CS4313

Deliverable 4

All Database tables
Description of all functionality
All HTML Pages
All PHP scripts
List of copied material and sources

David O Neill 0813001
James McHugh 0052833
Keiran Duggan 0866504

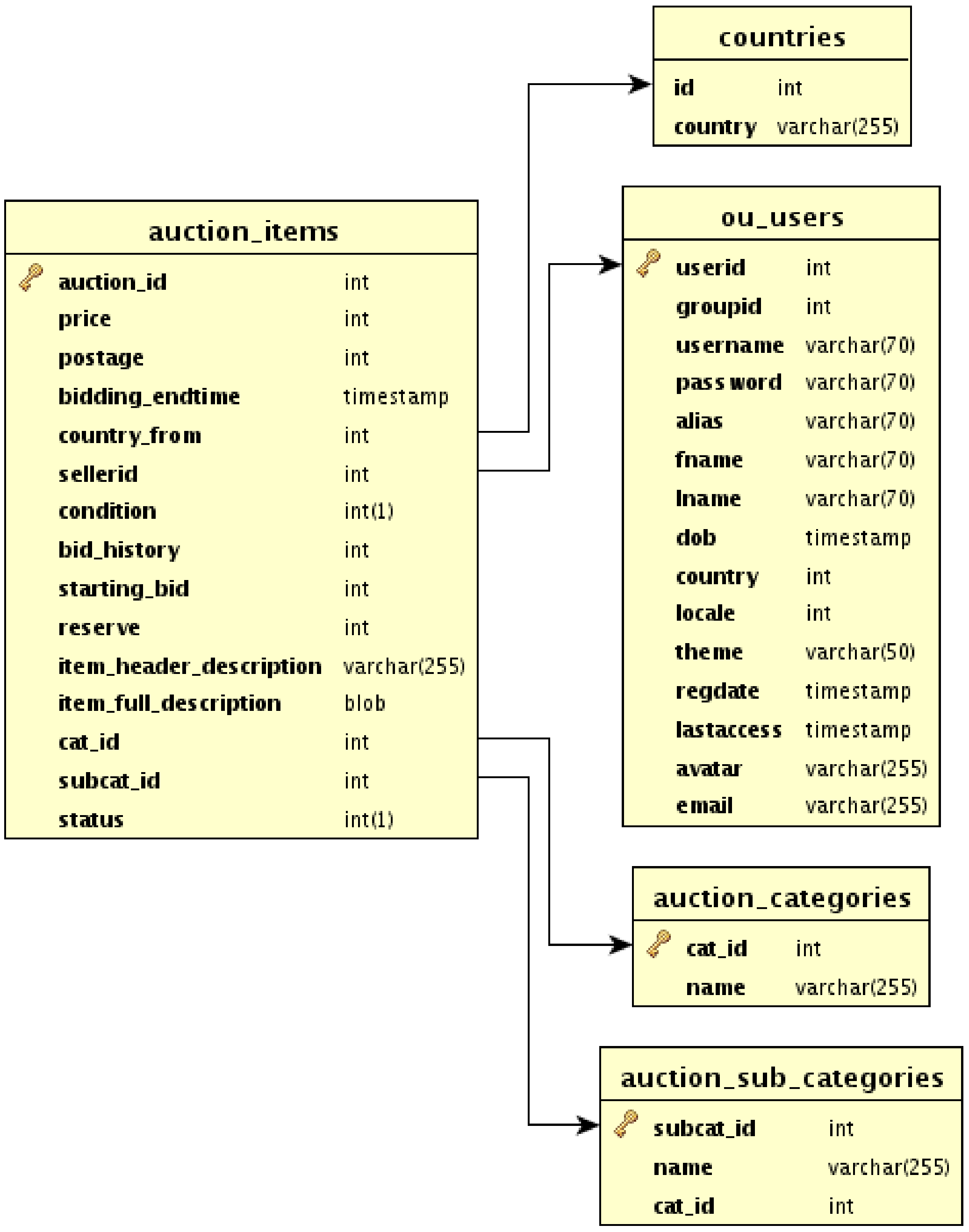
19/4/2010

Table of Contents

Database

Tables

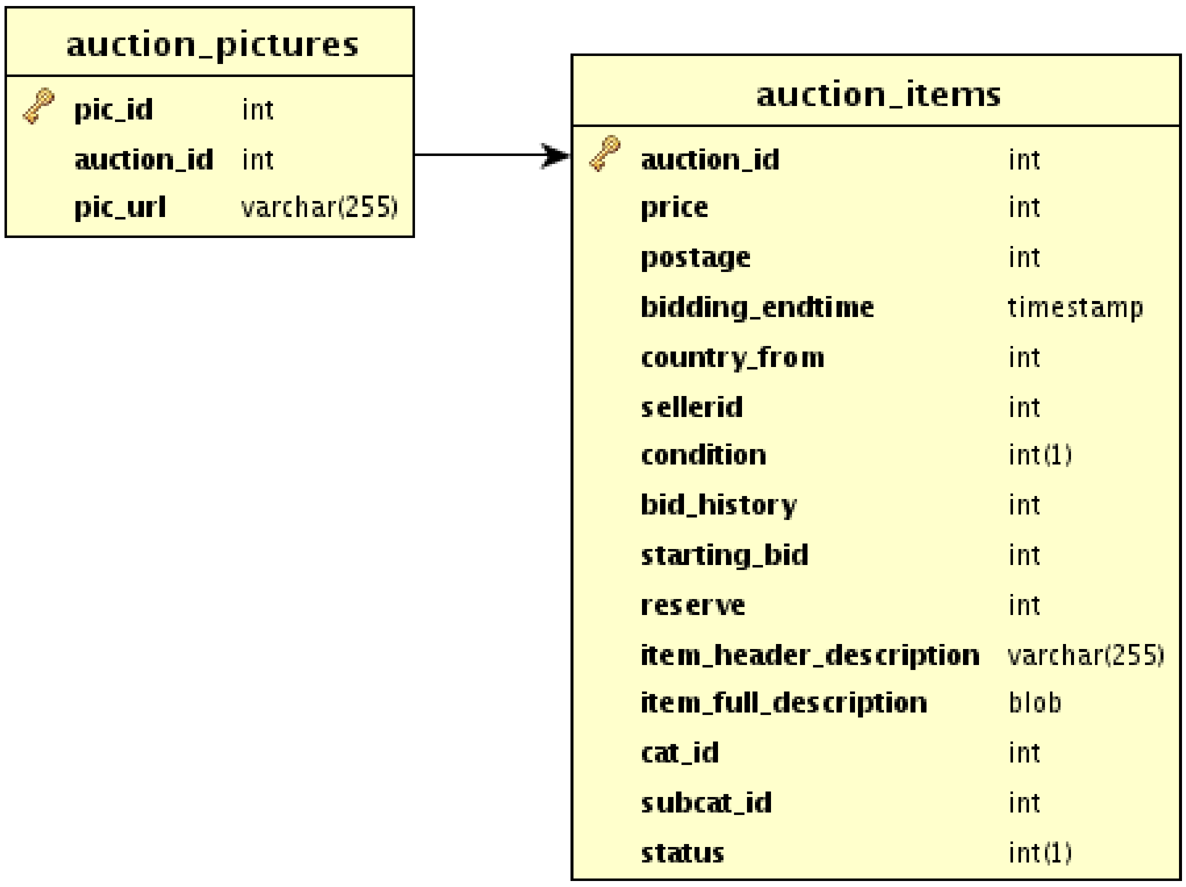
Auction items



SQL

```
CREATE TABLE auction_items (
  auction_id int(11) NOT NULL AUTO_INCREMENT,
  price int(11) NOT NULL,
  postage int(11) NOT NULL,
  bidding_endtime timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  country_from int(11) NOT NULL,
  sellerid int(11) NOT NULL,
  `condition` int(1) NOT NULL,
  bid_history int(11) NOT NULL,
  starting_bid int(11) NOT NULL,
  reserve int(11) NOT NULL,
  item_header_description varchar(255) NOT NULL,
  item_full_description blob NOT NULL,
  cat_id int(11) NOT NULL,
  subcat_id int(11) NOT NULL,
  `status` int(1) NOT NULL,
  PRIMARY KEY (auction_id),
  KEY country_from (country_from),
  KEY sellerid (sellerid),
  KEY cat_id (cat_id),
  KEY subcat_id (subcat_id)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1;
```

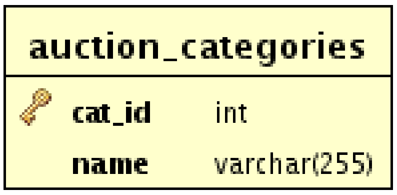
Auction pictures



SQL

```
CREATE TABLE IF NOT EXISTS `auction_pictures` (
  `pic_id` int(11) NOT NULL AUTO_INCREMENT,
  `auction_id` int(11) NOT NULL,
  `pic_url` varchar(255) NOT NULL,
  PRIMARY KEY (`pic_id`),
  KEY `auction_id` (`auction_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=9 ;
```

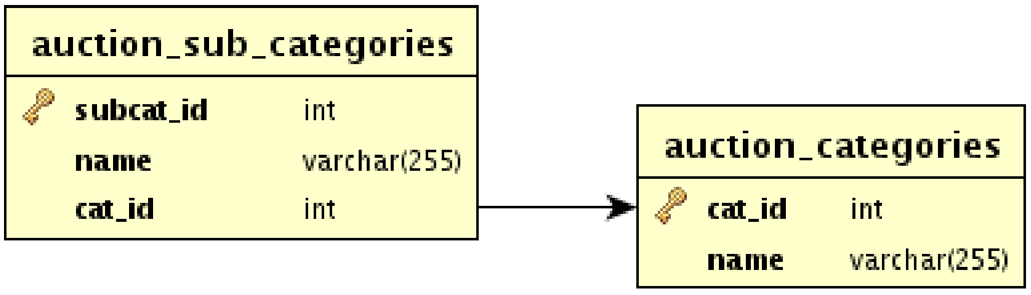
Auction categories



SQL

```
CREATE TABLE auction_categories (
  cat_id int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  PRIMARY KEY (cat_id)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1;
```

