Documention

Kimai
Open Source Time Tracking

Your Time.

Time Tracking for everyone

Table of Contents

Kimai

Welcome to the short documentation for the Kimai time-tracking application.

We could use your help improving the documentation. Please contact us in the forum if you want to help or read the Developer/Documentation section.

Download for offline usage

There are some downloads available for Kimai documentation:

- This document as PDF

There is also a German version, contributed by a Kimai user:

- German manual as DOCX
- German manual as ODT
- German manual as PDF

Installation

The following requirements must be met on the server installation:

- PHP version 5.4 or above (not tested with PHP 7.1 yet)
- MySQLi, php-xml and iconv extension for PHP
- A MySQL database (Kimai supports table prefixes)
- FTP or shell access to set the correct filesystem permissions.

Installation on Ubuntu

- Install apache (or any other webserver)
- Install mysqli
- Install php
- Install additional required module php-xml with `sudo apt-get install php-xml`

Extract the Kimai release zip/tarball into a server directory accessible from the outside and navigate to this location using your browser. A step-by-step installation program is included with every new release of Kimai. Follow the instructions of the installer and correct missing or incorrect permissions using your FTP or shell account. If the step-by-step installer does not work, please report this error using the issue tracker.

Remove the “install” directory after the installation!

After you removed the “install” directory you may login into Kimai using the following credentials:

- user: admin
- password: changeme

Change the administrator password using the settings dialog. Keeping “changeme” as your password is a major security implication!

Updates

Step 1

Create a full backup!

You need to backup both: your **database** and your file system, especially the file **includes/autoconf.php**.

Step 2

To install a new version just replace the entire directory with the new package and **KEEP** the file **autoconf.php** which can be found inside the /includes folder.

Additionally you have to set write permissions to the **temporary/** folder.

Attention: Kimai will NOT work with wrong permissions!

For Linux that could be:

```
chmod -R 777 temporary/
```

It would be even better, if you give writing permission to your PHP or Webserver user only, instead of making it world writable. The last can be or is a risk, especially in shared hosting environments.

Step 3

Ready for update:

Visit your Kimai installation with your web browser and the auto update process will perform all necessary tasks (eg. http://www.example.com/kimai/index.php - do **NOT** execute the installer!).

A new version updates the database automatically. Before that a backup of your data will be created. Normally you'll not even notice this process. If anything goes wrong during the database-update you can recover the data via phpMyAdmin.

Step 4

If the update was successful, finalize everything by removing the **updater** and **installer** folder. For Linux that could be:

```
rm -rf updater
rm -rf installer
```

Using Kimai

This is the Kimai manual - our end-user documentation. Its english version is not very comprehensive, we could use your help in improving it. Please contact us in the forum.

Timesheet

Thats where you track your time. We hope the interface is easy to understand and use.

Tracking your time

Just select a customer and all corresponding projects will be highlighted. Next step is to choose the project you are working on and last but not least the task you are doing. Now you are ready to track your time. Just hit the big green button in the top right corner to start tracking. To stop your current record hit the red stop-button either on the left of your current record or the top right corner. Afterwards you can choose another customer, project and/or task and continue to record.

Editing your time

If you have forgot to quit tracking or selected a wrong task: don't panic! There is a little pencil next to your recorded events. By clicking on it you will be able to rearrange a mistake that might have happened.

Attention: The start and end-time of events that have been recorded after that mistake will not be modified automatically. Have a look if there are overlapping entries.

If you are in need of some extra info to your current task use the comments bubble displayed beneath the running time. Here you can add whatever info you think can help you with explained what you did.

Administration

In the admin panel you can do every administration that is necessary for the running system.

Customers

Here you can add, change and delete customers. If you do not set a password the customer cannot login to Kimai. By selecting a group you enable all members of that group to see that customer. All fields except the customer name are optionally.

Projects

Here you can add, change and delete projects. Projects have to be assigned to a customer. Further you can set rates for this project if they differ from the standard rates (for details about rates see chapter Rates). If there is a budget it can be set here, too.

Tasks

Here you can add, change and delete tasks. A task has no relation to a customer or a project, but you can select which groups can use them. Additionally you can set rates here, too (for details about rates see chapter Rates).

Users

Here you can add, change and delete users. Users work with Kimai and can add, change or delete data like their own time records and other data. What a user can do depends on roles and rights. The roles and rights are set using global roles, membership roles and groups.

Groups

Here you can add, modify and delete groups and also define their membership role. Every group can have an unlimited amount of membership roles.

Global Roles

User get their permissions from their global roles and from their group membership

Membership roles

The membership role defines with role a user has inside of a group.

