


.main {
	width:1024px;
}

.header {
	height:12px;	/// used to be a rule of thumb, do not make the header/navigation too large
	color:teal;
}

.navigation {
	height:12px;	/// used to be a rule of thumb, do not make the header/navigation too large
	color:red;
}

.content {
	height:300px;
	color:white;
}

.footer {
	height:12px;	/// used to be a rule of thumb, do not make the header/navigation too large
	color:grey;
}
```

Now make sure it all works by visiting http://vpn.local/wpie/Example.html to make sure it works.

Open a new text document, we're going to template the above page, so we can use the layout on every single page we need to write.

```
<?php   // start the php document

function my_header(){
?>

/// we'll insert our html here!

<?php
}

?>  // close the php document
```

Nov 20, 2014

CheckoutCrypto Documentation

What we're doing here is we're open and closing the php code where necessary. PHP and HTML work together to display the page.

Now continue the trend with all the other parts of the page, like this:

```
<?php   // start the php document

function my_header(){
?>

/// we'll insert our html here!

<?php
}

function my_navigation(){
?>

/// we'll insert our html here!

<?php
}

function my_content(){
?>

/// we'll insert our html here!

<?php
}
function my_footer(){
?>

/// we'll insert our html here!

<?php
}

?>   // close the php document
```

Finally, we can copy parts from our original Example.html into the appropriate sections.

```
<?php   // start the php document

function my_header(){
?>

<html>
```

Nov 20, 2014

CheckoutCrypto Documentation

```
<head><title>PHP Example</title>
 <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
        <div class=”main”>  <!-- Total Outter page box -->
                <div class=”header”><p>Header/logo here</p></div>

<?php
}


function my_navigation(){

?>

<div class=”navigation”><p>Navigation bar here</p></div>

<?php
}

function my_content(){
?>

<div class=”content”><p> Some content here</p></div>

<?php
}
function my_footer(){
?>
                <div class=”footer”><p>Footer here</p></div>
        </div>
</body>
</html>

<?php
}

?>   // close the php document
```

Now that we have our layout completed, save the php file as: layout.php
Open another text document, this will be our index page.

```
<?php  /// open the php document

include('layout.php');    /// include the layout.php file we just wrote, if the path is different, adjust it here

my_header();  /// call the header
my_navigation();  /// call the navigation
my_content();  /// call the content
my_footer();  /// call the footer

?>  // close the php document
```

Nov 20, 2014

CheckoutCrypto Documentation

Save the file as index.php You can view the page at http://vpn.local/wpie/index.php

EXAMPLE 2

prerequisite: previous example 1.

The bonus is, now you can make unlimited pages very quickly, while only changing what actually exists in my_content.

Open another text document, this will be our new example.php page.

```
<?php  /// open the php document

include('layout.php');    /// include the layout.php file we just wrote, if the path is different, adjust it here

my_header();  /// call the header
my_navigation();  /// call the navigation
my_content($_GET['type']);  /// call new content type with parameter e.g. vpn.local/wpie/example.php?type=
my_footer();  /// call the footer

?>  // close the php document
```

Save it as Example.php.
Open another text document, this will be our new func.php library. We'll simply fill this document with any extra functions we need.

```
<?php  /// open the php document

function NewStuff(){

        echo “some new stuff displayed, on a new looking page”;
}

?>  // close the php document
```

Save it as func.php

Open layout.php from previous example 1.

add this line below the <?php opening line:

```
include('func.php');
```

then edit/replace the my_content function so it looks like this.

```
function my_content($someParam){
```

Nov 20, 2014

CheckoutCrypto Documentation

```
        if($someParam == “new”){
        echo '<div class=”content”>';
        NewStuff();
        echo '</div>';
        }else{
                echo '<div class=”content”><p> Some original content here</p></div>';
        }
}
```

Save it. View it at http://vpn.local/wpie/example.php?type=new then try without param
http://vpn.local/wpie/example.php?type=somethingwhatever

Introduction to MySQL

Nov 20, 2014

CheckoutCrypto Documentation

Cheat Sheet
CREATE
SELECT
INSERT
UPDATE
DELETE

varchar someColumnName(charactersize e.g. 100);
int someColumnName
datetime - specific columnn for timestamps

Introduction:

MySQL is one of the most commonly used, open-source, database projects. MySQL is fast and highly versatile. Unfortunately it's not the preferred database for mobile, that would be SQLite. That won't be covered here, but they're very similar in syntax(in most cases).

No matter what software you're using to connect to mysql, whether its c/c++, java, javascript or php, you will always need to start by opening a connection with your mysql user, password and database. After each connection you will need to close the connection or risk opening multiple connections at once.

MySQL Administration PHPMYADMIN

You already have a user created for you, login to phpmyadmin (our mysql administration panel), to create a database.

In your browser go to : **http://127.0.0.1/phpmyadmin/**

You can create a database, name it whatever you like

For now, let's try filling this database with some tables! Still in phpmyadmin, click the SQL tab, here you can enter direct SQL queries.

What we did earlier, in order to give you a user account with a database granted with permissions:

```
CREATE USER 'wizardpie'@'localhost' IDENTIFIED BY  '***';

GRANT USAGE ON * . * TO  'wizardpie'@'localhost' IDENTIFIED BY  '***' WITH
MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0
MAX_USER_CONNECTIONS 0 ;