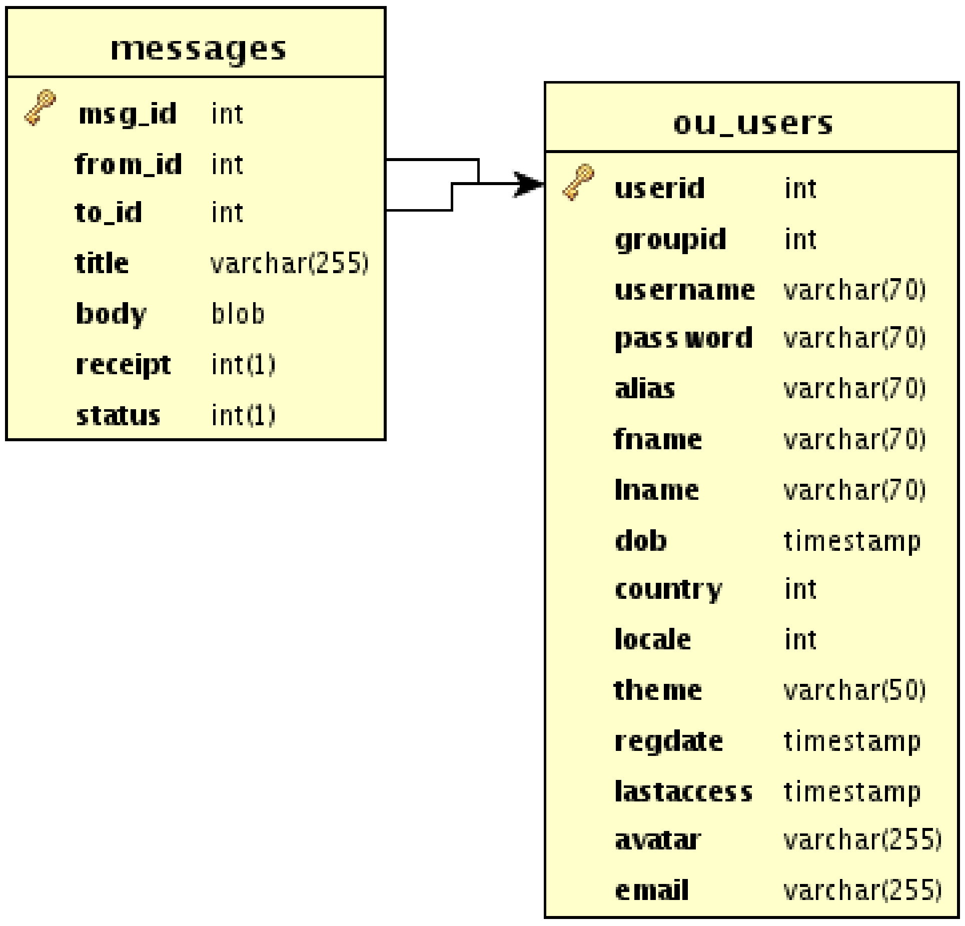
Auction sub categories



SQL

```
CREATE TABLE auction_sub_categories (
  subcat_id int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  cat_id int(11) NOT NULL,
  PRIMARY KEY (subcat_id),
  KEY cat_id (cat_id)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1;
```

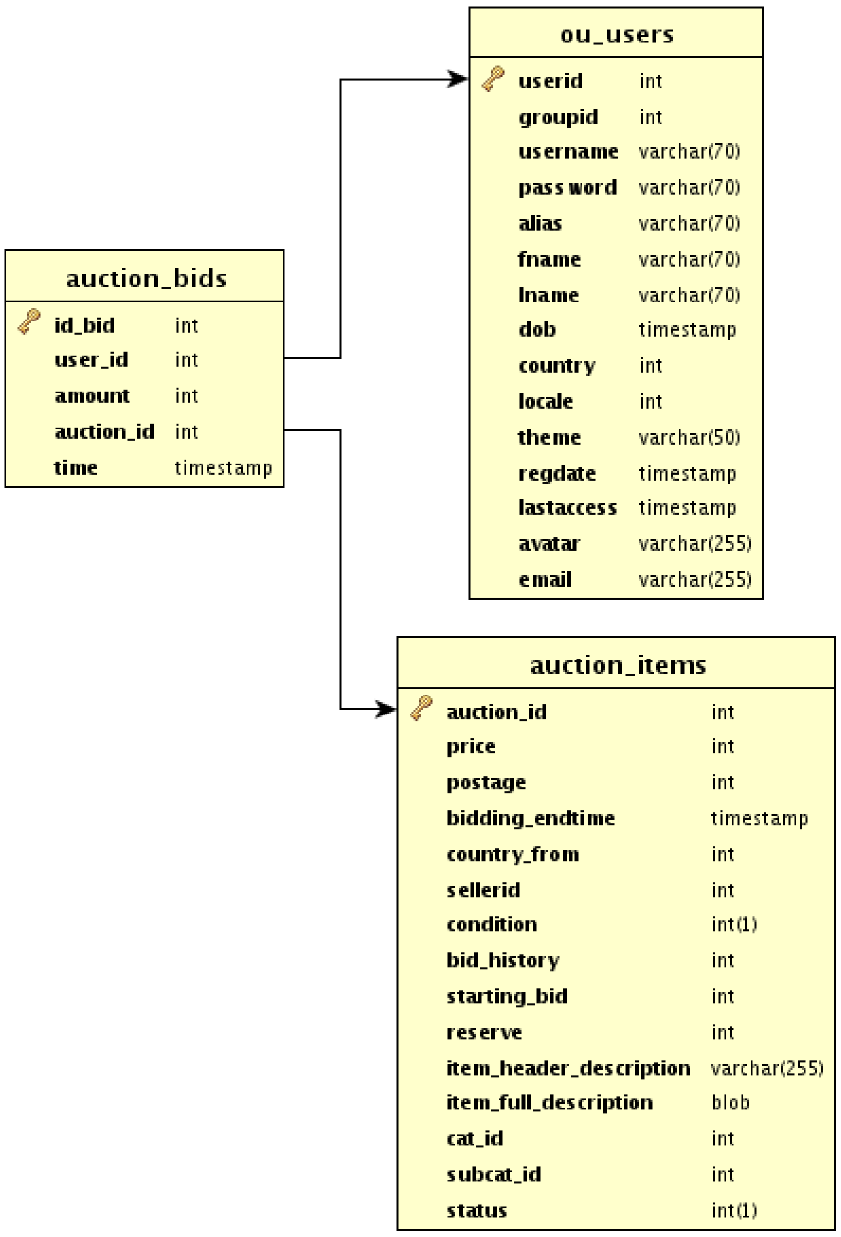
Messages



SQL

```
CREATE TABLE messages (
  msg_id int(11) NOT NULL AUTO_INCREMENT,
  from_id int(11) NOT NULL,
  to_id int(11) NOT NULL,
  title varchar(255) NOT NULL,
  body blob NOT NULL,
  receipt int(1) NOT NULL,
  `status` int(1) NOT NULL,
  PRIMARY KEY (msg_id),
  KEY from_id (from_id),
  KEY to_id (to_id)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1;
```

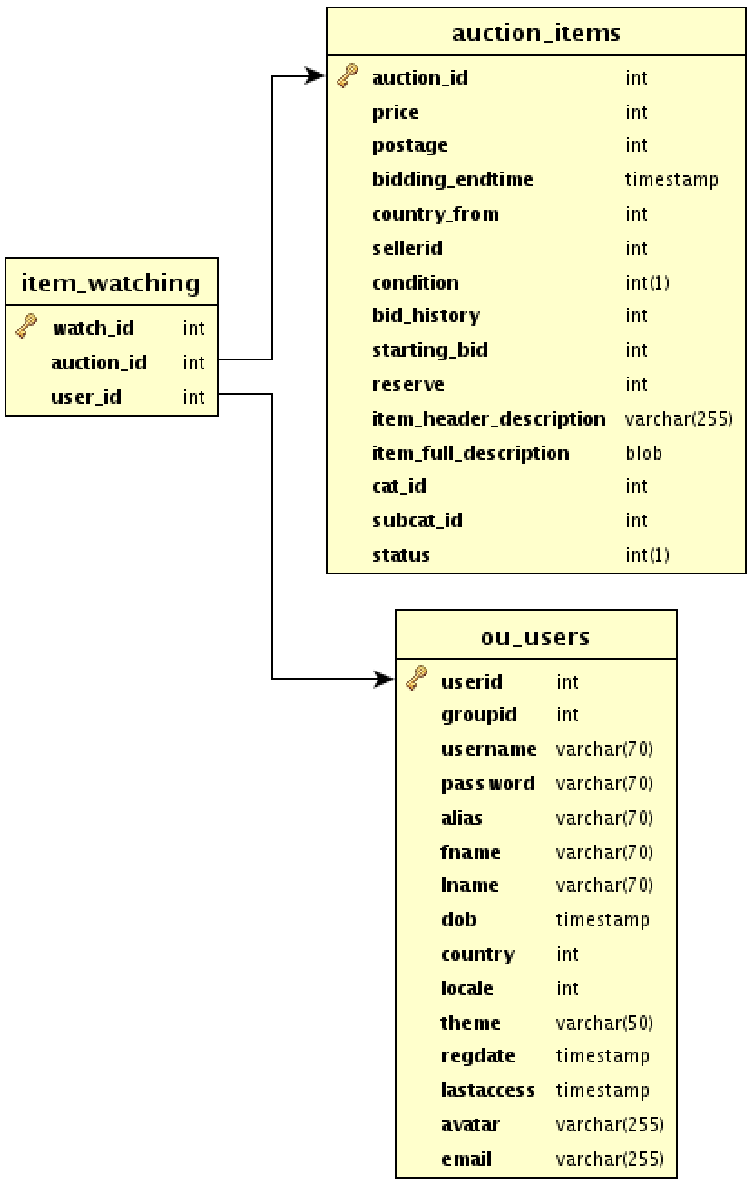
Auction Bids



SQL

```
CREATE TABLE auction_bids (
  id_bid int(11) NOT NULL AUTO_INCREMENT,
  user_id int(11) NOT NULL,
  amount int(11) NOT NULL,
  auction_id int(11) NOT NULL,
  `time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (id_bid),
  KEY auction_id (auction_id),
  KEY user_id (user_id)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1;
```

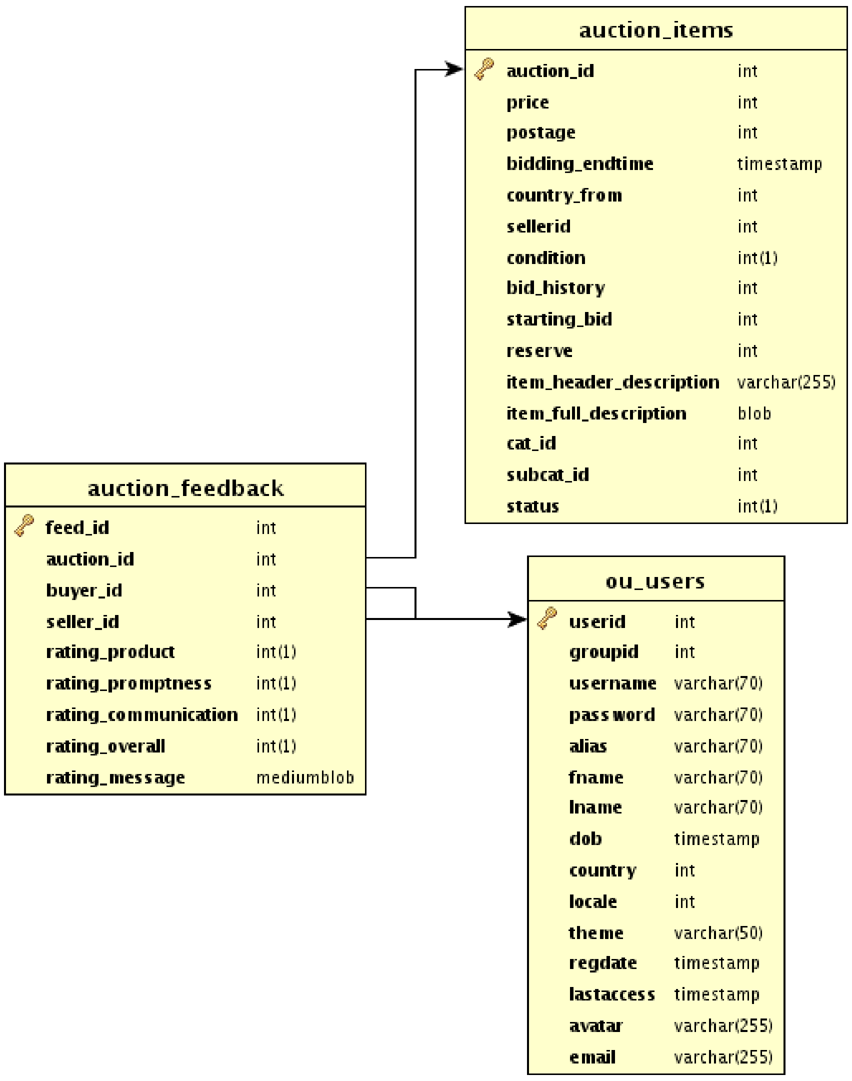
Item watching



SQL

```
CREATE TABLE item_watching (
  watch_id int(11) NOT NULL AUTO_INCREMENT,
  auction_id int(11) NOT NULL,
  user_id int(11) NOT NULL,
  PRIMARY KEY (watch_id),
  KEY auction_id (auction_id),
  KEY user_id (user_id)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1;
```