Status

Use this function to define status labels.

Advanced

Here you can define general settings like number of failed login-tries, bantime, currency, etc.

Database

Backup the database with the maintenance function before updating Kimai. Other database functions might be implemented here in the future.

Projects

Here you can add, change and delete projects.

Budget

Default rate: This rate will apply for every user and every task if no other rate overwrites it.

My rate: This rate is user specific and will only apply for the current user.

Fixed rate: No matter how much time a user works on a project, this amount of money will be payed.

Budget: The amount of money that is available for a project.

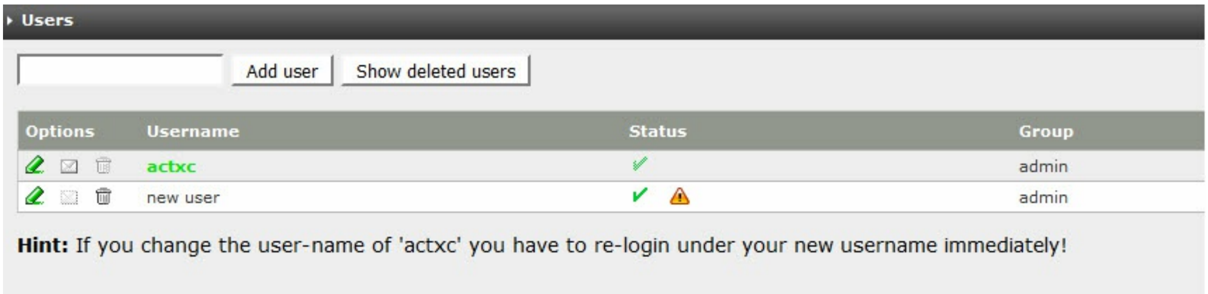
Effort: To be written...

Approved: To be written...

Users

Here you can add, change and delete users.

Users work with the Kimai application and can add, change or delete data like their own time records. They can edit also other data like customers, projects e.g. depending on their roles and rights. The roles and rights are set using global roles, membership roles and groups.

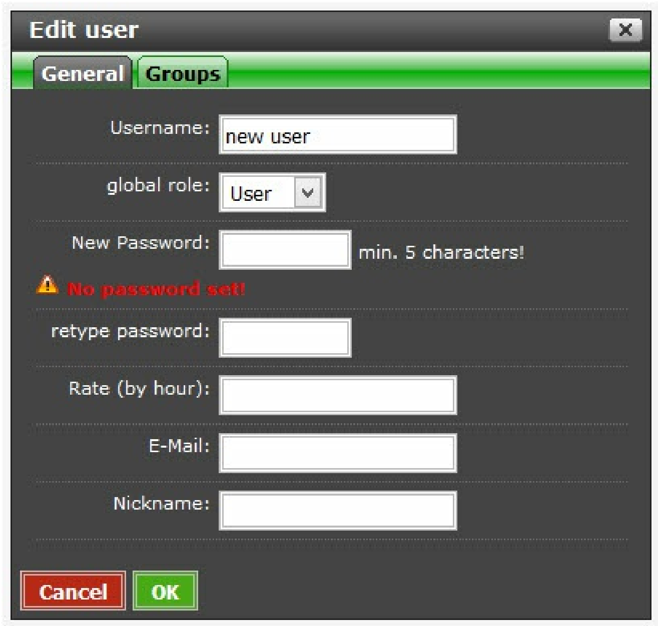


Add Users

Edit user > General

Add a name for the new user and press 'Add user'.

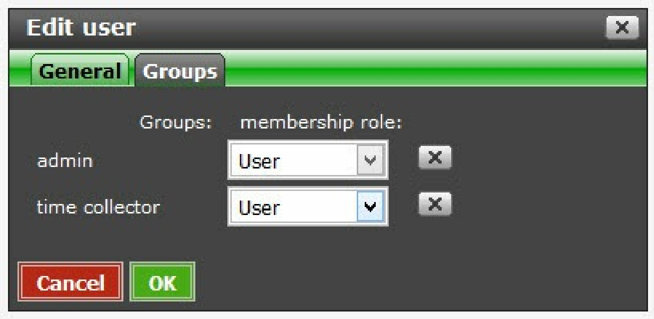
To let the user reset the password, a email adresse is necessary. The Nickname is used



Auser can only have one global role at a time but can belong to unlimited groups.

If the password field is empty when pressing OK, the password will not change. If a password was never set for the user, a warning is displayed in the dialog box and in the user list.

Edit user > Groups



If you enter a valid email address you can send emails from your local account by clicking the mailer symbol.