CREATE DATABASE IF NOT EXISTS  `wizardpie` ;
```

Really I just add you in the user tab and check marked “create table and grant permissions” but this is the actual syntax of the command above.

Now to fill it with tables you can add them manually in phpmyadmin, or even through ssh. For now we'll use the preferred method most CMS use for their modules. We'll create one SQL file to do all our database preparation work (for when our example gets redistributed, etc).

Configure Mysql

make sure you have a database, with permissions e.g. you're able to sign in from phpmyadmin

Nov 20, 2014

CheckoutCrypto Documentation

Put MySQL Login Variables in seperate file called dbconfig.php like so:

```
<?php

Class ccDbConfig {

   function config() {
      $itm['driver'] = 'mysql';
      $itm['host'] = '127.0.0.1';   /// or any other database location
      $itm ['database'] = 'yourdatabase';
      $itm['username'] = 'youruser';
      $itm['password'] = 'password';

      if(isset($itm)) {
         return $itm;
      } else {
         return 'false';
      }
   }

}

?>
```

What we're doing here is creating an object/class, called “ccDbConfig” within that object is 1 function, within that function we have 1 array and 1 condition, witin that array we have 5 variables. The condition makes sure the array contains data, if so return data, if not return false.

Connecting to MySQL

In another file called database.php copy the following:

```
<?php

Class ccDb {
   function connectDb() {
      include_once('dbconfig.php');
      $c = new ccDbConfig();
      $ccDbConfig = $c->config();
      $ccDb = new PDO($ccDbConfig['driver'].":host=".$ccDbConfig['host'].";dbname=".
$ccDbConfig['database'], $ccDbConfig['username'], $ccDbConfig['password']);
      return $ccDb;
   }
   /// add queries here
```

CheckoutCrypto Documentation

```
}
?>
```

What we're doing here is initializing a new object “$ccDbConfig” with all our configuration settings that we originally put in dbconfig.php. This allows us the ability to utilize the same mysql settings repeatedly without having to rewrite them, also the best part is it isolates them incase you move or change any mysql settings. This is called encapsulation.

PREPARED MYSQL QUERIES

CREATE TABLE

First we're beginning our connection to sql by calling our “connect” function which handles all mysql configuration. Then we're conducting a mysql query to create the table.

```
   function makeATable() {
      try {
         $ccDb = $this->connectDb();
         $stmt = $ccDb->prepare("CREATE TABLE Persons(FirstName CHAR(30),LastName
CHAR(30),Age INT)");
         $stmt->execute();
         return $ccDb->lastInsertId();
      } catch (exception $e) {
         echo $e;
      }
      return false;
}
```

INSERT

from the above file (database.php) adding the below function

```
   function addToTable($firstname, $lastname, $age) {
      try {
         $ccDb = $this->connectDb();
          $stmt = $ccDb->prepare("INSERT INTO Persons (FirstName,LastName,Age) VALUES
(:f_name,:l_name,:age)");
         $stmt->bindValue(':f_name', $firstname, PDO::PARAM_STR);
$stmt->bindValue(':l_name', $lastname, PDO::PARAM_STR);
$stmt->bindValue(':age', $age, PDO::PARAM_STR);
         $stmt->execute();
         return $ccDb->lastInsertId();
```

CheckoutCrypto Documentation

```
        } catch (exception $e) {
            echo $e;
        }
        return false;
    }
```

Now in order to utilize this refer to “Implementation” below or continue adding other queries.

UPDATE

```
    function amendATable($firstname, $lastname, $age) {
        try {
            $ccDb = $this->connectDb();
              $stmt = $ccDb->prepare("UPDATE INTO Persons (LastName,Age) VALUES (:l_name,:age )
WHERE FirstName = :f_name");
            $stmt->bindValue(':f_name', $firstname, PDO::PARAM_STR);
$stmt->bindValue(':l_name', $lastname, PDO::PARAM_STR);
$stmt->bindValue(':age', $age, PDO::PARAM_STR);
            $stmt->execute();
            return $ccDb->lastInsertId();
        } catch (exception $e) {
            echo $e;
        }
        return false;
    }
```

updates last name and age of based on the user's firstname.

SELECT

```
    function getATable($firstname) {
        try {
            $ccDb = $this->connectDb();
              $stmt = $ccDb->prepare("SELECT Age, FirstName, LastName FROM Persons WHERE
FirstName :f_name)");
            $stmt->bindValue(':f_name', $firstname, PDO::PARAM_STR);
            $stmt->execute();
            $row = $stmt->fetchAll(PDO::FETCH_ASSOC);     /// grab all the queried Rows in a nice array
        } catch (exception $e) {
            echo $e;
        }
```

CheckoutCrypto Documentation

```
    if(is_array($rows) AND count($rows) == 1) {
        $row = $rows[0];
    } else {
        return false; //this shouln't happen
    }
    if(isset($row['Age'])) {
        $result['age'] = intval($row['Age']);
        $result['fname'] = strtolower($row['FirstName']);
     $result['lname'] = strtolower($row['LastName']);
    }
    if(isset($result)) {
        return $result;
    }
```

Select the age, firstname,

Implementation

make a file called my_queries.php copy below.

```
<?php
   function makeTable(){
        include_once('database.php');
          $db = new ccDb();
          $tableResult = $db->makeATable();
  }

 function fillTable(){

        $firstname = “joe”;
        $lastname = “average”;
        $age = 26;

         include_once('database.php');
         $db = new ccDb();
         $tableResult = $db->addToTable($firstname, $lastname, $age);
 }

?>
```

Nov 20, 2014

CheckoutCrypto Documentation

You can now call this from any php file by include('my_queries.php'); Then makeTable(); or fillTable(); anywhere!

