```
<?
/*
 * A static class to send automatic e-mails
 *
 *
 *  All of the automatic emails of this system are sent through this class.
 *  It uses a template system by which the admin users can define the
 *  the texts of the emails. Within the text the admin can set some
 *  variables like name and client number which will be replaced by
 *  name and the client number of the receiver.
 *
 *  This class is the base abstract class. The child classes include
 *  the data if the template and the used variables.
 *
 *
 *   Simple example template:
 *   ------------------------------------------------------------
 *    Dear Mr CLIENT.NAME (CLIENT.NRO)
 *
 *    Thanks for orderring PRODUCT.NAME, (Id: PRODUCT.ID)
 *    Price ORDER.PRICE
 *
 *    Thank You
 *    xxxx Sales Team
 *    www.xxx.com
 *    tel 2345676
 *    email sales@xxx.com
 *   -------------------------------------------------------------
 *
 *    On the CMS the admin can modify the the content with lowe cases.
 *    Also there is a possiblity for translations in many languages.
 *
 */



abstract class  Core_Models_Business_Mail extends Zend_Mail
{

     protected  $mUsedKeys = array();
     protected  $mTypeKey;
     protected  $mMailData;
     protected  $mLocale;
     protected  $mModule;

     private    $mConfig;
     private    $mMailConfig;
     private    $mMailTypeTemplate;
     private    $mDebugBody;
     private    $mSendersDataSet = false;


     /**
      * construct
      *
      * @return     void
      * @access     public
      */
     public function __construct() {
        $this->mConfig          = Zend_Registry::get('configCore');
        $this->mMailConfig      = $this->mConfig->mailSettings;
        $this->mMailTypeTemplate    = new
        Core_Models_Business_Db_MailTypeTemplate($this->mModule);
     }


     /**
     * Send and email
     *
     * @return void
     * @access public
     */
    public function send()
    {
        if  (!$this->mSendersDataSet) {
            // Set default from e-mail, this can be overriden by function
            //  'setSenderData'
            $this->setFrom($this->mMailConfig->mijservicePuntEmail,
            $this->mMailConfig->mijservicePuntName);
        }

        $this->composeSubjectAndBody();
`       //  Send email. In the test and development enviromnment
       //  this can sent to a test mail or the content can be
       // written in to a log file (See variables  mMailConfig in
       // configCore.ini)
        if ($this->mMailConfig->sendMail ) {
            $this->send();
        }
        if ($this->mMailConfig->writeToLog ) {
            error_log("\n ================================= MAIL
================================");
            error_log("\n From : "      . $this->getFrom()     . "\n");
            error_log("\n To : "        . implode(',', $this->getRecipients())
. "\n");
            error_log("\n Subject : "   . $this->getSubject()  . "\n");
            error_log("\n Body : "      . $this->mDebugBody . "\n");
            error_log("\n
===========================================================================");
        }
    }


    /**
     *  get used  template Keys
     *
     * Return the keys used in the template
     *
     * @return array
     * @access public
     */
    public function getUsedTemplateKeys()
    {
        return $this->mUsedKeys;
    }


    /**
     * set data for for the email and its template
     *
     * This function takes the following paramters
     *
     * $aTemplateParams     template paramters to be replaced in the
     *                      template.
```

```
 * $aLocale             Locale (language) to be used
 *
 * @param array         $aTemplateParams
 * @param string        $aLocale
 * @return void
 * @access public
 */
public function setTemplateData( $aTemplateParams, $aLocale)
{
    foreach ($this->mUsedKeys  as $key) {
        $this->mMailData[$key] = $aTemplateParams[$key];
    }
    $this->mLocale = $aLocale;
}

  /**
 * set the receiver(s) of this mail
 *
 * $aRecipientsArr      recipients of this mail.
 *                      format:
 *                      $aRecipientsArr = array(
 *                              array(
 *                                  'email' => 'exsmple@example.nl',
 *                                  'name'  => 'Example Name'   )
 *                      )
 *
 * @param  array $aRecipientsArr
 * @return void
 * @access public
 */
public function setRecipients($aRecipientsArr )
{
    // Send only to tests mail if configured
    if ($this->mMailConfig->testEmail) {
        $this->addTo($this->mMailConfig->testEmail, 'Some Recipient');
        return;
    }

    foreach ($aRecipientsArr as $lRecipient)
    {
        $this->addTo($lRecipient['email'], $lRecipient['name']);
    }
}


 /**
 * set data for for the email and its template
 *
 * @param string    $aEmail
 * @param string    $aName
 *
 * @return void
 * @access public
 */
public function setSenderData($aEmail, $aName)
{
    $this->setFrom($aEmail, $aName);
    $this->mSendersDataSet = true;
}


/**
```

```
     * Compose Subject and Body from the given variables
     *
     * @param type $aComponents
     * @param type $aLocale
     * @return type
     */
    private function composeSubjectAndBody()
    {
       // $lTemplate = $this->mMailTypeTemplate->getTemplate($this->mTypeKey,
$this->mLocale);
        $lTemplate =
$this->mMailTypeTemplate->getTemplateFromCache($this->mTypeKey, $this->mLocale);
        $lTemplateSubject   = $lTemplate['subject'];
        $lTemplateBody      = $lTemplate['body'];

        foreach ($this->mMailData as $key => $value) {
          $lTemplateSubject = str_replace($key, $value, $lTemplateSubject) ;
          $lTemplateBody    = str_replace($key, $value, $lTemplateBody) ;
        }
        $this->mDebugBody = $lTemplateBody;
        $this->setSubject($lTemplateSubject);
        $this->setBodyText($lTemplateBody);
        $this->setBodyHtml(str_replace("\n", "<br>", $lTemplateBody));
    }



    /**
     * Attach pdf
     *
     * @param string    $aPdfFile   Name of the file with full path
     * @param string    $lFileName  Name of the attached pdf file in the email
     * @return void
     * @access public
     */
    public function attachPdf($aPdfFile, $lFileName)
    {
        $lFilecontents      = file_get_contents($aPdfFile);
        $lAtt               = $this->createAttachment($lFilecontents);
        $lAtt->disposition  = 'Zend_Mime::DISPOSITION_INLINE';
        $lAtt->encoding     = 'Zend_Mime::ENCODING_BASE64';
        $lAtt->filename     = $lFileName;
        $lAtt->type         = 'application/pdf';
    }


    /**
      * Create a mail class dynammcally base on the module and the typekey
      *
      * Typekeys refer to the classes defined in the folder
      * Application/Modules/CURRENT_MODULE/Models/Mail/
      * The parameter  $aTypeKey is the rewuired  class name.
      *
      * @param string  $aModule
      * @param string $aTypeKey   See above
      * @return  Core_Models_Business_Mail
      * @access public
      */
    static public function getClass($aModule, $aTypeKey)
    {
        require_once  $aModule. '/Models/Mail/' . $aTypeKey . '.php';
        $lMailClass = '' . $aModule. '_Models_Mail_' . $aTypeKey;
        $lMailObj   = new $lMailClass();
        return $lMailObj;
    }
}
```