By default, a user is created in the group of its creator - wich is in most of the cases a user with admin roles. Make sure that you add and remove the correct roles for the new user. In most cases you need to remove the admin group.

Budget

This extension displays a graphical overview of your projects budget.

Expenses

Here you can add expenses to a project. Expenses affect the projects budget.

Export

With this extension you can export tracked time per User into an Excel-file, csv-file or a pdf-file. As a regular user you can only see your time. If you are at least a group leader you can select members of your group and export their data for reporting.

Invoices

Create invoices for your clients. Use can choose between several templates. One invoice is always cerated for one client, where you decide which projects will be included. You can also select a timeframe (limiting the timesheet entries) which will be used for generating the invocie.

In most cases you do not want to use an existing template. But you can modifiy existing or create new templates for you and your company.

There are also existing templates to download that are shared with the community.

Projects

What is an internal project?

An internal project is (as the name describes) only meant for internal tasks and visible to user and admins but NOT to customers.

Rates

If you have a look into the table kimai_rates you may see something like this:

User	Project	Event	Rate
1	2	5	10.00
2	NULL	5	20.00
2	2	NULL	30.00
2	NULL	NULL	40.00
NULL	2	5	50.00
NULL	NULL	5	60.00
NULL	2	NULL	70.00

The entries are already sorted. Kimai takes the first row that exists (maybe the one with the most hits).

Explanation of the table:

- user 1 on project 2 and task 5 has a rate of 10.00 € per hour
- user 2 on task 5 has a rate of 20.00 € per hour
- user 2 on project has a rate of 30.00 € per hour
- user 2 has a rate of 40.00 € per hour
- project 2 and task 5 have a rate of 50.00 € per hour
- task 5 has a rate of 60.00 € per hour
- project 2 has a rate of 70.00 € per hour

Some examples:

- User 3 is working on project 2: his rate is 70.00 €
- User 2 is working on project 1 and task 5: his rate is 20.00 €
- User 2 is working on any project except the defined ones for him: his rate is 40.00 €

This table can only be edited with SQL statements or a database-tool (05/03/2010).

Fees

How are fees calculated?

Here is the order in which is checked if a rate exists. The first one found is then used:

- my rate of the activity
- my rate of the project
- my rate set via preferences
- rate of the activity
- rate set for the project

See function `get_best_fitting_rate` in Database class.

Developer

There are several ways to get a copy of Kimai.

Releases

From time to time we will release ZIP archives. These are most often beta versions, to be tested by our user base for problems and bugs. Prodution ready releases will be linked specifically in the download section at our website.

You can download the current developer-version as ZIP from our master branch at GitHub.

From The Repository

To get the developer version use the repository. Please keep in mind, that this version is NOT stable! We like to receive bug reports to be faster with fixing them, but we can not guarantee support for these versions.

We recommend using a fork, so any changes made can be pushed back and then submitted back via pull-requests:

- Fork Kimai
- Clone it locally: `git clone https://github.com/YOUR-USERNAME/kimai.git`

Then, if you want to make changes and submit them, use a feature branch:

- `git checkout -b my-feature`
- `git commit -am "Added my feature"`
- `git push origin my_kimai`

Create a pull request from your GitHub repo page.

Build Kimai from source

Nightly Builds can be created by zipping the "core"-folder. Thats all you have todo.

For beta versions we have to edit the file "includes/version.php" and change $kga['status'] to "beta".

For a release we have to do the following:

1. Deactivate "Debug Extension" by prepending a # to the folder name
2. Change $kga['status'] to an empty string
3. Add error_reporting(E_ERROR); to "basics.php"

Introduction

This document gives an overview of how Kimai works. It is intended for developers (core and extensions).

Glossary

First lets talk about the wording used in Kimai to describe things:

Floater (or Dialog)

Those are the dragable “windows” which open as an overlay in the page. For example the preferences dialog or the dialog to add new timerecords.

Sublists

The lists in the lower part which show users, customers, projects and events.

Processor

PHP files which handle AJAX calls.

Extension

Every tab you see in Kimai (like timesheet, export, admin panel) is an extension.