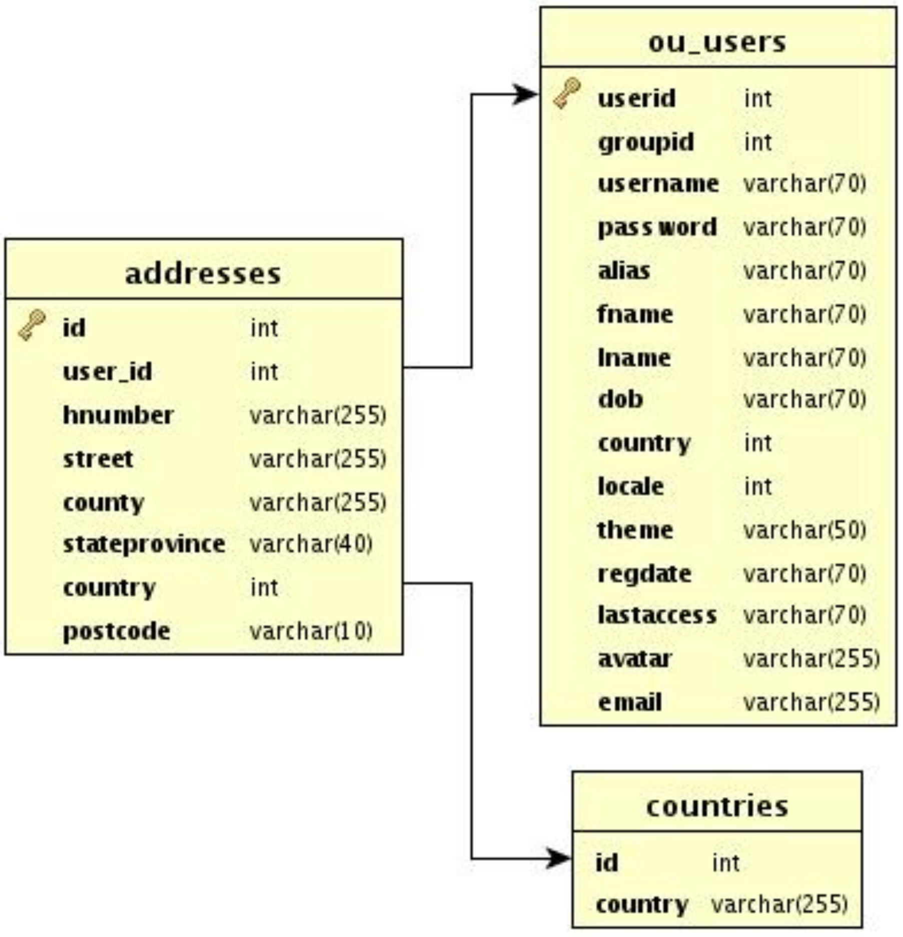
Auction feedback



SQL

```
CREATE TABLE auction_feedback (
  feed_id int(11) NOT NULL AUTO_INCREMENT,
  auction_id int(11) NOT NULL,
  buyer_id int(11) NOT NULL,
  seller_id int(11) NOT NULL,
  rating_product int(1) NOT NULL,
  rating_promptness int(1) NOT NULL,
  rating_communication int(1) NOT NULL,
  rating_overall int(1) NOT NULL,
  rating_message mediumblob NOT NULL,
  PRIMARY KEY (feed_id),
  KEY auction_id (auction_id),
  KEY buyer_id (buyer_id),
  KEY seller_id (seller_id)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1;
```

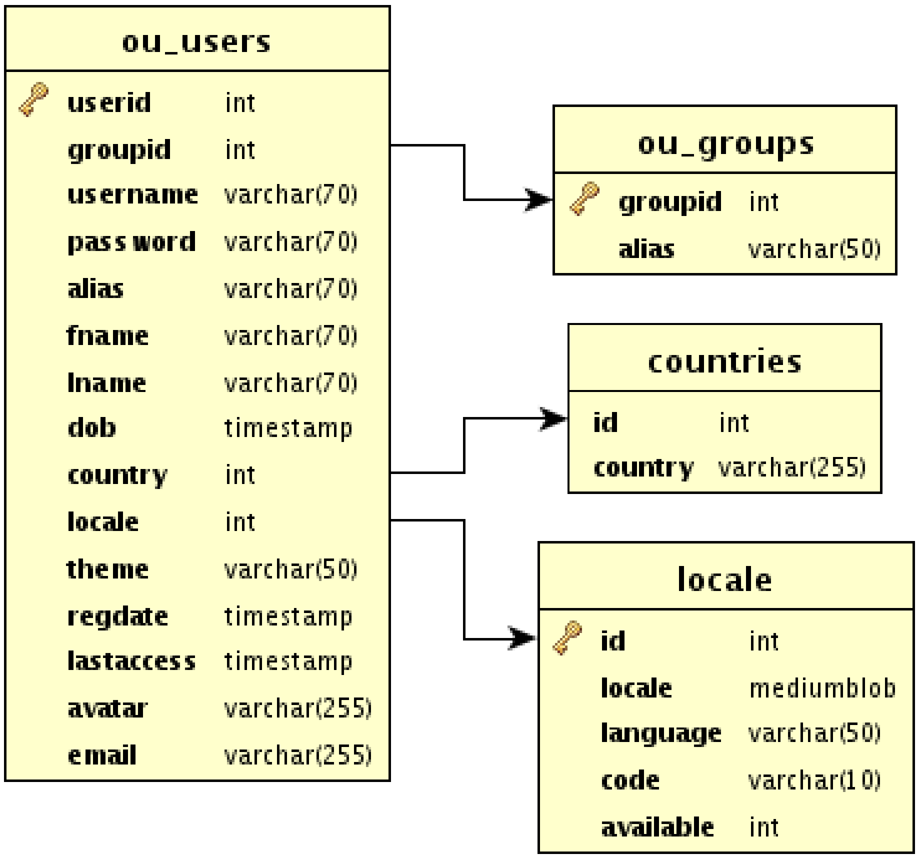
Addresses



SQL

```
CREATE TABLE addresses (
  id int(11) NOT NULL AUTO_INCREMENT,
  user_id int(11) NOT NULL,
  hnumber varchar(255) NOT NULL,
  street varchar(255) NOT NULL,
  county varchar(255) NOT NULL,
  stateprovince varchar(40) NOT NULL,
  country int(11) NOT NULL,
  postcode varchar(10) NOT NULL,
  PRIMARY KEY (id),
  KEY user_id (user_id),
  KEY country (country)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1;
```

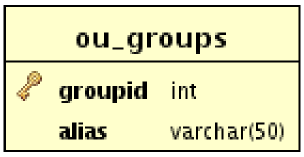
OU users



SQL

```
CREATE TABLE `ou_users` (
  `userid` int(11) NOT NULL AUTO_INCREMENT,
  `groupid` int(11) NOT NULL,
  `username` varchar(70) NOT NULL,
  `password` varchar(70) NOT NULL,
  `alias` varchar(70) NOT NULL,
  `fname` varchar(70) NOT NULL,
  `lname` varchar(70) NOT NULL,
  `dob` varchar(70) NOT NULL,
  `country` int(11) NOT NULL,
  `locale` int(11) NOT NULL,
  `theme` varchar(50) NOT NULL,
  `regdate` varchar(70) NOT NULL,
  `lastaccess` varchar(70) NOT NULL,
  `avatar` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  PRIMARY KEY (`userid`),
  KEY `groupid` (`groupid`),
  KEY `country` (`country`),
  KEY `locale` (`locale`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;
```

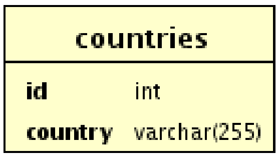
OU groups



SQL

```
CREATE TABLE `ou_groups` (
  `groupid` int(11) NOT NULL AUTO_INCREMENT,
  `alias` varchar(50) NOT NULL,
  PRIMARY KEY (`groupid`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=7 ;
```

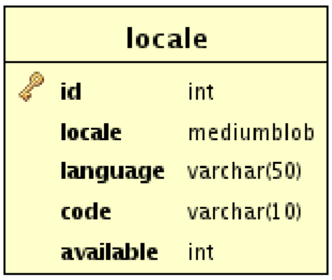
Countries



SQL

```
CREATE TABLE `countries` (
  `id` int(11) NOT NULL,
  `country` varchar(255) NOT NULL,
  KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Locale



locale	
id	int
locale	mediumblob
language	varchar(50)
code	varchar(10)
available	int

SQL

```
CREATE TABLE `locale` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `locale` varchar(50) NOT NULL,
  `language` varchar(50) NOT NULL,
  `code` varchar(10) NOT NULL,
  `available` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=93 ;
```

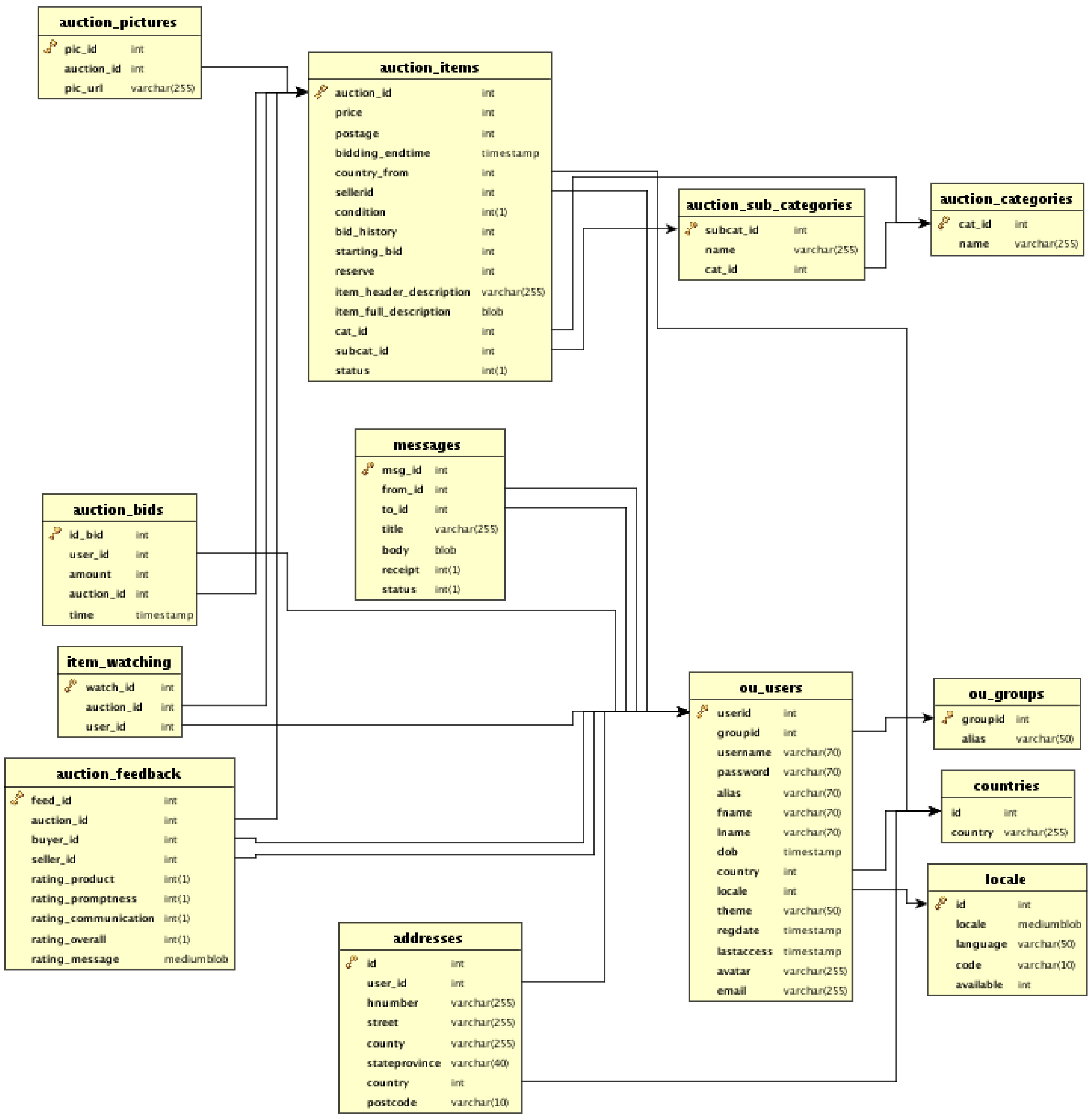
Referential Integrity Constraints

SQL