```
<?php
include('layout.php');   /// if you followed our earlier php tutorial.
include('my_queries.php);

my_header();   /// if you followed our earlier php tutorial.
makeTable();
fillTable();
my_footer();   /// if you followed our earlier php tutorial.

?>
```

Another way is to simply put the database.php object in it's own class(object) and call it elsewhere if you want to resuse the same queries or need to sanitize (e.g. our API).

```
Class ccApi {
    function dbConnect() {
        if(!(isset($db))) {
            include_once('database.php');
            $db = new ccDb();
        }
        return $db;
    }

    function makeTable(){
           $db = $this->dbConnect();
            $tableResult = $db->makeATable();
   }

   function fillTable($fname, $lname, $age){
         $db = $this->dbConnect();
          $rates = $db->addToTable($fname, $lname, $age);
   }
}
```

CheckoutCrypto Documentation

Drupal Site Modules

ccAccount

Nov 20, 2014

CheckoutCrypto Documentation

ccAccount.install

Table: ccdev_account
Columns**:** basic_id, bundle_type, user_id, walletname, api_key, isMaintenance, isFrozen, total_balance, total_transactions, default_coin, verified, delay_seconds, created

ccAccount.module

function ccAccount_user_register_form_submit(&$form, &$form_state)

parameters: form variables, form_state
description: Modifies user registration form such that an apikey is automatically generated on registration form submit. Immediately creates: an OTP password(needs debugging) row, a balance row (for that new user, with one specific coin(BTC)), then it tries to create a new address in the wallet, thus create a new account in the daemon for BTC under that user's new generated wallet. We generate a walletname during this step, we do not assosciate walletnames directly with names or user ids, besides this table(ccdev_account).

function ccAccount_form_alter(&$form, &$form_state, $form_id)

parameters: form variables, form_state, form id
description: Modifies User Registration form submit button, in order to link to our custom submit options

function validate_user_callback()

description: Validate Account details for ccAccount.js, this was for a specific checkoutcrypto theme in which the plan was to display user account details in a popup menu. No longer in use.

function ccAccount_entity_info()

description: Implements hook_entity_info().

function ccAccount_settings_form()

description: Modifies Admin Configuration menu adding an account generator section for use with generating multiple specific global site api keys

function ccAccount_settings_form_validate($form, &$form_state)

parameters: form variables, form_state
description: Validates Admin Configuration Account generator section for use with generating multiple specific global site api keys

function ccdev_accounts_uri($basic)

Nov 20, 2014

CheckoutCrypto Documentation

parameters: basic (node id)
description: creates a dynamic URI path, where all new specific rows will be apended to(e.g. if you made a new row in the Account it could be viewed on its own page with the link Account/basic/SOMEID

function ccAccount_menu()

description: Creates links for each page within the module. Gives permissions and sets parameters for each link, in addition to a page title for the browser top.

function ccAccount_info_page()

description: Callback page for the 'Account' link

function ccAccount_billing_page()

description: Callback page for the 'Account/Billing' link

function ccAccount_summary_page()

description: Callback page for the 'Account/Dashboard' link

function ccAccount_permission()

description: A list of Module permissions, accessible from admin menu -> permissions

function ccdev_accounts_list_entities()

description: Returns a render array with account entities.

function ccdev_accounts_title($entity)
parameters: entity
description: Callback for a page title when this entity is displayed.

class ccAccountBasicController

public function create()
public function save()
public function delete()

Nov 20, 2014

CheckoutCrypto Documentation

lib.inc

function ccInsertMaintainers($users)
parameters: users (array of user names)
description: Insert a new API key for each of the global admin api accounts e.g. if you want site specific keys for specific modules, this generates the keys from admin -> configuration -> Account keys e.g. rate, refresh etc. These are only if we need to differentiate between users and the site.

function ccGenApiKey($userid, $update, $coin = FALSE)
parameters: userid, update(true/false [update/insert]), coin(default)
description: Generate a key for a specific user ID, update with a new key if you already have a key

function updateAccountCoin($userid, $coin)
parameters: userid, coin (default)
description: update a user's default account coin

function getAllAccountInfo(){

description: Query all the user's ccAccount data, using the user's unique drupal user id.

function writeBillingSummary()

description: Create a new billing summary row for the user, e.g. assume they will have a new summary for their bills to be tallied, each user needs to have a new row when they register.

function getBillingSummary($userid)
parameters: drupal userid
description: Query all user's billing summary data from row(billing tally), based on their drupal user id.

function checkIfKeyExists()
global user
description: Query if a user has generated a key yet, returns true/false.

function getServerKey()

description: Query a Server Key, generated for site authentication to our api(also known as maintainer api keys).

function addEmailTrans($userid)
parameters: drupal userid
description: Add a single interval to a user's email Transaction count – for limitations and counts

Nov 20, 2014

CheckoutCrypto Documentation

function getGroupInfo($grpid)
parameters: unique group id
description: Query the group data and all columns for a group package, based on the group's unique id (row identifier)

cc.inc

This is the main connection from our site to the API, it takes in parameters such as which call and what variables to make, then it utilizes cURL to contact our api locally, conduct the same form of call a regular user can do. Hence the need for specific private calls, utilizing private, site, apikeys.