Folder structure

- **core** these are the dispatcher files for kimai (entry points for web/mobile/api/floater)
- **css** stylesheets which are independant of any skin
- **extensions** every subfolders is one extension, for details see the chapter below
- **grfx** graphics which are independant of any skin (like the logo)
- **includes** Files which are needed in almost every other file.
  - **autoconf.php** Contains the configuration given by the user during setup. (Database connection information)
  - **basics.php** Simple wrapper to include all usually necessary files like functions and database layer.
  - **db_layer_mysql.php** as an example this is the database layer when the mysql extension is used to connect to the database
  - **func.php** Database independant functions are placed here.
  - **kspi.php** “Kimai Standard Processor Initialization” is included by every processor. Some setup is done here.
- **installer** Installer for Kimai.
- **js** Contains javascript files for the core.
- **languages** Every file is shown as a possible langauge (without the .php file extension) in the preferences dialog.
- **libraries** External libraries which are used (and possibly modified) by Kimai.
  - **jQuery** Some JQuery files, like datepicker.
  - **tcpdf** Library to create PDF files using pure PHP.
- **skins** Currently contains only one skin. Other folders will be shown as available skins. Only the styles.css is mandatory. But for the sake of readability you should split your stylesheets up into several files.
- **templates** Templates used by the core, e.g. login forms and sublists.
- **temporary** Used for temporary file creation.

Login Process

Login happens at the /index.php file. This is where you land when you open the kimai folder in your browser. Both, customers and users can login through the same login mask. For conflicting names (user and customer have the same name) the customer account takes precendence.

After a successfull login the user is redirected to /core/kimai.php. Here all extensions are loaded and the main user interface is being build.

Extensions

When you develop an extension you have to think of a name and a so called key. This key should be prepended to every javascript method, class, id, … just anything that is used within your extension to prevent conflicting names.

Every extensions needs a compile folder. This is where the smarty template engine stores the files which are passed to the client based on the template files from the templates folder. Floaters get their seperate folder under the templates folders. The folder css is obviously for CSS files which are needed by the extension. The js folder usually contains two javascript files. An _init.js and a _func.js file. The first should contain code to setup the extension. The second should only contains javascript functions which are used within this extension.

Only after an extension/tab is selected its content is being loaded. This is handled by the init.php file in the extensions folder. The processor.php file handles AJAX calls and floater.php creates those dragable windows.

Translations

Kimai is continually evolving so translations get outdated pretty fast. Therefore we're always happy, if people invest their time into improving the translations. Translations to new languages are very welcome as well!

Where to start?

The languages/ folder in Kimai is where all translation files are stored. All files in that folder, which end in .php are displayed to the user as possible translations. For now we use a short code to identify the langauge.

Format of the translation files

The files are simple PHP scripts which return an array (that in turn can contain arrays) which maps translation keys to their respective translation in the language. So if you have

`'ext_invoice' => array( “invoiceTitle” => “Create invoice”, “foo” => “Hello World” ),`

only “Create invoice” and "Hello World" is what is shown to the user and has to be translated. The strings “ext_invoice”, “invoiceTitle” and "foo" are the keys which Kimai uses to find the translated string (don't touch them).

Adding a new language

Doing so is pretty easy to describe, but it is some work for you. Take the file languages/en.php as starting point. This file contains all strings which can be translated in Kimai. Save that file under the short name of the language with the file extension .php (e.g. es.php for spanish) and start translating.

Updating a translation

If you want to improve an existing translation there are mainly two things you will do. Either you will change existing translated string or you will add missing translations. To do so check where the keys of the PHP arrays differ from the reference file en.php. If you find any keys missing in en.php you can remove them in your language file.

Publishing a translation

You can either sent the files in via a new issue or (what would make look even cooler) you sent us a Pull request.

Invoice templates

Creating templates for the invoice extension is actually pretty easy.

You can create templates Open Office documents or HTML pages. Existing templates are stored in the directory `extensions/ki_invoice/invoices/` .

There is a repository for sharing ready-to-use Invoice templates where you can find a Debug HTML invoice, which will help you in inspecting all possible variables while developing your own template.

Open Office templates

When writing an ODT (Text Document) or an ODS (Spreadsheet) file you can use following variables in the form of `[var.variablename]` (the brackets are required).

Variables you can use:

- `[var.today]` Current Date
- `[var.ttltime]` Total amount of hours
- `[var.invoiceID]` Defined as "CUSTOMER-NAME - YEAR - MONTH". See /extensions/ki_invoice/print.php.
- `[var.beginDate]` Selected starting date
- `[var.endDate]` Selected end date
- `[var.dueDate]` Defined as Current Date %2B 1 Month. See /extensions/ki_invoice/print.php.
- `[var.currencySign]` Currency sign (e.g. €)
- `[var.currencyName]` Currency name (e.g. Eur)
- `[var.invoiceDate]` Timestamp when this invoice was generated
- `[var.dateFormat]` Date format for formatting the invoiceDate (Kimai config: date_format_2)
- `[var.vatRate]` VAT rate
- `[var.vat]` Total VAT sum
- `[var.amount]` Total sum without VAT
- `[var.total]` Total sum including VAT