```
ALTER TABLE `addresses`
  ADD CONSTRAINT `addresses_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `ou_users` (`userid`),
  ADD CONSTRAINT `addresses_ibfk_2` FOREIGN KEY (`country`) REFERENCES `countries` (`id`);

ALTER TABLE `auction_bids`
  ADD CONSTRAINT `auction_bids_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `ou_users` (`userid`),
  ADD CONSTRAINT `auction_bids_ibfk_2` FOREIGN KEY (`auction_id`) REFERENCES `auction_items`
(`auction_id`);

ALTER TABLE `auction_feedback`
  ADD CONSTRAINT `auction_feedback_ibfk_1` FOREIGN KEY (`auction_id`) REFERENCES `auction_items`
(`auction_id`),
  ADD CONSTRAINT `auction_feedback_ibfk_2` FOREIGN KEY (`buyer_id`) REFERENCES `ou_users` (`userid`),
  ADD CONSTRAINT `auction_feedback_ibfk_3` FOREIGN KEY (`seller_id`) REFERENCES `ou_users` (`userid`);

ALTER TABLE `auction_items`
  ADD CONSTRAINT `auction_items_ibfk_1` FOREIGN KEY (`country_from`) REFERENCES `countries` (`id`),
  ADD CONSTRAINT `auction_items_ibfk_2` FOREIGN KEY (`sellerid`) REFERENCES `ou_users` (`userid`),
  ADD CONSTRAINT `auction_items_ibfk_3` FOREIGN KEY (`cat_id`) REFERENCES `auction_categories`
(`cat_id`),
  ADD CONSTRAINT `auction_items_ibfk_4` FOREIGN KEY (`subcat_id`) REFERENCES
`auction_sub_categories` (`subcat_id`);

ALTER TABLE `auction_pictures`
  ADD CONSTRAINT `auction_pictures_ibfk_1` FOREIGN KEY (`auction_id`) REFERENCES `auction_items`
(`auction_id`);

ALTER TABLE `auction_sub_categories`
  ADD CONSTRAINT `auction_sub_categories_ibfk_1` FOREIGN KEY (`cat_id`) REFERENCES
`auction_categories` (`cat_id`);

ALTER TABLE `item_watching`
  ADD CONSTRAINT `item_watching_ibfk_1` FOREIGN KEY (`auction_id`) REFERENCES `auction_items`
(`auction_id`),
  ADD CONSTRAINT `item_watching_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES `ou_users` (`userid`);

ALTER TABLE `messages`
  ADD CONSTRAINT `messages_ibfk_1` FOREIGN KEY (`from_id`) REFERENCES `ou_users` (`userid`),
  ADD CONSTRAINT `messages_ibfk_2` FOREIGN KEY (`to_id`) REFERENCES `ou_users` (`userid`);

ALTER TABLE `ou_users`
  ADD CONSTRAINT `ou_users_ibfk_1` FOREIGN KEY (`groupid`) REFERENCES `ou_groups` (`groupid`),
  ADD CONSTRAINT `ou_users_ibfk_2` FOREIGN KEY (`country`) REFERENCES `countries` (`id`),
  ADD CONSTRAINT `ou_users_ibfk_3` FOREIGN KEY (`locale`) REFERENCES `locale` (`id`);
```

Overview

Relations Overview



Design & Implementation

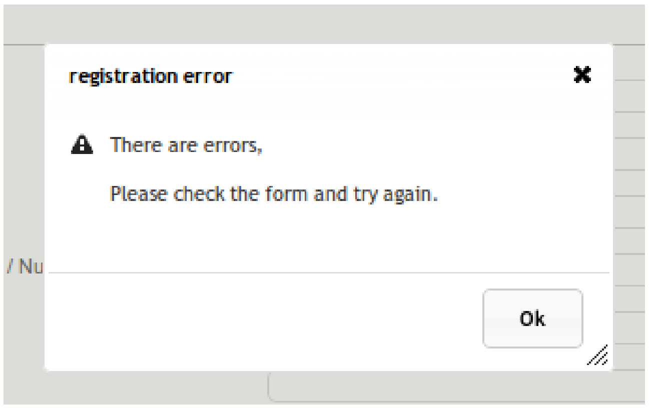
Requirements

Major Processes

User Registration

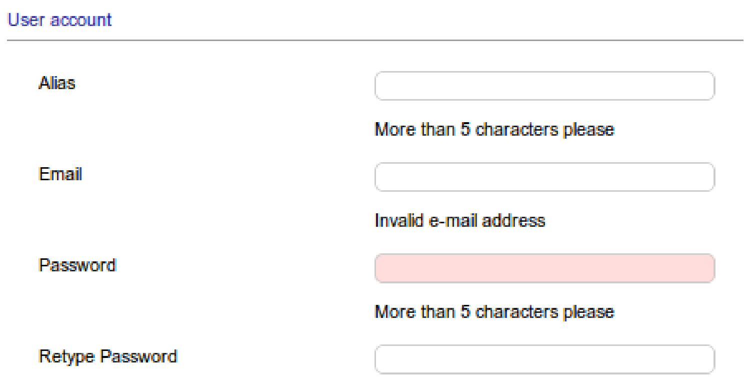
The PHP logic implements a series of validation checks to prevent the user from entering unwanted data. Email, username, minimum password length, addresses, DOB etc. are further checked with regular expressions to ensure coherent data. Any errors will be shown to the user by indicating which fields had incorrect data. Also the PHP logic has built in methods for taking AJAX requests, to check availability of username and email already is use prohibitions. Activation, although can be completed by email; is disabled as the server not have ‘sendmail ‘enabled.

ALL html, CSS and JavaScript completed. JQuery and plugins are used for AJAX requests to the server on form elements to verify email, username are not already in use. Using JQuery, problems fields are identified immediately indicating to the user which fields have problem data. Dates are handled by JQuery plugin DATEPICKER to ensure valid dates. JQueryUI Modal windows give further indication to the user when problems occur to increase the visibility of system status and error prevention. (Nielsen's heuristics) Minor improvements to be made to CSS and avatar upload.


registration error

There are errors,

Please check the form and try again.

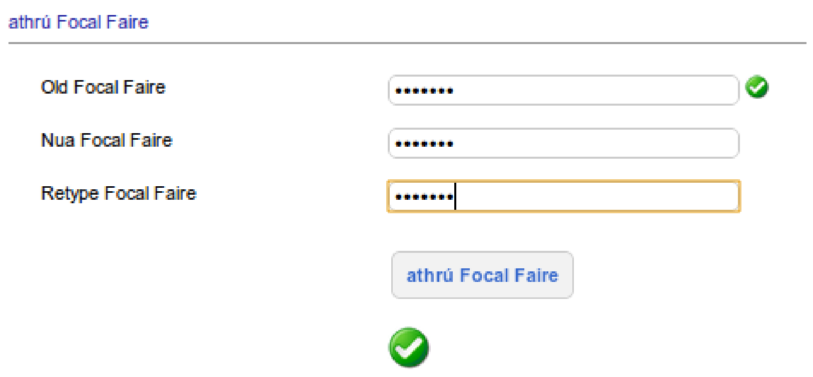

User account

Alias	More than 5 characters please
Email	Invalid e-mail address
Password	More than 5 characters please
Retype Password	

Update registration

The PHP logic implements a series of validation checks to prevent the user from entering unwanted data. Any errors will be shown to the user by indicating which fields had incorrect data. Ajax and supported PHP logic upload allows the user to change the avatar, address, password and other details without refreshing the page.

ALL html, CSS and JavaScript completed. JQuery and plugins are used to show errors and prevent errors in user entered data. All updates are done using AJAX. Minor improvements to be made, (change email address) but cannot be implemented till sendmail is enabled.


athrú Focal Faire

Old Focal Faire	••••••• ✓
Nua Focal Faire	•••••••
Retype Focal Faire	•••••••

Login

The PHP logic implements a series of validation checks to prevent the user from entering unwanted data. Any errors will be shown to the user by indicating which fields had incorrect data. On login complete the users profile will decide which locale will be used on the following pages.   If a user is banned they will be notified and prevented from logging in.

ALL html, CSS and JavaScript completed. JQuery and plugins are used to show error and prevent error in user entered data.

Logout

The user’s session is invalidated and the script returns the user to the main page.

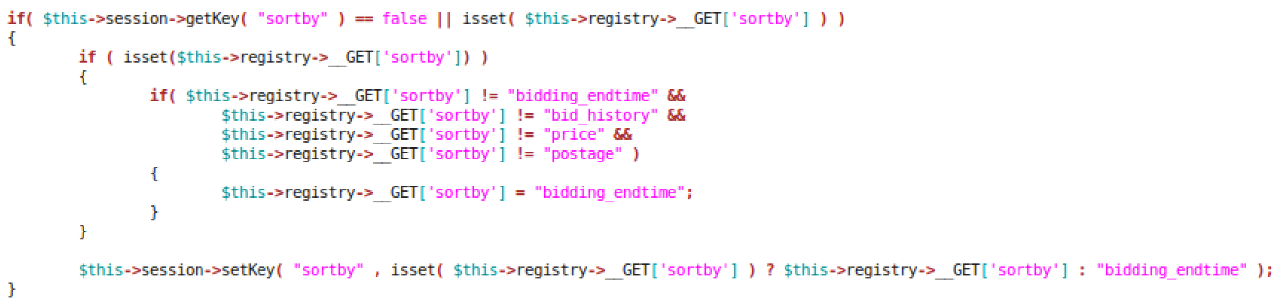
Submit Auction

The PHP logic implements a series of validation checks to prevent the user from entering unwanted data. Any errors will be shown to the user by indicating which fields had incorrect data. Prices, times etc are checked using regular expressions to check for coherent data. If there are any errors the page will be represented with notice errors next to the offending fields. PHP implemented to handle temporary storage of upload images until the process is complete.

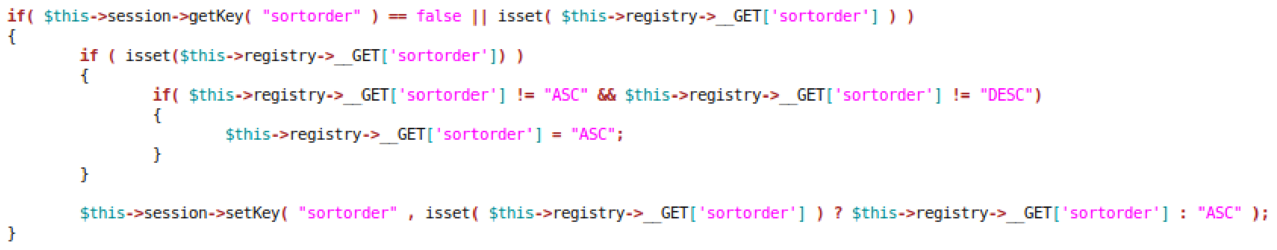
ALL html, CSS and JavaScript completed.   JQuery and plugins are used to show error and prevent error in user entered data. Ajax is used to upload images and is immediately presented on the auction item page. The user has the ability to remove a picture immediately if they wish to do so. Times and prices fields are validate by JQuery Date picker, time picker, price picker to prevent invalid data before submitting to the server.

Browse Auctions

The sections are divided into categories and sub categories. PHP implements pagination and items per page. Auctions are shown only if the are newer than the current date. Listing can be viewed by category or by category and sub category. Sorting is also implemented, currently is allows for sorting by date, price, bids, postage and also by name. A series of validation checks prevents the user from sending mal formed GET queries which control the flow of these features. If malformed data is sent the script which go back to standard view.