```
class CheckoutCryptoAPI {

  public function query($params)

   $base_url = 'http://127.0.0.1/api/api.php';
    $arguments = '?apikey='.$apikey;

'getnewaddress',
'send',
'pendwithdraw',
'sendfunds',
'getstatus',
'getbalance',
'gettransaction',
'getreceivedbyaddress',
'getrate',
'refreshworker',
'gettradeaddress',
'gettradestatus',
'gettradereceived'

  function urlRequest($url)
```

CheckoutCrypto Documentation

ccAdmin

Nov 20, 2014

CheckoutCrypto Documentation

ccAdmin.install

Table: ccdev_admin
Columns**:** basic_id, bundle_type, 'disable_all_coins', 'disable_worker' , 'disable_transaction', 'disable_withdraw', 'disable_getnewaddress', 'disable_getbalance', 'disable_rate' , 'worker_status'

ccAdmin.module

For now the admin module functions as a regular vanilla module, complete with a controller, links, permissions, etc, however, none of it is used at the moment, this is for future control, monitoring, stabilizing maintenance of the platform, such that you can shut down parts of the API when necessary.

This table is checked every API call. Use it to shut things down, change default 0 to 1 on any column.

ccBalance

Nov 20, 2014

CheckoutCrypto Documentation

ccBalance.install

Table: ccdev_balance
Columns**:** basic_id, bundle_type, uid, coin_name, coin_code, coin_pending, coin_withdraw, coin balance, coin_autopay, coin_autoaddress, updated

ccBalance.module

function ccBalance_entity_info()

description: Node's entity info is linked from here

function ccdev_balance_uri($basic)

description: A single balance can be linked with dynamic links e.g. Balance/basic/SOMEBALANCEID

function ccBalance_preprocess(&$variables, $hook)
parameters: variables, hook
description: this is an overridden theme hook for functionality that runs prior to the page load (the header basically)

function ccBalance_menu()

description: this links to a function containing all the links within the menu override hook.

function ccBalance_info_page()

description: Links to a function containing the display content for a basic balance info page, on this page we actually go further and link to a table to display all a user's coins.

function ccBalance_block_info()

description: Links to a function containing the summary of our balance block and how, when to display it.

function ccBalance_block_view($block_key)
parameters: block key
description: Links to a function containing the block content for our balance block, so that we can utilize our balances anywhere we'd like within our template. Our balance block actually utilizes a balance table

Nov 20, 2014

CheckoutCrypto Documentation

function ccBalance_permission()

description: Links to a function containing the balance module's permissions, whether they can add, edit, remove, balances (nodes of this content).

function ccdev_balance_title($entity)
parameters: entity id
description: The title displayed on top of brower, on any node page of this module.

function ccdev_balance_view($entity, $view_mode = 'tweaky')

description: The module's view, within a menu callback e.g. admin menu, this code would enable such functionality, though it isn't in use yet.

function ccBalance_autopay_callback($ajax, $coincode = NULL)
parameters: ajax true/false, coincode (default = NULL)
description: Callback for autopay popup, ctools, checks to see if OTP row was created

function ccBalance_withdraw_callback($ajax, $coincode = NULL)
parameters: ajax true/false, coincode (default = NULL)
description: Callback for withdraw popup, ctools, checks to see if OTP row was created

class ccBalanceBasicController

public function create()
public function save()
public function delete()

lib.inc

function _ccBalance_make_link($args = '')
parameters: args, pass param in link, to generate
description: Make a link to the withdraw balance popup

function _ccBalance_make_link_autopay($args = '')
parameters: args, pass param in link, to generate
description: Make a link to the autopay balance popup

function getUserApiKey()

description: Query for the user's API key

function getSpecificKey($userid)

Nov 20, 2014

CheckoutCrypto Documentation

description: Query for a specific API key based on the drupal user id

function getServerApiKey($walletname)
parameters: walletname

description: Query for a server(maintainer) API key, based on the walletname

function getBalance($userid, $coincode)
parameters: userid, coincode
description: Query for a single user's coin balance based on a coin code

function calcTotal($amt, $coincode, $userid)
parameters: amt, coincode, userid
description: Calculate what the total withdraw will cost, based on the user's group ID (which package/group they've been assigned or paid for)

function autopay_insert($coin, $amount, $address)
parameters: coin, amount, address
description: Insert a new autopay amount, address, coin for a single drupal user's coin balance

function ccValidateTwoFact($pin, $key)