Customer related variables:

- `[var.companyName]` Company of the customer
- `[var.customerName]` Name of the customer
- `[var.customerStreet]` Street of the customer
- `[var.customerCity]` City of the customer
- `[var.customerZip]` Zip code of the customers city
- `[var.customerComment]` Comment, which is set for the customer
- `[var.customerPhone]` Phone number of the customer
- `[var.customerFax]` Fax number of the customer
- `[var.customerMobile]` Mobile number of the customer
- `[var.customerEmail]` Email address of the customer
- `[var.customerContact]` Name of the customer's contact person
- `[var.customerVat]` VAT of the customer

Project related variables:

An invoice can contain items for multiple projects. All projects are available from `[var.projects]` . Elements of the array can be accessed using a zero-based index. For example the name of the first project is `[var.projects.0.name]` .

- `[var.projects.0.name]` Name of the project
- `[var.projects.0.comment]` Comment of the project

A row has these variables:

- `[row.desc]` Name of the task (event)
- `[row.hour]` Duration of the timesheet entry in decimal hours
- `[row.fduration]` Duration of the timesheet entry formatted as "hours:minutes"
- `[row.amount]` Wage of the entry
- `[row.date]` Day of the entry
- `[row.description]` Description of the timesheet entry
- `[row.comment]` Comment of the timesheet entry

Problems with ODT documents

If you get a blank page or an error similar to the following

```
Fatal error: Uncaught exception 'tinyDocException'  with message 'the PHP global variable named 'c<text:span' does not  exist or is not set yet.' in
#0 extensions/ki_invoice/TinyButStrong/tinyButStrong.class.php(1870):  tinyDoc->meth_Misc_Alert(Object(clsTbsLocator), 'the PHP global  ...', true)
#1 extensions/ki_invoice/TinyButStrong/tinyButStrong.class.php(669):  clsTinyButStrong->meth_Merge_AutoVar('<?xml version="...', true)
#2 extensions/ki_invoice/TinyButStrong/tinyDoc.class.php(269):  clsTinyButStrong->Show(0)
#3 extensions/ki_invoice/print.php(204):  tinyDoc->saveXml()
#4 {main} thrown in extensions/ki_invoice/TinyButStrong/tinyDoc.class.php on  line 997
```

the variable name has a style applied. Select the (or all) variable name(s) and reset them to the default format. Then you can format the whole variable name as you want it.

HTML templates

Using HTML templates is possible since version 0.9.3.

If you want to create a new HTML template, you have to crete a new subdirectory in `extensions/ki_invoice/invoices/` with an `index.html` or `index.phtml` file inside.

For example, creating the HTML template called "ACME" you would create: `extensions/ki_invoice/invoices/ACME/index.html` . The easiest way is to duplicate the directory "my_company" and rename the copy.

Authenticator

By default kimai uses its internal user management, where users and passwords are stored in the Kimai database. But there are more authenticators, which can be used to connect to existing user repositories.

Configuration

The authenticator that will be used is configured in `includes/auth.php` within the key `$authenticator`.

```
$authenticator = 'kimai';
```

kimai is the last part of the classname without the namespace and the first character in lowercase. For example "ldap" comes from Kimai_Auth_Ldap: remove Kimai_Auth_ and lowercase the first character in the word "ldap".

If the used authenticator supports configuration parameters, you can set those with the file `includes/auth.php` (supported since Kimai > 1.0.1).

Therefor you need to create the file `includes/auth.php` with the content:

```
<?php
return array(
    'key_1' => 'value',
    'key_2' => 'value',
);
```

Set the parameters according to the authenticator documentation below ('key_1' and 'key_2' can be safely removed, they are just examples for the structure of the file).

If you use Kimai 1.0.1 or lower, you have to set these Configuration-parameters in the PHP files directly, for example in the LDAP class here.

Kimai

The built-in authenticator, using the Kimai database.

- Change `$authenticator = "kimai";` in `includes/autoconf.php`

It has no configuration parameters and is working out-of-the-box.