```
if( $this->session->getKey( "sortby" ) == false || isset( $this->registry->__GET['sortby'] ) )
{
	if ( isset($this->registry->__GET['sortby']) )
	{
		if( $this->registry->__GET['sortby'] != "bidding_endtime" &&
			$this->registry->__GET['sortby'] != "bid_history" &&
			$this->registry->__GET['sortby'] != "price" &&
			$this->registry->__GET['sortby'] != "postage" )
		{
			$this->registry->__GET['sortby'] = "bidding_endtime";
		}
	}

	$this->session->setKey( "sortby" , isset( $this->registry->__GET['sortby'] ) ? $this->registry->__GET['sortby'] : "bidding_endtime" );
}
```



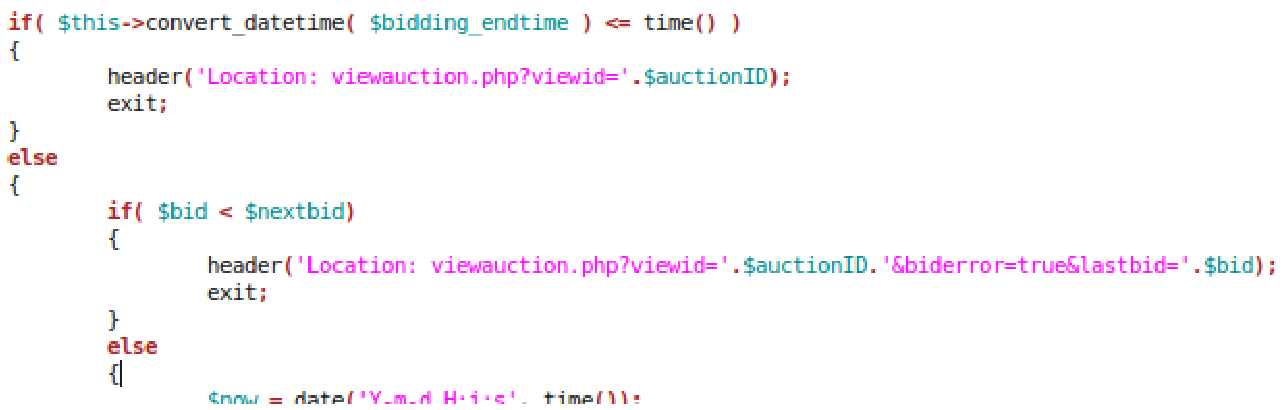
```
if( $this->session->getKey( "sortorder" ) == false || isset( $this->registry->__GET['sortorder'] ) )
{
	if ( isset($this->registry->__GET['sortorder']) )
	{
		if( $this->registry->__GET['sortorder'] != "ASC" && $this->registry->__GET['sortorder'] != "DESC")
		{
			$this->registry->__GET['sortorder'] = "ASC";
		}
	}

	$this->session->setKey( "sortorder" , isset( $this->registry->__GET['sortorder'] ) ? $this->registry->__GET['sortorder'] : "ASC" );
}
```

View auction

The view auction page presents all auctions present and past. The output differs if the auction is complete and presents the winner on the page. The place bid option is removed. An auction is selected by providing the auctioned in a GET request. Validation checks take place to prevent malformed data from being sent in the GET request.

Bidding

The bidding process implements a series of checks to prevent a user from bidding under the current bid. Firstly the user must be logged in, on the auction page the next bidding multiple will be presented, the use can bid higher be must be at least the bid presented in order to successfully bid. Once the bid is added the user can view his current bids on the ‘my bids’ section of the web page. He can see his / hers current bid and the current high bid.



```
if( $this->convert_datetime( $bidding_endtime ) <= time() )
{
        header('Location: viewauction.php?viewid='.$auctionID);
        exit;
}
else
{
        if( $bid < $nextbid)
        {
                header('Location: viewauction.php?viewid='.$auctionID.'&biderror=true&lastbid='.$bid);
                exit;
        }
        else
        {
                $now = date('Y-m-d H:i:s', time());
```

Searching

The simple search is an auto suggests form on the main page. It allows the user to search by a name. The advanced search however allows the user to submit a series of predicates.   The request and its predicates are sent to the server via AJAX. The PHP implements a query builder to query the SQL database and find the results the user is looking for.   Finally the information is sent back to the browser and JavaScript is used to create a table which is then sort able using JavaScript.

Feedback

After an auction is completed, the winning party has the opportunity to leave feedback on the item won. They access this through 'my bids' where a feedback button is available on the auctions they have won and where feedback has not already been left. The feedback page allows the user to leave feedback and levels of acceptance of promptness, communication, etc. Once feedback is left, the seller is able to view the feedback on the feedback page.

Watch item

On the auction page a watch item button is available to allow users to add the item to their watching list. On the watching page the user can quickly view items there are watching and have the ability to remove the item form their watch list.

Administrator

Delete User

The administrator has the ability to remove a user. When removing the users the auctions for that user, pictures, feedback, messages, address is removed also because of the foreign key constraints existing in the database.

Ban User

Banning users changes there group to group6. When the banned user attempts to login they are presented with error. Immediately there session is invalidated so as they cannot do anything with the account.

Update auction

The administrator has the ability to change the Time and date of the auction end time and the ability to completely remove the auction if they wish to do so.

Delete auction

The administrator first selects the user which is auctioning the item. Then a list of his or hers items is shown and the administrator selects the auction to remove. On removing the pictures and feedback are also removed.

Non Functional Requirements

Model – View - Controller

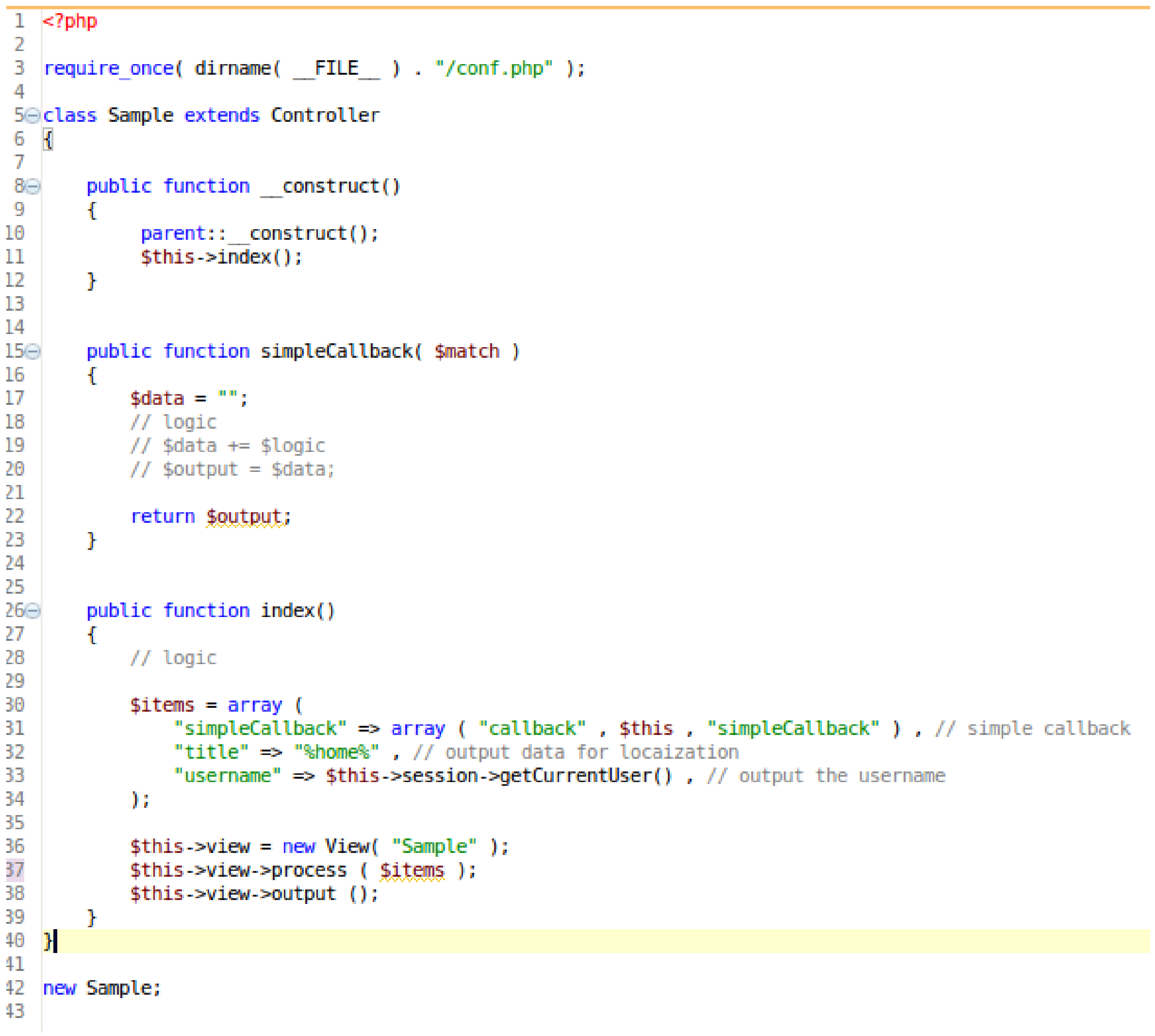
Sample controller

The business logic is done in the controller instances before handing off to the view for output. Once to business logic is completed an array of Keys (replacements) and there values are constructed to tell the view which items are to be replaced within the output template html.

Three distinct types of values are present in the example below.

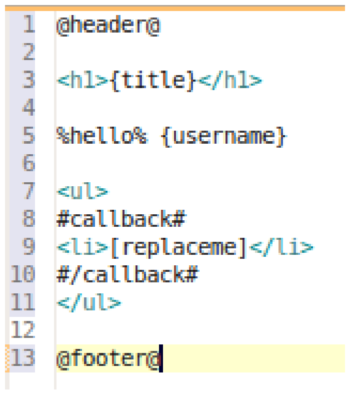
- A call back to the controller instance for a matched block that requires more processing.
- A replacement that requires a language translation
- A standard replacement.

This array is processed by the view, providing the name of the template to use in the View constructor.



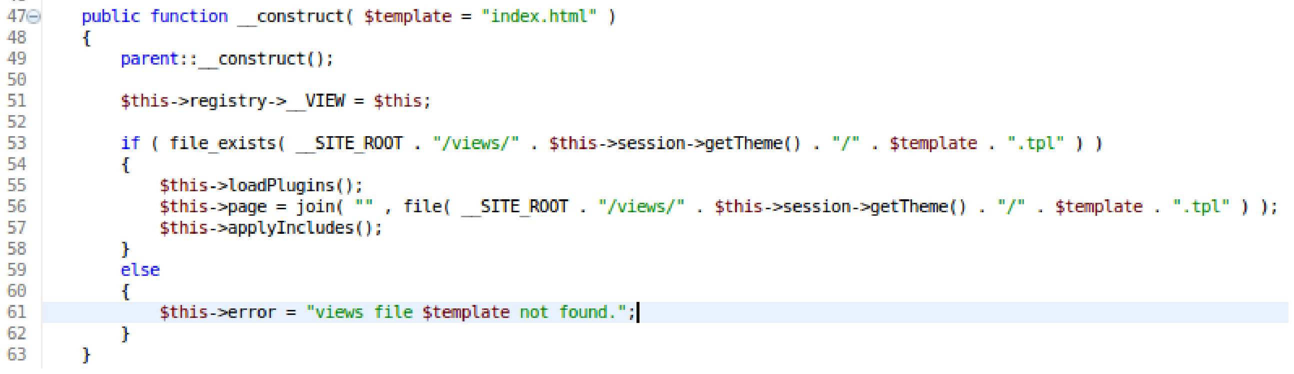
```php
<?php