parameters: pin, key
description: Validate a 2factor OTP is correct, based on the generated 2fa key for a user.

function checkMaxTrans()

description: Check if a user has reached a limitation based on Max Transaction count

api.inc

function ccWithdrawRequestTwoFact($apikey, $twofa, $RECIPIENT, $amount, $coin_code)

parameters: apikey, 2fa, recipient_address, amount, coin_code
description: Withdraw Funds, using this api request, with the params, and apikey, this is for 2fa prefered clients

function ccWithdrawRequest($apikey, $RECIPIENT, $amount, $coin_code)

parameters: apikey, 2fa, recipient_address, amount, coin_code
description: Withdraw Funds, using this api request, with the params, and apikey, this is for email OTP

Nov 20, 2014

CheckoutCrypto Documentation

prefered clients

function ccSendFundsRequest($apikey, $RECIPIENT, $amount, $coin_code)

parameters: apikey, 2fa, recipient_address, amount, coin_code
description: Send Funds by email, using this api request, with the params, and apikey

forms.inc

function autopay_form($form, $form_state)

parameters: form, form_state
description: generate and return content for popup autopay form (ctools)

function withdraw_form($form, $form_state)

parameters: form, form_state
description: generate and return content for popup withdraw form (ctools)

function OTP_form($form, $form_state)

parameters: form, form_state
description: Display OTP error if no OTP row set for user (no authentication preference for this user)

function ccdev_balance_form($form, $form_state)

parameters: form, form_state
description: create a basic node form for the data in the ccdev_balance table, basically, with this function one could create a vanilla balance for debug etc, mainly for use with the default data controller.

forms_sub.inc

function autopay_form_validate($form, &$form_state)

parameters: form, form_state
description: Validate Amount, Address, balance, OTP, etc, from autopay form

function autopay_form_submit($form, &$form_state)

parameters: form, form_state
description: Submit validated autopay form data to the API

Nov 20, 2014

CheckoutCrypto Documentation

function withdraw_form_submit($form, &$form_state)

parameters: form, form_state
description: Submit validated withdraw form data to the API

function withdraw_form_validate($form, &$form_state)

parameters: form, form_state
description: Validate Amount, Address, balance, OTP, etc, from withdraw form

function OTP_form_submit($ajax, $data)

parameters: ajax, data
description: dismisses OTP error

function ccdev_balance_form_validate($form, &$form_state)

parameters: args, pass param in link, to generate
description: validate basic default balance form for use with datacontroller. (not presently in use)

table.inc

getBalanceTable()

description: Query all the rows from the balance(ccdev_balance) entity, display in a row

CheckoutCrypto Documentation

ccCoin

ccCoin.install

Nov 20, 2014

CheckoutCrypto Documentation

Table: ccdev_coin
Columns**:** basic_id, bundle_type, coin_name, coin_code, coin_rate, coin_rate_btc, coin_fee, coin_txfee, coin_enabled, min_amount, max_amount, coin_community, coin_validate, coin_image, coin_description, added

ccCoin.module

function ccCoin_entity_info()

description: Links to a function which returns all the necessary module entity info

function ccdev_coin_uri($basic)

parameters: basic node coin id
description: Creates a dynamic URI for each coin created, giving it a unique page link e.g. Coin/basic/SOMECOINID

function ccdev_fiat_uri($basic)

parameters: basic node fiat id
description: Creates a dynamic URI for each fiat created, giving it a unique page link e.g. Coin/basic/SOMEFIATID

function ccCoin_menu()

description: Links to a function which returns the menu links necessary for the menu hook.

function ccCoin_block_info()

description: A summary of information for how to display our Coin Block

function ccCoin_block_view($block_key)

parameters: block_key
description: A function which contains the contents for our Coin Block. This content contains a table of data

function ccCoin_preprocess(&$variables, $hook)

parameters: variables, hook
description: functions that need to run in header to modify the popup theme before our page content is loaded

function ccCoin_info_page()

Nov 20, 2014

CheckoutCrypto Documentation

description: The ccCoin info page callback, links to a function with the page content

function ccCoin_permission()

description: links to a function containing module's permissions for add/edit/create coins

function ccdev_coin_title($entity)

parameters: entity id
description: browser title for each page of this module

function ccdev_coin_view($entity, $view_mode = 'tweaky')

parameters: entity id,
description: creates a seperate view for ccCoin in the event we want to add a coin via a admin menu or another place where we want a view.

class ccCoinBasicController