HTTP

A Basic-Auth authenticator

- Change `$authenticator = "http";` in `includes/autoconf.php`
- create .htaccess

```
AuthType Basic
AuthName "kimai"
AuthUserFile  /absolute/path/to/.htpasswd
Require valid-user
```

- Create a .htpasswd file online generator
- Login with `http://admin:changeme@kimai.localhost/index.php`

Configuration-parameters

- **HTAUTH_ALLOW_AUTOLOGIN:** Set true to allow web server authorized automatic logins
- **HTAUTH_FORCE_USERNAME_LOWERCASE:** Set true to search for lower-case usernames only
- **HTAUTH_USER_AUTOCREATE:** Set true to create Kimai user for web server authorized users
- **HTAUTH_PHP_AUTH_USER:** Check for PHP_AUTH_USER server variable
- **HTAUTH_REMOTE_USER:** Check for REMOTE_USER server variable
- **HTAUTH_REDIRECT_REMOTE_USER:** Check for REDIRECT_REMOTE_USER server variable

Default settings and full example for `includes/auth.php` :

```
<?php
return array(
    'HTAUTH_ALLOW_AUTOLOGIN' => true,
    'HTAUTH_FORCE_USERNAME_LOWERCASE' => false,
    'HTAUTH_USER_AUTOCREATE' => false,
    'HTAUTH_PHP_AUTH_USER' => false,
    'HTAUTH_REMOTE_USER' => true,
    'HTAUTH_REDIRECT_REMOTE_USER' => false,
);
```

LDAP

Basic LDAP authenticator.

- Change `$authenticator = "ldap";` in `includes/autoconf.php`

Configuration-parameters

- **LDAP_SERVER:** URL of your LDAP-Server
- **LDAP_FORCE_USERNAME_LOWERCASE:** Case-insensitivity of some server may confuse the case-sensitive-accounting system
- **LDAP_USERNAME_PREFIX:** Prefix for user LDAP query
- **LDAP_USERNAME_POSTFIX:** Postfix for user LDAP query
- **LDAP_LOCAL_ACCOUNTS:** Accounts that should be locally verified
- **LDAP_USER_AUTOCREATE:** Automatically create a user in Kimai after successful login

Default settings and full example for `includes/auth.php` :

```
<?php
return array(
    'LDAP_SERVER' => 'ldap://localhost',
    'LDAP_FORCE_USERNAME_LOWERCASE' => true,
    'LDAP_USERNAME_PREFIX' => 'cn=',
    'LDAP_USERNAME_POSTFIX' => ',dc=example,dc=com',
    'LDAP_LOCAL_ACCOUNTS' => array('admin'),
    'LDAP_USER_AUTOCREATE' => true,
);
```

Advanced LDAP-Authentication

An advanced LDAP authenticator, that allows further configuration options.

- Change `$authenticator = "ldapadvanced";` in `includes/autoconf.php`

Configuration-parameters

- **host:** This is the URI to connect with your LDAP-Server. This can be something like `ldap://ldap.example.com` or `ldaps://ldap.example.com:1234`
- **bindDN:** This is the DN of a user with read access to the LDAP. Leave empty when your LDAP supports anonymous bind
- **bindPW:** the password for the user with read access to the LDAP. Leave empty also for anonymous bind
- **searchBase:** Where do your searches start in the ldap. This is normaly something like `o=example,c=org`
- **userFilter:** What filter shall be used to search for a user. The string `%1$s` will be replaced with what the user entered as login name. You can use that string multiple times to enable login by UID **and** email. The filter would then be `(|(uid=%1$s)(mail=%1$s))`
- **groupFilter:** What filter shall be used to heck for group memberships. The string `%1$s` will be replaced by the value of the attribute defined by `usernameAttribute` of the user-entry. The string `%2$s` will be replaced by the DN of the users entry;
- **usernameAttribute:** The attribute to be sed to check for group memberships **as well as** retrieving the username to be used by kimai
- **commonNameAttribute:** This attribute defines the alias of the user in kimai
- **groupidAttribute:** This attribute contains the value that is represented in the `allowedGroupIds`
- **mailAttribute:** This attribute holds the users email-address that will be ported to kimai
- **allowedGroupIds:** An array of values defined by `groupidAttribute` . Members of the LDAP-groups referenced here will be allowed access to kimai!
- **forceLowercase:** Whether the username for kimai shall be lowercased or not.
- **nonLdapAcounts:** A list of kimai-usernames that shall **not** be autehnticated via LDAP but via the default kimai-authentication-adapter
- **autocreateUsers:** Shall uses authenticated via LDAP be created automatically in kimai. If set to false the users have to be added

manually to kimai and only password-verification will be handled via LDAP

- **defaultGlobalRoleName:** The name of the default role newly created users will be associated with
- **defaultGroupMemberships:** An array of group=>role mappings the user shall also be associated with

Default settings and full example for `includes/auth.php` :

```php
<?php
return array(
    'host' => 'ldap://localhost',
    'bindDN' => '',
    'bindPW' => '',
    'searchBase' => 'dc=example,c=org',
    'userFilter' => 'uid=%s',
    'groupFilter' => 'memberUid=%1$s',
    'usernameAttribute' => 'uid',
    'commonNameAttribute' => 'cn',
    'groupidAttribute' => 'cn',
    'mailAttribute' => 'mail',
    'allowedGroupIds' => array('kimai-access'),
    'forceLowercase' => true,
    'nonLdapAcounts' => array('admin'),
    'autocreateUsers' => true,
    'defaultGlobalRoleName' => 'User',
    'defaultGroupMemberships' => array('Users' => 'User'),
);
```

Active Directory

Kimai support authentication with Microsofts Active Directory through LDAP.