require_once( dirname( __FILE__ ) . "/conf.php" );

class Sample extends Controller
{

    public function __construct()
    {
         parent::__construct();
         $this->index();
    }


    public function simpleCallback( $match )
    {
        $data = "";
        // logic
        // $data += $logic
        // $output = $data;

        return $output;
    }


    public function index()
    {
        // logic

        $items = array (
            "simpleCallback" => array ( "callback" , $this , "simpleCallback" ) , // simple callback
            "title" => "%home%" , // output data for locaization
            "username" => $this->session->getCurrentUser() , // output the username
        );

        $this->view = new View( "Sample" );
        $this->view->process ( $items );
        $this->view->output ();
    }
}

new Sample;

```

Sample Template



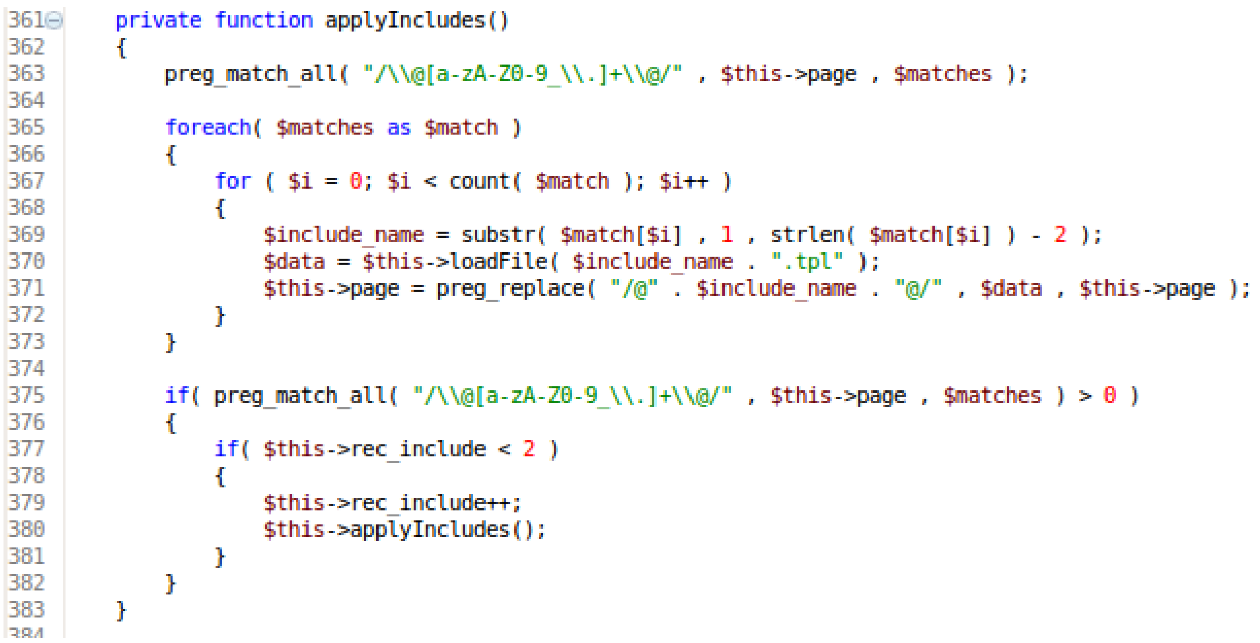
```
@header@

<h1>{title}</h1>

%hello% {username}

<ul>
#callback#
<li>[replaceme]</li>
#/callback#
</ul>

@footer@
```

View Constructor



```
    public function __construct( $template = "index.html" )
    {
        parent::__construct();

        $this->registry->__VIEW = $this;

        if ( file_exists( __SITE_ROOT . "/views/" . $this->session->getTheme() . "/" . $template . ".tpl" ) )
        {
            $this->loadPlugins();
            $this->page = join( "" , file( __SITE_ROOT . "/views/" . $this->session->getTheme() . "/" . $template . ".tpl" ) );
            $this->applyIncludes();
        }
        else
        {
            $this->error = "views file $template not found.";
        }
    }
```

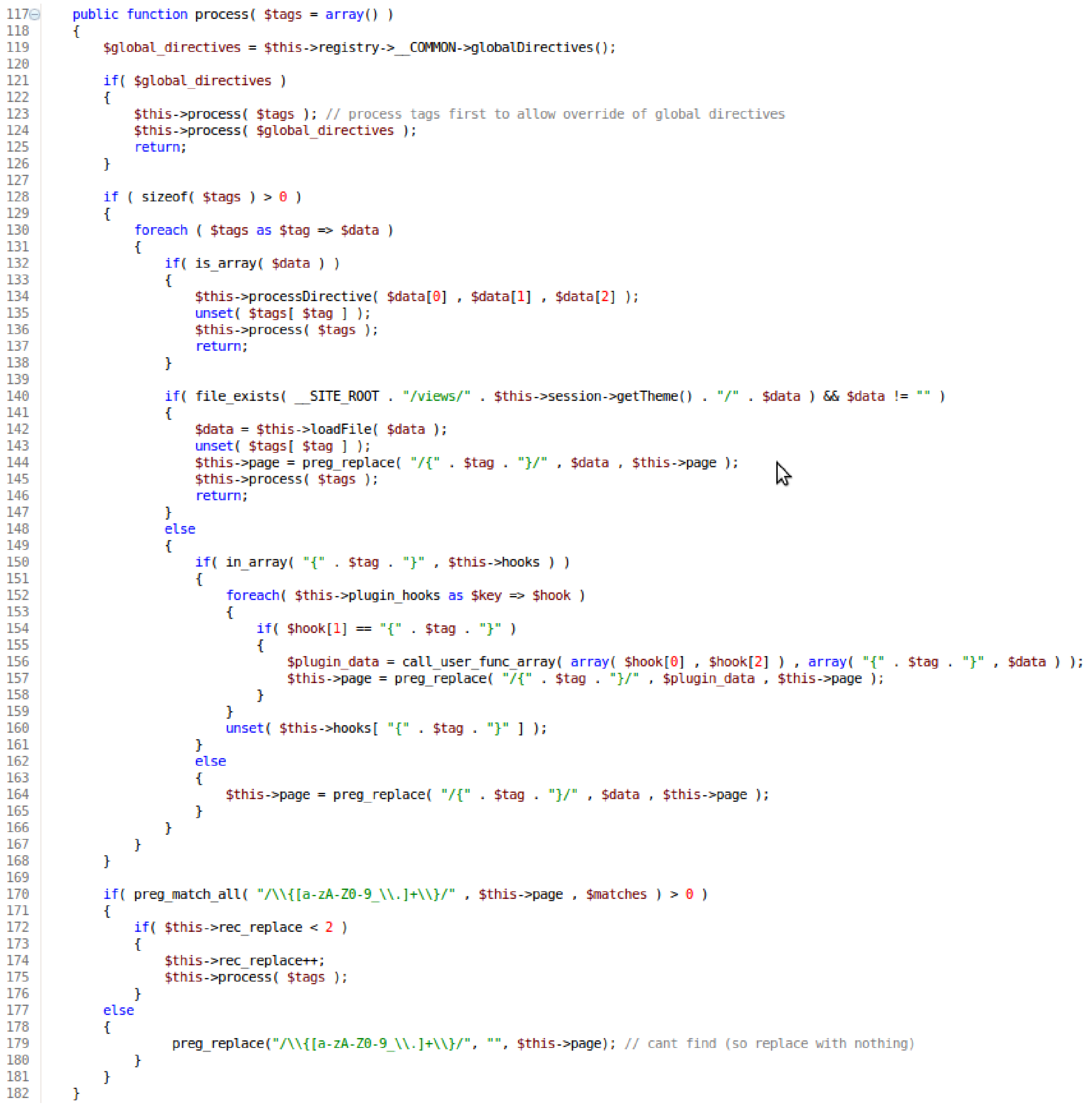
View Includes



```
    private function applyIncludes()
    {
        preg_match_all( "/\\@[a-zA-Z0-9_\\.]+\\@/" , $this->page , $matches );

        foreach( $matches as $match )
        {
            for ( $i = 0; $i < count( $match ); $i++ )
            {
                $include_name = substr( $match[$i] , 1 , strlen( $match[$i] ) - 2 );
                $data = $this->loadFile( $include_name . ".tpl" );
                $this->page = preg_replace( "/@" . $include_name . "@/" , $data , $this->page );
            }
        }

        if( preg_match_all( "/\\@[a-zA-Z0-9_\\.]+\\@/" , $this->page , $matches ) > 0 )
        {
            if( $this->rec_include < 2 )
            {
                $this->rec_include++;
                $this->applyIncludes();
            }
        }
    }
```

Includes are handled within the templates by matching the @INCLUDE_THIS@ syntax. The file contents of file are then included in the final output. Levels of recursion are allowed, if the file included has further includes.

View Processing

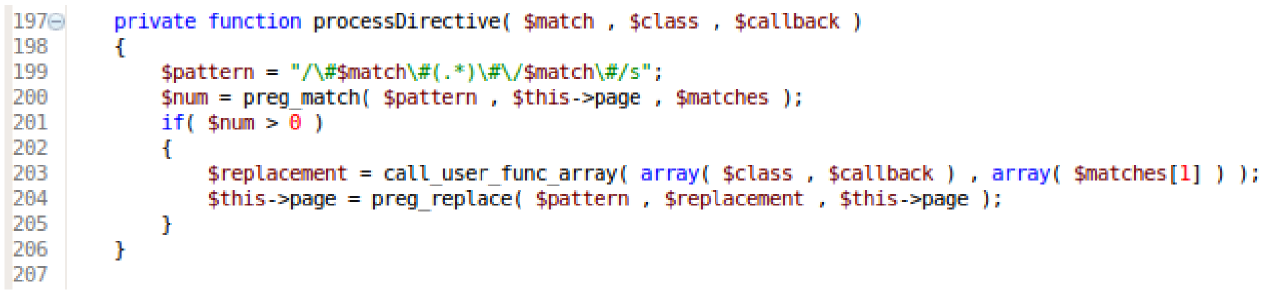