```
 public function create()
 public function save()
 public function delete()
```

lib.inc

function addUserOTP($userid, $coin_code, $coin_name)

parameters: userid, coin_code, coin_name
description: creates a single row in ccdev_otp for use with a user's OTPs(one time passwords, authentication requests, withdraws etc).

function addUserBalance($userid, $coin_code, $coin_name)

parameters: userid, coin_code, coin_name
description: creates a single row in ccdev_balance for the user's coin they have enabled. E.g. enable LTC litecoin balance created. This row is referenced frequently for pending_withdraw, pending_balance, checkbalance cron calls

function getLastUser()

Nov 20, 2014

CheckoutCrypto Documentation

description: get last userID created in drupal users table

function _ccCoin_make_link($args = '')

description: creates a link to the enable coin popup.

function _ccCoin_make_edit_link($args = '')

description: creates a link to the enable coin popup.

function getCoinData($coincode)

parameters: coin_code,
description: Queries the ccdev_coin row based on a coin_code and returns a large array of all the specified coin's data.

function getAllCoinData()

description: Queries the ccdev_coin table for all rows, all columns, for all coins, returns a 2D array of coins+columns.

function getCoinTxFee($coincode)

parameters: coin_code
description: Query the tx(miner's) fee for a specfic coin_code.

function getCoinRate($coincode, $amount)

parameters: coin_code, amount
description: Calculate total based on rate and amount of, a specific coin.

function getAccountData($coincode)

parameters: coin_code,
description: Query the coin's balance for a user.

function getAllCoinCodes()

description: Query all coin_names, coin_codes, coin_images,

function ConvertAmount($amount, $amount_type, $coin_to, $coin_from)
parameters: amount, amount_type, coin_to, coin_from
description: Calculate amount of coin incoming to coin outgoing

Nov 20, 2014

CheckoutCrypto Documentation

function getSpecificCoinImg($coin)

parameters: coin_code,
description: Query a specific coin image.

function getIndividualCoinImg($coins, $coincode)

parameters: coins, coin_code,
description: Retrieve a coin image from an array of coins.

function getSpecificCoinRate($coin)

parameters: coin_code,
description: Query a specific coin_rate, coin_rate_btc

function getCoinValidateCode($coin)

parameters: coin_code
description: Query a specific coin_validate code (address validation code)

api.inc

function ccGetInitAddress($userid, $coin)

parameters: drupal userid, coin_code
description: Query API for an address for the user, based on a specific coin.

function ccGetAddress($coin)

parameters: drupal userid, coin_code
description: Query API for an address for the user, based on a specific coin.

function ccApiStatus($queue_id, $coin)

parameters: queue_id, coin_code
description: Query API for status of an API call, based on a specific coin.

function ccRefreshRequest() {

description: Query API layer, to notify worker to refresh its cache (as we've changed something it relies on at initialization).

ccGroup

ccGroup.install

Nov 20, 2014

CheckoutCrypto Documentation

Table: ccdev_groups
Columns**:** basic_id, bundle_type, grp_name, grp_description, grp_max_transactions, grp_max_emails, grp_cost, grp_type, grp_payment_length, grp_size, grp_SKU, created

ccGroup.module

function ccGroup_entity_info()

description: Links to a function containing the basic entity info for group content

function ccdev_groups_uri($basic)

parameters: node id,
description: creates a dynamic URI for each row/node of the group module

function ccGroup_menu()

description: Links to a function containing the basic group module page links necessary for the menu hook

function ccGroup_permission()

description: Links to a function containing the group permissions e.g. add/edit/create/view groups

function ccGroup_info_page()

description: Links to a function containing the content for the main group info page

function ccGroup_admin_page()

description: Links to a function containing the content for the group admin menu page

function ccdev_groups_load($basic_id = NULL, $reset = FALSE)

parameters: basic id
description: necessary hook to load the group's node's into an entity, we don't utilize this anymore

function ccGroup_manage_callback($ajax, $group_id = NULL)
parameters: ajax (true/false), group id
description: Group manage ctools popup for adding/editing a user's group

function ccdev_groups_title($entity)

Nov 20, 2014

CheckoutCrypto Documentation

parameters: entity id,
description: creates a title at the top of the browser for every page/node of the group module

function ccdev_groups_view($entity, $view_mode = 'tweaky')

parameters: entity id,
description: Links to a function containing the basic module page links necessary for the menu hook

function ccdev_groups_add()

description: opens a group-add form based on the variables listed in ccGroupBasicController-> create()

function ccdev_groups_form_validate($form, &$form_state)

parameters: form, form_state,
description: validates variables from a group-add form

function ccdev_groups_form_submit($form, &$form_state)

parameters: form, form_state,
description: submits variables from group add form to database -> insert group

class ccGroupBasicController

public function create()
public function save()
public function delete()

form.inc

function ccdev_groups_form($form, &$form_state, $entity)

parameters: form, form_state,
description: Creates a custom group-add form, for use with ctools in a nice clean “add group” popup.

function ccGroup_manage_form($form, $form_state)

parameters: form, form_state,
description: Creates a custom group-edit form, for use with ctools in a nice clean “manage group” popup.

form_sub.inc

Nov 20, 2014

CheckoutCrypto Documentation

function ccGroup_manage_form_submit($form, &$form_state)
parameters: form, form_state,
description: submits variables from group-edit form to database -> update group

lib.inc

function _ccGroup_make_link($args = '')

description: Make a custom ctools link for group manage popup

function ccGroupRefresh()

description: Refresh all groups on worker, this is a call to the API-worker wake up interface, except it only tells worker to refresh its cache!

function getGroupData($grpid)

parameters: group id,
description: Query for a specific group based on the group id.

function getAllGroups()

description: Creates a custom group-edit form, for use with ctools in a nice clean “manage group” popup.

function getUserGroup($userid)

parameters: user id,
description: Query for a specific group based on the user's drupal id.