- Change `$authenticator = "activeDirectory";` in `includes/autoconf.php`

Configuration-parameters

- **enhancedIdentityPrivacy:** Supports the "Enhanced Identity Privacy" option, see Microsoft Technet

Default settings and full example for `includes/auth.php` :

```php
<?php
return array(
    'enhancedIdentityPrivacy' => 'false',
);
```

As this class is a subclass of the LDAP-Advanced authenticator (see above), you can set all Configuration-parameters from there as well, for example the host:

```php
<?php
return array(
    'host' => 'ldap://localhost',
    'enhancedIdentityPrivacy' => 'false',
);
```

Extensions

For example the timesheet, export and expenses screen (which you get by clicking the corresponding tab) is called an extension. To disable an Extensions just add a hash (#) symbol in front of the folder name. To re-enable it, remove the hash.

For developing an extension please refer to the Kimai forum.

About

Kimai was developer by multiple persons, which spent their free time to develop this web-based time-tracking application.

Acouple of names are:

1. Thorsten Höltge
2. Severin Leonhardt
3. Kevin Papst
4. Oleg Britvin
5. Martin Klemkow
6. Vasilis van Gemert

Many more people contributed to Kimai:

RRZE - Regionales Rechenzentrum Erlangen, Maximilian Kern, Enrico Ties, Thomas Wensing, Kelvin Luck, Urs Gerig, Willem van Gemert, Torben Boe and HamBug Studios, Klaus Franken, Chris (Urban Willi), Andreas Berndt, Niels Hoffmann, Günter Hengsbach, Paul Brand, Joaquín G. de la Zerda, Allesandro Bertoldo, José Ricardo Cardoso.

See a list of all contributors (since we switched to GitHub) at https://github.com/kimai/kimai/graphs/contributors

Please note: This software is free of charge. That doesn't mean, that you are allowed to remove copyright-notes! Neither on the login-panel nor in the credits-window. You are not allowed to remove the link to the credits. If these rules are offending you or you don't agree with them - please DON'T use Kimai.

FAQ

Falls diese kurze FAQ deine Frage nicht klären kann oder Du mehr wissen willst, dann melde Dich bei GitHub und erstelle ein Support Issue.

Wann wird die nächste Version veröffentlicht?

Das Kimai Projekt wird von den Entwicklern in Ihrer Freizeit programmiert. Da dies kein kommerzielles Projekt ist, können wir der Weiterentwicklung leider nicht immer so viel Zeit widmen, wie wir es gerne würden.

Kann ich meine Zeitdaten sichern?

Der einfachste Weg ein Backup der Kimai Datenbank anzufertigen ist die Nutzung eines MySQL Backup Tools. Da wären z.B. die Export Funktion von phpMyAdmin, das Tool MySQL Commander oder phpMyBackupPro.

Wieso kann ich keine Zeiten eintragen, die vor dem Installationszeitpunkt liegen?

"...ich habe das Problem, das ich keine Zeiten nachtragen kann, die älter als dem Installationsdatum sind. Ich würde gerne die Daten seit Beginn 02/09 eingeben. Jedoch wird mir im Dateselector das Installationsdatum angezeigt, sodass ich nicht erkennen kann, ob der ältere Eintrag übernommen worden ist. In der Datenbank finde ich den Eintrag."

Der Tag, an dem der Kimai-Kalender beginnt, lässt sich per Konfigurationsdatei nachträglich einstellen. Öffnen Sie dazu die Datei [kimai]/include/vars.php und verändern Sie folgenden Teil:

Von `$kga['calender_start'] = "";` zu `$kga['calender_start'] = "31/12/2008";`

Ich kann nur einen Teil der Zeiteinträge für einen Zeitraum sehen, aber nicht alle

Die Anzahl der angezeigten Datensätze ist standardmäßig auf 100 limitiert. Sie können diese Begrenzung unter Einstellungen (im Headerbereich neben dem Logout-Button) ändern.

Kann ich mich am Projekt beteiligen?