```
117  public function process( $tags = array() )
118  {
119      $global_directives = $this->registry->__COMMON->globalDirectives();
120
121      if( $global_directives )
122      {
123          $this->process( $tags ); // process tags first to allow override of global directives
124          $this->process( $global_directives );
125          return;
126      }
127
128      if ( sizeof( $tags ) > 0 )
129      {
130          foreach ( $tags as $tag => $data )
131          {
132              if( is_array( $data ) )
133              {
134                  $this->processDirective( $data[0] , $data[1] , $data[2] );
135                  unset( $tags[ $tag ] );
136                  $this->process( $tags );
137                  return;
138              }
139
140              if( file_exists( __SITE_ROOT . "/views/" . $this->session->getTheme() . "/" . $data ) && $data != "" )
141              {
142                  $data = $this->loadFile( $data );
143                  unset( $tags[ $tag ] );
144                  $this->page = preg_replace( "/{" . $tag . "}/" , $data , $this->page );
145                  $this->process( $tags );
146                  return;
147              }
148              else
149              {
150                  if( in_array( "{" . $tag . "}" , $this->hooks ) )
151                  {
152                      foreach( $this->plugin_hooks as $key => $hook )
153                      {
154                          if( $hook[1] == "{" . $tag . "}" )
155                          {
156                              $plugin_data = call_user_func_array( array( $hook[0] , $hook[2] ) , array( "{" . $tag . "}" , $data ) );
157                              $this->page = preg_replace( "/{" . $tag . "}/" , $plugin_data , $this->page );
158                          }
159                      }
160                      unset( $this->hooks[ "{" . $tag . "}" ] );
161                  }
162                  else
163                  {
164                      $this->page = preg_replace( "/{" . $tag . "}/" , $data , $this->page );
165                  }
166              }
167          }
168      }
169
170      if( preg_match_all( "/\\{[a-zA-Z0-9_\\.]+\\}/" , $this->page , $matches ) > 0 )
171      {
172          if( $this->rec_replace < 2 )
173          {
174              $this->rec_replace++;
175              $this->process( $tags );
176          }
177      else
178      {
179           preg_replace("/\\{[a-zA-Z0-9_\\.]+\\}/", "", $this->page); // cant find (so replace with nothing)
180          }
181      }
182  }
```

The first element in the replacement tags is an array which is identified as the processing directive. Line 132 identifies that a call back is required. The process directive method is called in the view and the values are handed off to this method. When the method is done processing the output is return and inputted in the html.

The second element is a replacement; line 164 handles this replacement by matching {title} in the view and replacing it with %home%. The third element is handled the same way.

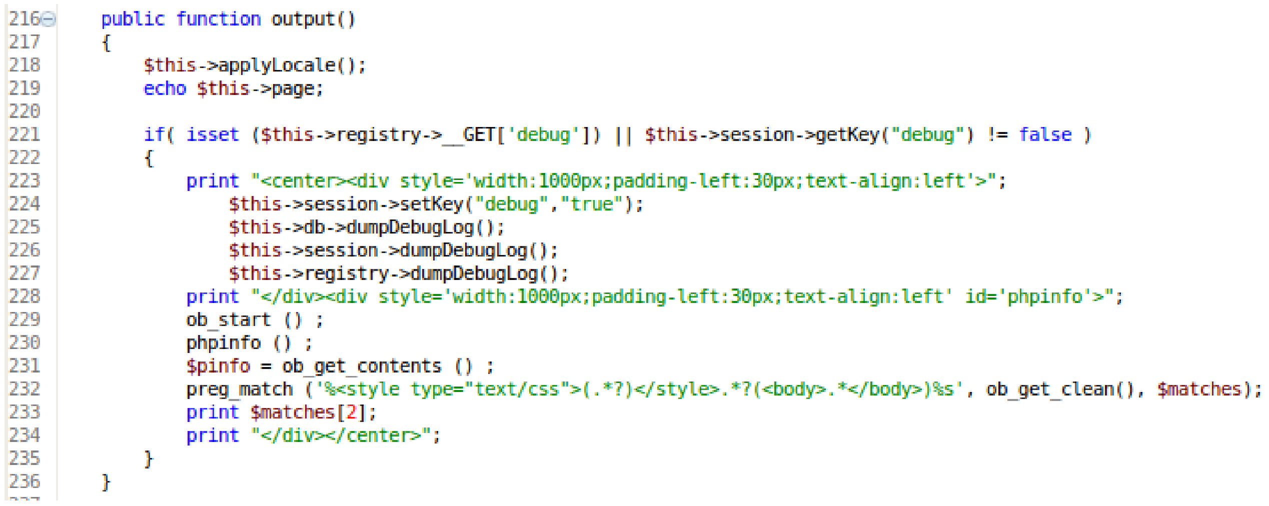
View Process Directive



```php
    private function processDirective( $match , $class , $callback )
    {
        $pattern = "/\#$match\#(.*)\#\/$match\#/s";
        $num = preg_match( $pattern , $this->page , $matches );
        if( $num > 0 )
        {
            $replacement = call_user_func_array( array( $class , $callback ) , array( $matches[1] ) );
            $this->page = preg_replace( $pattern , $replacement , $this->page );
        }
    }

```

Using the call user function array method in PHP we call the controller instance method specified and pass the html matched (Sample view lines 8 -10) to this method for further processing.

View Output

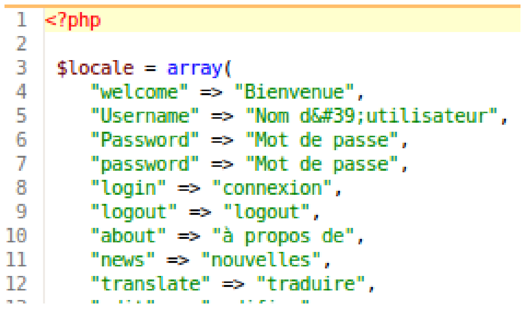


```php
    public function output()
    {
        $this->applyLocale();
        echo $this->page;

        if( isset ($this->registry->__GET['debug']) || $this->session->getKey("debug") != false )
        {
            print "<center><div style='width:1000px;padding-left:30px;text-align:left'>";
                $this->session->setKey("debug","true");
                $this->db->dumpDebugLog();
                $this->session->dumpDebugLog();
                $this->registry->dumpDebugLog();
            print "</div><div style='width:1000px;padding-left:30px;text-align:left' id='phpinfo'>";
            ob_start () ;
            phpinfo () ;
            $pinfo = ob_get_contents () ;
            preg_match ('%<style type="text/css">(.*?)</style>.*?(<body>.*</body>)%s', ob_get_clean(), $matches);
            print $matches[2];
            print "</div></center>";
        }
    }
```

Once the complete page is built the page is then printed to the output stream.

Localisation

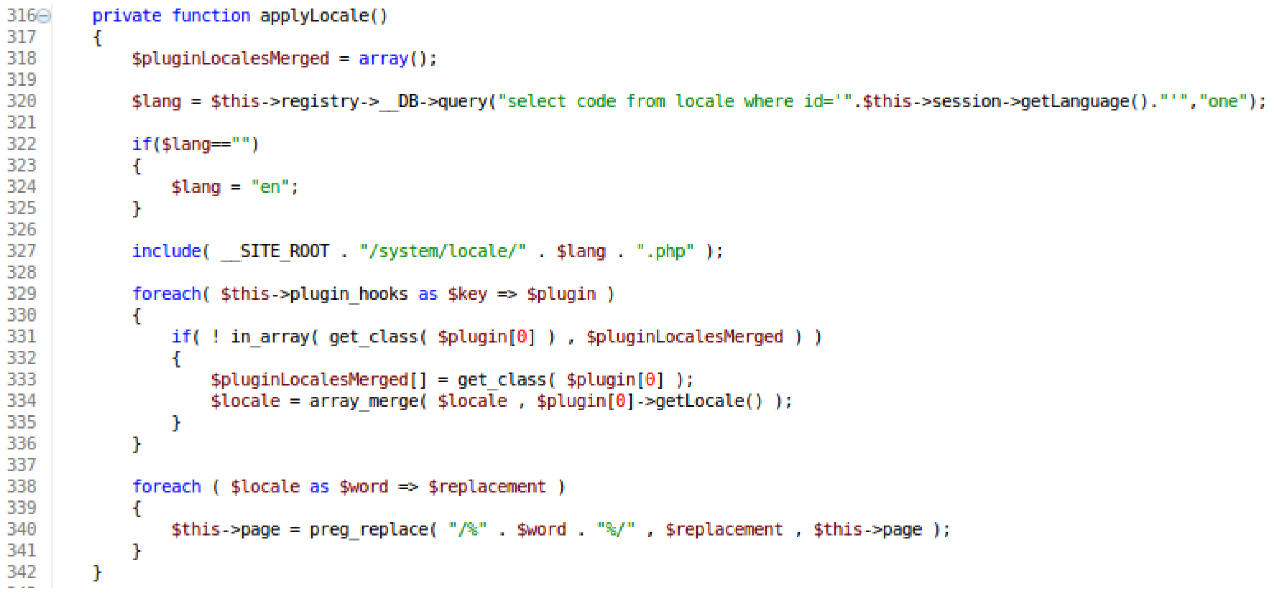
Localisation support was implemented using the Google translate service. Language packs where created using associative arrays. The key represented the text to replace and the value was the replacement text.