ccHosting

ccHosting.install

Nov 20, 2014

CheckoutCrypto Documentation

Table: ccdev_hosting
Columns**:** basic_id, bundle_type, user_id, site_name, site_description, site_demon, site_cms, site_admin_user, site_admin_pass, site_mysql_admin, site_mysql_pass, site_mysql_table, group_price, isEnabled, created

ccHosting.module

function ccHosting_entity_info()

description: Links to a function containing the basic entity info for hosting content

function ccHosting_menu()

description: Links to a function containing the basic hosting module page links necessary for the menu hook

function ccHosting_info_page()

description: Links to a function containing the content for the main hosting info page

function ccHosting_permission()

description: Links to a function containing the hosting permissions e.g. add/edit/create/view hosting

function ccHosting_manage_callback($ajax, $site_id = NULL)

parameters: ajax (true/false), group id
description: Group manage ctools popup for adding/editing a user's group

function ccdev_hosting_title($entity)

parameters: entity id,
description: creates a title at the top of the browser for every page/node of the group module

function ccdev_hosting_view($entity, $view_mode = 'tweaky')

parameters: entity id,
description: Links to a function containing the basic module page links necessary for the menu hook

class ccHostingBasicController

Nov 20, 2014

CheckoutCrypto Documentation

public function create()
public function save()
public function delete()

form.inc

function ccdev_hosting_form($form, &$form_state, $entity)

parameters: form, form_state entity id,
description: Basic Form with all the fields needed for adding a new row to ccdev_hosting.

function manage_form($form, $form_state)
parameters: entity id,
description: Basic Form with all the fields needed for modifying a current row of ccdev_hosting.

form_sub.inc

function ccdev_hosting_form_submit($form, &$form_state)

parameters: form, form_state
description: Basic Form submit, save new hosting entity (new row in ccdev_hosting)

function manage_form_submit($form, &$form_state)

parameters: form, form_state
description: Basic Form Edit submit, update row in ccdev_hosting

lib.inc

function _ccHosting_make_link($args = '')

description: create a link for a Hosting Popup Form.

function getSiteData($siteid)

parameters: basic_id
description: Query a row from ccdev_hosting basic on a specific site id.

Nov 20, 2014

CheckoutCrypto Documentation

ccOTP

Nov 20, 2014

CheckoutCrypto Documentation

ccOTP.install

Table: ccdev_otp, ccdev_auth
Columns**: otp:** basic_id, bundle_type, uid, coin_name, coin_code, coin_amount, coin_address, callback_action, secret, data, sent, valid, created
auth: basic_id, bundle_type, user_id, pref_otp, twofa_key, validated, pending_otp, created

ccOTP.module

function ccOTP_menu()

description: Links to a function containing the menu hook, and all links present in the module

function ccOTP_generate_form($form, &$form_state)

parameters: form, form_state
description: Links to a function containing a testing/experimentation(default) otp_generate_form

function ccOTP_validate_form($form, &$form_state)

parameters: form, form_state
description: Links to a function containing the form displayed, on a page, when a user clicks the authentication link (OTP url) sent to the user's profile email.

function ccOTP_form_user_profile_form_alter(&$form,$form_state)

parameters: form, form_state
description: Hook alter, links to a function containing the altered form, for a user's profile, OTP section.

function otp_gauth_form($form,&$form_state)

parameters: form, form_state
description: Links to a function containing a form for google authenticate form. A qr code is displayed, user can then verify they have stored the 2fa key in their google authenticator by generating a new key, and hitting submit to complete the configuration.

function ccOTP_theme()

description: Links to a function containing the OTP mail template and path

Nov 20, 2014

CheckoutCrypto Documentation

forms.inc

function otp_validate_form($form, &$form_state)

parameters: form, form_state
description: Displays a form for an authentication page, after an email link is clicked.

function otp_profile_mod(&$form)

parameters: form,
description: Modifies User Profile form adding a OTP section

function otp_gauth()

description: Displays a google 2fa authenticate form, complete with a qr code to scan into google authenticate, a textfield for the submission.

function otp_gen_form()

description: Displays the default form for generating a new OTP test link

function ccOTP_user_profile_form_submit($form, &$form_state)

parameters: form, form_state
description: Modifies User Profile Submit form, setting(updating/inserting) new OTP preference into database ccdev_otp table

forms_sub.inc

function otp_gauth_form_validate($form, &$form_state)

parameters: form, form_state
description: Validate the 2fa key entered in the OTP 2fa, activation, form.

Nov 20, 2014

CheckoutCrypto Documentation

function ccOTP_validate_form_submit($form, &$form_state)

parameters: form, form_state
description: Validate the OTP url after being clicked and then the page is submitte. Then complete final OTP action, give success message

function ccOTP_generate_form_submit($form, &$form_state)

parameters: form, form_state
description: Generate and insert new OTP, send email to user's profile email

lib.inc

function random_password($length)

parameters: length
description: Generate a random password for OTP

function ccOTP_otp_generate($args)

parameters: args
description: Generate a random OTP URL, encoded with a random string

function ccOTP_otp_insert($args)

parameters: args
description:Insert a new OTP in ccdev_otp and new auth method in ccdev_auth (if user changes from email to 2fa)

function ccOTP_otp_setValid($args)

parameters: args (basic_id, coin_code)
description: Update ccdev_otp, set an OTP as validated

function ccOTP_validateAuth($userid)

parameters: args (user_id)
description: Update ccdev_auth, set an OTP authentication method (2fa/email), set validated

Nov 20, 2014

CheckoutCrypto Documentation

function ccOTP_otp_decode($args)

parameters: args (signature, secret, data)
description: Decode OTP validation URL

function ccOTP_otp_get_auth_url($signature, $basic_id)

parameters: signature, basic_id
description: Finalize OTP validation URL string

function getSpecificApiKey($uid)

parameters: userid
description: get a specific API key

function gen_two_factor()

description: Create 2factor secret key, return string

function validate2Factor($userid, $secret, $twofa)

parameters: userid, secret, twofa
