YAWIK Documentation

Release 0.28

CROSS Solution

Mar 29, 2017

Contents

1 About YAWIK **3**
1.1 Important Links for developers . . . 4

2 Installation **5**
2.1 Requirements . . . 5
2.2 Setup . . . 7
2.3 Configuration . . . 10

3 Upgrade **13**
3.1 0.24 => 0.25 . . . 13

4 Configuration **15**
4.1 Authentication . . . 15
4.2 Mail . . . 17
4.3 Jobs . . . 17
4.4 Sitename . . . 19
4.5 Apache . . . 19
4.6 MongoDB . . . 20
4.7 Debugging . . . 20

5 Modules **21**
5.1 Core . . . 22
5.2 Auth . . . 39
5.3 CV (Curriculum vitae) . . . 39
5.4 Applications . . . 40
5.5 Organizations . . . 41
5.6 CompanyRegistration . . . 41
5.7 Orders . . . 43
5.8 Jobs . . . 43
5.9 PDF . . . 49
5.10 Geo . . . 49
5.11 Solr . . . 50
5.12 YawikXingVendorApi . . . 51
5.13 Settings . . . 52
5.14 JobsByMail (Work in Progress) . . . 52

6 Guidelines **55**

6.1 Programming Guidelines related to code-maintenance . . . 55
6.2 Common JavaScript Trigger . . . 56
6.3 Naming Conventions . . . 56

7 Customize **57**
7.1 CSS . . . 58
7.2 Formular Fields . . . 58
7.3 View Helper Scripts . . . 59

8 API **61**
8.1 Transfering Jobs . . . 63

9 Frequently Asked Questions **65**
9.1 FAQ: Mails . . . 65
9.2 FAQ: Navigation . . . 65
9.3 FAQ: Translation . . . 67
9.4 FAQ: General . . . 67
9.5 FAQ: Documentation . . . 68
9.6 FAQ: XML Feeds . . . 68
9.7 FAQ: Customize Formulars . . . 68

10 Indices and tables **71**

Contents:

CHAPTER 1

About YAWIK

YAWIK offers a web based solution for managing job applications. Jobs ads can be entered or pushed to the system. The system assigns application forms to job ads. Applicants and Recruiters can connect to YAWIK using social networks.

So what is YAWIK?

YAWIK is a modular system for human resources. It can be used as a job board, as a simple data entry tool for job openings or as an application management system. It should give applicants the opportunity to quickly and easily create a Hire-Me-Page. Currently it is possible to integrate YAWIK into a corporate website by extending it with an own module (see *Customize*). On the long term it is designed to become a distributed system for connecting recruiters and applicants.

YAWIK is a PHP web application. It's based on ZF2 "Zend Framework 2" and mongo. The target group of YAWIK are companies and candidates.

As started in 2013, YAWIK is quite new but stable enough to be used by aprox 20 companies to manage applications.

Why do we do this?

We believe that:

- Candidates should be able to easily apply to a job advertisement
- Candidates should have sovereignty over their application data
- Recruiters should be able to easily find candidates
- Open Source and Human Resources fits together

How came YAWIK to be?

YAWIK was initiated by Carsten Bleek, owner of "CROSS Solution". "CROSS Solution" was able to convince customers about the YAWIK idea. An initial group of [sponsors](https://yawik.org/sponsoren/) was found, and YAWIK was born.

Important Links for developers

- Sources: https://github.com/cross-solution/YAWIK
- Demo: https://yawik.org/demo/
- Scrutnitzer: https://scrutinizer-ci.com/g/cross-solution/YAWIK/
- Coveralls: https://coveralls.io/github/cross-solution/YAWIK
- Openhub: https://www.openhub.net/p/YAWIK

CHAPTER 2

Installation

you need at least a webserver, a mongo database and PHP. We're developing using apache with mod_php5 enabled. But you can use ngix, too. If you don't have a local mongoDB available you can try a provider like mlab.com or google.

Requirements

- php >= 5.6.* (PHP7 is currently not supported)
- Zend Framework 2.5.* (we're currently updating to ZF3, ZF2 Support will be dropped in 0.29+)
- mongodb >= 2.4.* (you should use >= 2.6 ... see below)
- php5-mongo
- php5-intl
- php5-curl (only needed to install dependencies via composer)
- php5-xsl (only needed to install dependencies via composer)
- php5-openssl (only needed to install dependencies via composer)
- php5-mbstring (only needed, if the PDF module is used)

YAWIK should run on any OS, which supports the above software components. In real life, we've seen YAWIK running on Linux Ubuntu, Debian, FreeBSD and OSX. It's possible to run YAWIK on AWS.