```php
<?php

 $locale = array(
    "welcome" => "Bienvenue",
    "Username" => "Nom d&#39;utilisateur",
    "Password" => "Mot de passe",
    "password" => "Mot de passe",
    "login" => "connexion",
    "logout" => "logout",
    "about" => "à propos de",
    "news" => "nouvelles",
    "translate" => "traduire",
```

The last step in the view before output the html page is replacing all matching %welcome% using a regular expression replacement. The locality is determined by the users $_SESSION['locale'] and the correct language pack is included. Lines 320 - 327

In the case of %welcome% it would be replaced by "Bienvenue". Lines 338 – 341



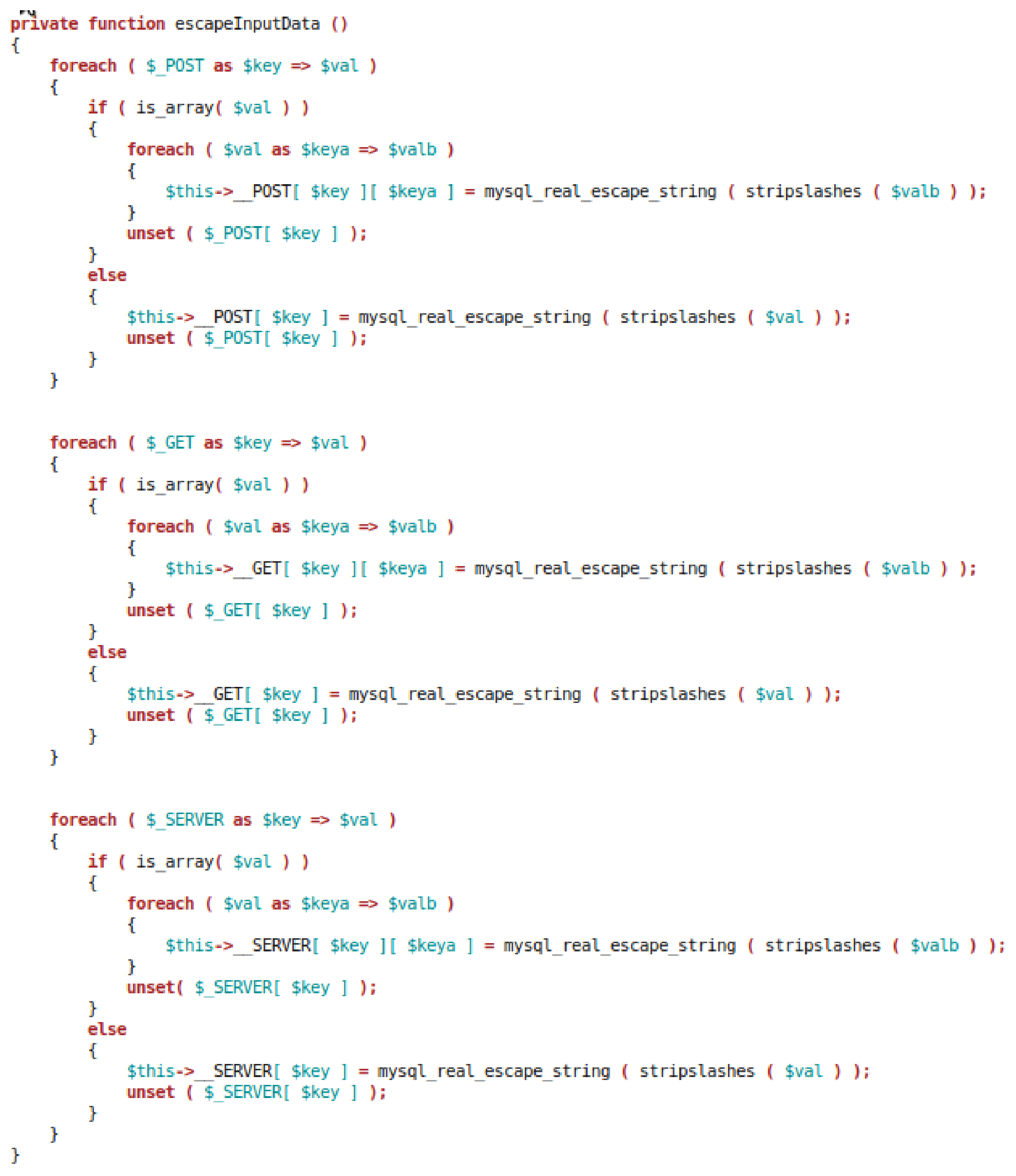
```php
316     private function applyLocale()
317     {
318         $pluginLocalesMerged = array();
319 
320         $lang = $this->registry->__DB->query("select code from locale where id='".$this->session->getLanguage()."'","one");
321 
322         if($lang=="")
323         {
324             $lang = "en";
325         }
326 
327         include( __SITE_ROOT . "/system/locale/" . $lang . ".php" );
328 
329         foreach( $this->plugin_hooks as $key => $plugin )
330         {
331             if( ! in_array( get_class( $plugin[0] ) , $pluginLocalesMerged ) )
332             {
333                 $pluginLocalesMerged[] = get_class( $plugin[0] );
334                 $locale = array_merge( $locale , $plugin[0]->getLocale() );
335             }
336         }
337 
338         foreach ( $locale as $word => $replacement )
339         {
340             $this->page = preg_replace( "/%" . $word . "%/" , $replacement , $this->page );
341         }
342     }
```

Depending on the locality is use the DOCTYPE for the html output was also replaced to confirm to the internet standards and ensure correct layout and reading orientation in the case of right to left languages.

Security

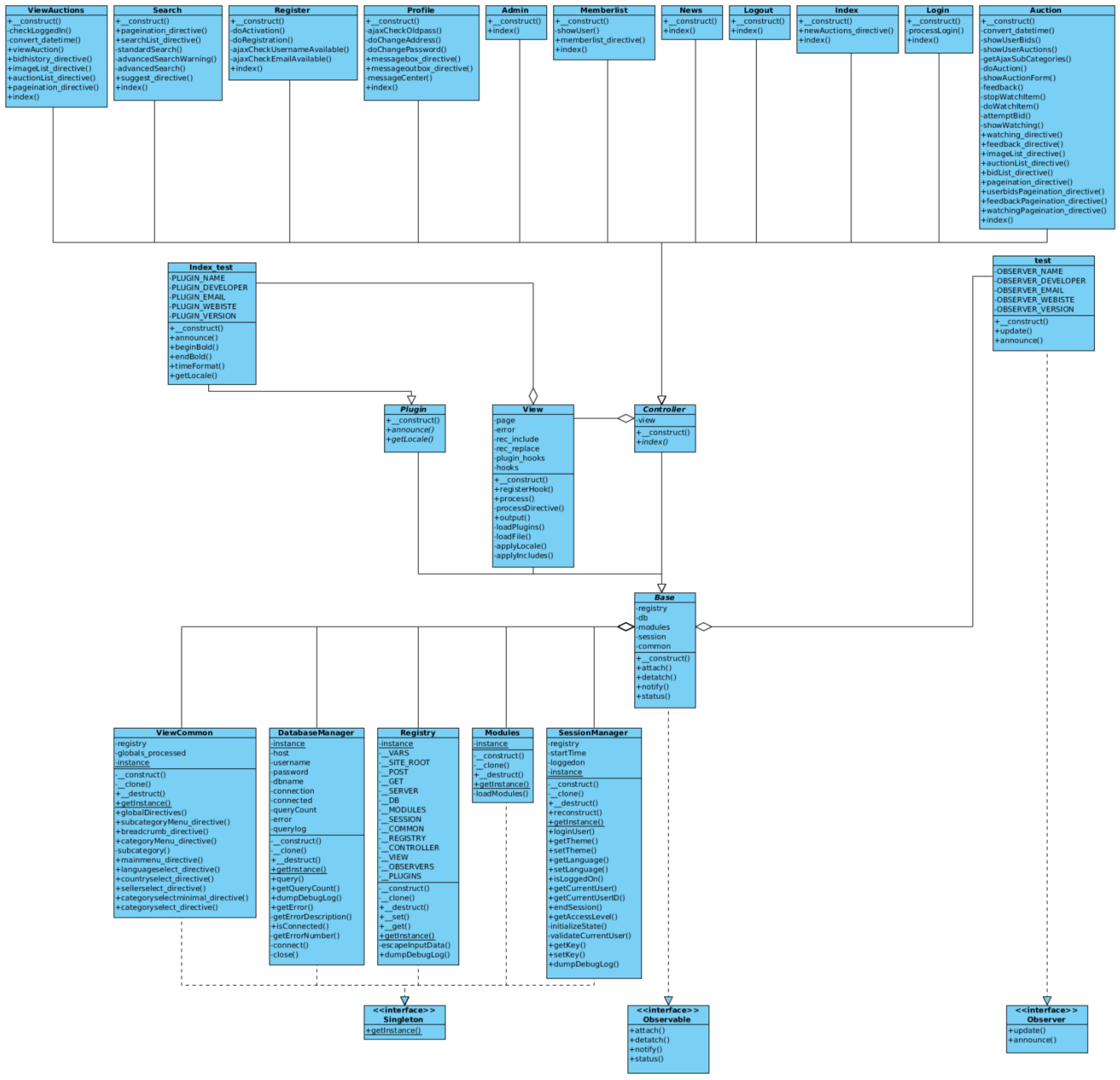
First level of security is the un-setting the $_POST, $_GET and $_SERVER. These session datasets are stored in the registry class for later retrieval for use in the controllers. Each key pair in the datasets is stripped of slashes and then using ‘mysql_real_escape_string’ is prepared for database usage.

As the inheritance graph indicates in the UML class diagram all of this processing is done before any of the controller logic, forcing the programmer to use the inbuilt protection overrides. Further checking is done in the Controller logic to ensure safe data.



```
private function escapeInputData ()
{
    foreach ( $_POST as $key => $val )
    {
        if ( is_array( $val ) )
        {
            foreach ( $val as $keya => $valb )
            {
                $this->__POST[ $key ][ $keya ] = mysql_real_escape_string ( stripslashes ( $valb ) );
            }
            unset ( $_POST[ $key ] );
        }
        else
        {
            $this->__POST[ $key ] = mysql_real_escape_string ( stripslashes ( $val ) );
            unset ( $_POST[ $key ] );
        }
    }

    foreach ( $_GET as $key => $val )
    {
        if ( is_array( $val ) )
        {
            foreach ( $val as $keya => $valb )
            {
                $this->__GET[ $key ][ $keya ] = mysql_real_escape_string ( stripslashes ( $valb ) );
            }
            unset ( $_GET[ $key ] );
        }
        else
        {
            $this->__GET[ $key ] = mysql_real_escape_string ( stripslashes ( $val ) );
            unset ( $_GET[ $key ] );
        }
    }

    foreach ( $_SERVER as $key => $val )
    {
        if ( is_array( $val ) )
        {
            foreach ( $val as $keya => $valb )
            {
                $this->__SERVER[ $key ][ $keya ] = mysql_real_escape_string ( stripslashes ( $valb ) );
            }
            unset( $_SERVER[ $key ] );
        }
        else
        {
            $this->__SERVER[ $key ] = mysql_real_escape_string ( stripslashes ( $val ) );
            unset ( $_SERVER[ $key ] );
        }
    }
}
```

Prototype UML Class Diagram



Copied materials

- JQuery
  - JavaScript Framework
    - http://jquery.com/
- JQueryUI
  - Tabs, Date picker, Dialog, effects
    - http://jqueryui.com/
- JQuery Plugins
  - Ajax Upload
    - http://plugins.jquery.com/project/uploadify
  - Cross Slide (images)
    - http://plugins.jquery.com/project/easyslideshow
  - Format currency
    - http://plugins.jquery.com/project/currency
  - Table Sorter
    - http://plugins.jquery.com/project/tableSort
  - Sizeable text area
    - http://plugins.jquery.com/project/TextAreaResizer
  - Time Picker
    - http://plugins.jquery.com/project/timeEntry
- Fancy Box
  - Image Preview
    - http://fancybox.net/
- GTranslate
  - Google Translate API Class
    - http://code.google.com/p/gtranslate-api-php/
- Regular Expressions
  - Email
    - http://php.net/manual/en/function.preg-match.php
  - Date
    - http://dotnetslackers.com/Regex/re-16542_Regex_matches_a_date_in_dd_mm_yyyy_format.aspx
- Reports Diagrams
  - Db Visualizer (Entity relationships)
    - http://www.dbvis.com/
  - yEd Graph Editor (Flow Charts)
    - http://www.yworks.com/
  - Visual Paradigm For UML
    - http://www.visual-paradigm.com/
- Theme
  - Simple auction template
    - http://www.freewebtemplates.com