description: Verify a 2fa code, is the correct code for a given key

function get_otp_type($userid)

parameters: userid
description: Query a user's OTP preference type (email/2fa)

function update_otp_pref($otp_pref, $secret, $validated, $userid)

parameters: otp_pref, secret, validated, userid
description: Update ccdev_auth with OTP preference

function set_otp_pref($otp_pref){

parameters: otp_pref
description: Insert OTP initial preference, send authenticated email otherwise, to confirm recipient is in fact owner of the account, before we switch preferences.

function ccOTP_remove($user, $coin, $action, $id = null)

Nov 20, 2014

CheckoutCrypto Documentation

parameters: user, coin, action, id
description: Remove ccdev_OTP row after completion, update ccdev_auth pending_otp = 0

function sanitizeOTP($args)

parameters: args (signature, basic_id)
description: Sanitize and Validate incoming OTP authorization request

function ccOTP_otp_validate($args)

parameters: args (signature, basic_id)
description: Validate, Decode, Query, for an OTP to ensure it's valid

function getSpecificAddressCode($coin)

parameters: coin_code
description: Query a specific coin's validation address code.

function ccOTP_otp_checkExists()

description: Query ccdev_otp to check if a specific user has an OTP at all.

Nov 20, 2014

CheckoutCrypto Documentation

ccSend

Nov 20, 2014

CheckoutCrypto Documentation

ccSend.install

Table: ccdev_send
Columns**:** basic_id, bundle_type, uid, otp_id, recip_email, reicp_name, recip_msg, coin_code, coin_amt, recip_address, sent, retrieved

ccSend.module

function ccSend_entity_info()

description: Links to a function which returns all the necessary module entity info

function ccdev_send_uri($basic)

parameters: basic node, email send id
description: Creates a dynamic URI for each email send created, giving it a unique page link e.g. Send/basic/SOMEEMAILID

function ccSend_preprocess(&$variables, $hook)

parameters: variables, hook
description: functions that need to run in header to modify the popup theme before our page content is loaded

function ccSend_menu()

description: Links to a function containing the menu hook, and all links present in the module

function ccSend_permission()

description: Links to a function containing the email send permissions e.g. add/edit/create/view email send

function ccdev_send_list_entities()

description: Returns a render array with email send entities.

function ccSend_info_page()

description: Links to a function containing the content for the main email send info page

Nov 20, 2014

CheckoutCrypto Documentation

function ccSend_complete_page()

description: Links to a function containing the content for the email send complete page

function ccdev_send_title($entity)

parameters: entity id,
description: creates a title at the top of the browser for every page/node of the email send module

function ccdev_send_view($entity, $view_mode = 'tweaky')

parameters: entity id,
description: Links to a function containing the basic module page links necessary for the menu hook

form.inc

function ccsend_funds($form, $form_state)

parameters: form, form_state,
description: Displays a form with all the inputs for all parameters of the crypto Send Email request.

function ccdev_send_form($form, &$form_state, $entity)

parameters: form, form_state, entity,
description: Displays a base crypto email send form, using the module's basic data controller, (default).

form_sub.inc

function send_funds_validate($form, &$form_state)

parameters: form, form_state,
description: Validate the parameter inputs of the crypto send email popup, including balance, fees, sanitization, etc

function send_funds_submit($form, &$form_state)

Nov 20, 2014

CheckoutCrypto Documentation

parameters: form, form_state,
description: Submit the paramit inputs for the modified send email popup, insert new ccdev_send row

function ccdev_send_form_validate($form, &$form_state)

parameters: form, form_state,
description: Validate the parameter inputs with the default crypto send email form.

function ccdev_send_form_submit($form, &$form_state)

parameters: form, form_state,
description: Submit the parameter inputs with the default crypto send email form using the basic data controller (default)

function send_OTP_form_submit($ajax, $data)

parameters: ajax, data
description: dismiss form, if OTP error.

lib.inc

function getAvailableCoins(){

description: Query for all available coins: coin_name, coin_code, coin_rate, coin_image

function _ccSend_make_link($args = '')

description: Create link to the crypto SendEmail ctools modal popup

function otp_insert($args)

parameters: args(uid, coin_name, coin_code, coin_amount, coin_address, callback_action)
description: Insert a new OTP for this sendEmail request

function writeSendFunds($entity)

parameters: entity
description: Insert each of the entity parameters in a new ccdev_send row

function updateSendFundsOTP($otp_insert, $basic_id)

Nov 20, 2014

CheckoutCrypto Documentation

parameters: otp_insert, basic_id
description: Update ccdev_send row, with an OTP_id

function updateSendFunds($basic_id, $address)

parameters: basic_id, address
description: Update ccdev_send with the address and date of email confirmation.

function get_otp_url($signature, $basic_id)

parameters: signature, basic_id
description: Generate a URL OTP string for echoing in an email request to the recipient, for authentication with OTP.

function writeOTP($entity, $sendFundID)

parameters: entity, sendFundID
description: Generate, update, send OTP request for crypto SendEmail.

function sanitizeEmail($email)

parameters: email
description: sanitize recipient email address

function sanitizeName($name)

parameters: email
description: sanitize recipient name

function sanitizeMsg($msg)

parameters: msg
description: sanitize sender's message

function sanitizeAmount($amt)

parameters: amt
description: sanitize sender's coin amount

function sanitizeAmountSize($amt, $coin_code)

parameters: amt, coin_code
description: sanitize sender's coin amount, specific to coin limitations

Nov 20, 2014

CheckoutCrypto Documentation

function verifyBalance($amount, $coincode, $user)

parameters: amount, coin_code, user
description: Verify an account has enough of a specific coin, to complete the email transaction.

CheckoutCrypto Documentation

ccService

Nov 20, 2014

CheckoutCrypto Documentation

ccService.install

Table: payment_cc
Columns**:** order_id, user_id, grp_id, coin_name, coin_code, coin_address, coin_amount, pay_status, pay_amount, queue_id, queue_address, exchange_rate, timestamp

ccService.module

form.inc

form_sub.inc

lib.inc

Nov 20, 2014

CheckoutCrypto Documentation