On FreeBSD, make sure, the php fileinfo extention is available. Fileinfo extention is needed by validating file uploads.

The YAWIK development is done on Ubuntu Linux. It is tested on Precise 12.04 and Trusty 14.04, Xenial 16.04 and Debian 8.

Install mongo Database

YAWIK runs with mongo 2.4. So you can use the mongod version, which is shipped with your distribution. However, you should use a later version. Otherwise you have to enable the text search, which is disabled in 2.4 by default. In 2.6 and above the text search is enabled by default.

You can install e.g. mongo 3.2 by: (Our demo is running 2.6, development is done with 3.0 and 3.2)

https://docs.mongodb.com/manual/administration/install-on-linux/

We've installed mongo the following way:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
echo "deb http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.2 multiverse" |␣
↪sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list
sudo apt-get update
sudo apt-get install -y mongodb-org
```

If your linux comes with systemd, you can start your mongod with `service mongo start`. If you need an init script, because your linux comes with sysv, you can fetch it from mongodb github repository

```
cd /etc/init.d/
curl https://raw.githubusercontent.com/mongodb/mongo/master/debian/init.d > mongod
chmod +x mongod
update-rc.d mongod defaults
```

Start your mongod with `/etc/init.d/mongod start`

Install YAWIK on Debian 8

If like to run YAWIK in an LXC container on proxmox. If we do so, we prefere debian 8, because it ships with php5.6. So we do not have to downgrade to php5.6. On the other hand it comes with sysv. However, installing YAWIK on Debian 8 you have to install the following packages.

```
apt-get install php5-mongo libapache2-mod-php5 php5-curl php5-xsl \
                 php5-intl php5-common php5-cli php5-json php5 apache2 curl npm␣
↪uglifyjs
```

`npm` and `uglifyjs` are needed to install assets (boostrap, jquery, select2 ...). You're only need those packages, if you want to install YAWIK git `cdgit`

Install YAWIK on Ubuntu 16.04

We assume, you have a running mongod. Otherwise check *getting a mongo db*

YAWIK should run on all operating systems, which support PHP (currently php 5.5 and php 5.6). Due to the changed in the php extentions for mongo (php-mongo and php-mongodb) YAWIK currently does not support php7. We'll wait for doctrine-mongodb-odm until they fully support PHP7. This means, you have to downgrade php on Ubuntu Xenial 16.04. Add the ondrej/php repository to your apt source lists.

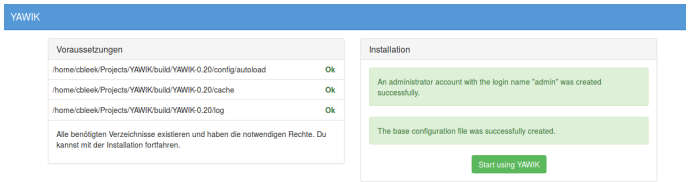
```
LC_ALL=C.UTF-8 add-apt-repository ppa:ondrej/php
aptitude update
aptitude install php5.6-mongo php5.6-curl php5.6-xsl php5.6-intl php5.6-common php5.6-
↪cli php5.6-json curl
```

Setup

Get the latest YAWIK Package from Sourceforge. Packages are build as ZIP or TGZ archive. They extract into a subdirectory YAWIK-x.y.z. If you preserve the permissions, the directories `cache` and `log` should be writable after extraction.

`tar` preserves permissions with the `p`-Option. So unpack a TGZ with `tar -xzpf YAWIK-y.x.z.tgz`. `unzip` preserves the permissions by default (at least on ubuntu 14.4). So unpack a ZIP archive with `unzip YAWIK-x.y.z.zip`

By pointing your browser to the `YAWIK-x.y.z/public` directory, an installation page appears. You'll be asked to enter a mongodb connection string, a username, a password and an email address.



Note: YAWIK will run in production mode by default. So if you make modifications to the config autoload files you have to remove the `cache/module-classmap-cache.module_map.php` and `cache/module-config-cache.production.php`.

Using Apache

If you want to use Apache, you probably need root access to the machine you've installed YAWIK on. In addition you need to enable the rewrite module of apache.