Wenn Du Interesse daran hast, Mitglied des Teams zu werden, melde Dich bitte zuerst bei GitHub an. Dank GitHub kannst Du uns jederzeit Pull-Requests zukommen lassen, auch die anderen Team-Mitglieder arbeiten so. Wenn Du viel beiträgst, nehmen wir Dich gerne in die Team Liste und die Liste der offiziellen Contributor auf :)

Wir suchen dringend jemanden, der gerne Dokumentation schreibt. Gute Englischkenntnisse sind Voraussetzung. Am liebsten native.

Du bist super in CSS und brauchst unbedingt etwas, womit Du Langeweile todschlagen kannst? Melde Dich, wir freuen uns über Mitstreiter die etwas von Frontend-Arbeiten verstehen.

Kann ich meine Einträge als Stundenzettel anzeigen/ausdrucken - z.B. "gruppiert nach Projekten und Tätigkeiten" oder "Alle Einträge für Kunde XYZ für die Zeit vom xx bis xx"?

In der aktuellen Beta-Version können die Einträge über die normale Benutzeroberfläche angezeigt werden, und es können PDFs erzeugt werden. Wenn Sie einen Zeitraum im oberen Teil des Fensters gewählt haben, werden ihnen in der Benutzeroberfläche nur Zeiten bzw. Summen für diesen Zeitraum angezeigt. Eine weitere Anzeigemöglichkeit ist die für bestimmte Tage (tragen Sie dafür einfach den gleichen Tag in 'von' und 'bis' feld ein).

Es gibt in der 'Friendly Hacks' Sektion bereits ein paar Extensions, die dieses Problem angehen.

Natürlich ist geplant, dass Kimai in Zukunft selbst Reports erstellen kann. Aber leider dauert die Entwicklung seine Zeit - gerade wenn man 'nebenbei' noch einen Vollzeit-Job hat ....

Ich möchte Beta-Tester werden. Wo bekomme ich die aktuelle Kimai-Version?

Der Beta-Test der neuesten Entwickler Version läuft kontinuierlich, unser Team ist zu klein um alles alleine zu schaffen. Bei Interesse meldet Euch bitte bei GitHub.

Du kannst entweder die Online Demo verwenden (siehe Webseite) oder noch besser: Du clonst dir den aktuellen develop branch von GitHub und testest diesen.

Kimai Team Documentation

This part of the documentation is related to the Kimai development team, you can safely skip this part if you are a end-user of the Kimai time-tracking application.

Developing for Kimai

Please read our CONTRIBUTING GUIDELINES directly at Github.

Creating a new release

For creating a new release, you need to have ant installed

- Change the version number in the following files and push them to master:
  - build.xml
  - composer.json
  - includes/version.php
    - set the version
    - ALWAYS increase the revision, otherwise people using the develop branch will be stuck in the updater(!)
    - set the release status (dev for pre-release versions. empty for release)
- Switch to your bash (this script is for unix based systems)

```
cd /tmp/
git clone https://github.com/kimai/kimai.git kimai-master
cd kimai-master
git tag -a -f -m 'Tagged version x.x.x' x.x.x
git push origin x.x.x
ant build
```

- Now open GitHub, create a new "Release" as Draft and upload the release ZIP from /tmp/kimai/kimai_xxx.zip to this draft
- Add one or two sentences and a link to this versions milestone with the "closed" filter
- Publish the release draft
- Publish a new blog post regarding this new release (see below)
- Update kimai.json and change the version number and news link, commit and push it to master (so the update check in Kimai administration will catch it as well)
- Edit the GitHub release and add a link to the blog post
- Let the rest of the team know and spread the word (Twitter, Facebook ...)

How to publish a new blog article on the Kimai website

Writing a new blog post is a simple as:

- Fork the website repo
- Create a new .md file in the _posts directory
- Send a pull request

Writing documentation

If you find some quirks in the documentation or found something worth mentioning, please don't hesitate to extend the documentation and send in pull requests.

Our repository for the documentation is located at:

- https://github.com/kimai/documentation

We use Markdown to write the documentation and the great GitbookIO tools to write and publish it.

You can either use the GitBook Editor or write the Markdown directly in your favorite editor. If you are going to add Chapters or Articles manually, make sure to include them in the file SUMMARY.md.

If you want to convert HTML documentation to Markdown, you can for example use the online tool heckyesmarkdown.com or to-markdown.js

Compiling

To use the gitbook command, install gitbook client via npm

```
npm install -g gitbook-cli
```

To compile the HTML version of the documentation, use this command:

```
gitbook build . --title="Kimai Documentation" --github=kimai/kimai
```

Output can be found in the _book/ folder.

Team member can use the build.sh file to create a new book version and commit it directly (it will be published directly).