```
sudo a2enmod rewrite && sudo service␣
↪apache2 reload
```

Then you have to make sure that the DocumentRoot of apache is pointing to `YAWIK/public` and apache is allowed to Access the YAWIK directory.

A VirtualHost section might look like.

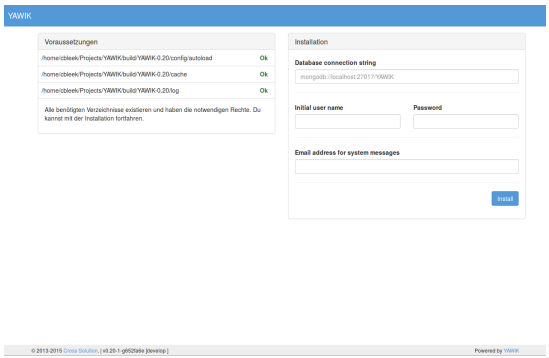
```
<VirtualHost *:80>
     ServerName yawik.example.com
     DocumentRoot /var/www/YAWIK/public
     AddDefaultCharset utf-8

     # set an env to disable caching.
     #SetEnv APPLICATION_ENV "development"

     <Directory /var/www/YAWIK/public>
          DirectoryIndex index.php

↪     Options Indexes FollowSymLinks MultiViews
          AllowOverride All
          # for apache >=2.4
          Require all granted

          # for apache <= 2.2
          # Allow from all
```



```
    </Directory>
</VirtualHost>
```

Place this in a file called `yawik.example.com.conf` in `/etc/apache2/conf` and execute

```
sudo a2ensite yawik.example.
↪com.conf && sudo service apache2 reload
```

now you should be able to login into your YAWIK by pointing a browser to

http://${YAWIK_HOST}

Note: Be sure you either export the variables YAWIK_HOST and YAWIK_HOME or replace them with the actual values in the apache config file.

Also your Webserver should not be able to access your build.properties. You can safely remove this file after you've run the installation is done.

Using Nginx

A configuration file for Nginx looks like this

```
server {
     listen         80;

      server_name my.yawik.host;

      root /your-location/YAWIK/public;
      index index.html index.htm index.php;
      charset utf-8;

      location / {
          try_files $uri $uri/ /index.php$is_args$args;
      }

      location ~ \.php$ {
          fastcgi_param  SCRIPT_FILENAME $document_root$fastcgi_script_name;
          fastcgi_pass unix:/run/php/php5.6-fpm.sock;
          fastcgi_param   APPLICATION_ENV  production;
          include /etc/nginx/fastcgi_params;
      }
}
```

Todo

We need more details on setup nginx here. - Where to put the server config - What commands to run.

Yawik can be downloaded at https://sourceforge.net/projects/yawik/files/

Setup for Developers

if you want to modify the YAWIK code, you should clone the repository from Github. The repository does not contain any dependency. You have to import all dependencies by executing the `install.sh` script located in the YAWIK root. This scripts imports all external libraries via composer. In addition, it creates the directories `log`, `cache` ùnd `config/autoload` and set the directory permissions to a+w.

```
git clone https://github.com/cross-solution/YAWIK
cd YAWIK
./install.sh
```

After the execution you are ready to point your browser to the `public` directory. You'll get the install wizard and after entering the initial user, the database connection and an email address you are ready to use YAWIK.

At this point your `` `config/autoload `` directory contains only one file `yawik.config.global.php` containing the database connection string. The initial user is created with the `àdmin` role in the database.

```
$ ls YAWIK/config/autoload
yawik.config.global.php
```

All other configurations are currently done manually by copying the `` `*.dist `` files from the modules configuration directory to the autoload directory and removing the ".dist" part.

Note: To disable the caching of the config autoload files you need to set an environment variable called `APPLICATION_ENV` to the value "development"

If you use apache, you can do this in your virtual section config with `SetEnv APPLICATION_ENV="development"`

Setup using composer

you can install yawik using composer

```
composer create-project cross-solution/yawik:dev-develop
```

This will clone the latest version from the develop branch, download all needed dependencies.

```
cd yawik
php -S localhost:8000 index.php
```

Point your browser to localhost:8000 and start using yawik

Example: Setting up Facebook, Xing or LinkedIn Login

```
YAWIK$ cp module/Auth/config/module.auth.global.php.dist config/autoload/module.auth.
↪global.php
```

All placeholders in the configuration files which match '%%.*%%' are deprecated. They are relics of the build.properties area. Since 0.20 an intall wizard is available which introduces an initial user with the `admin` role.

```
....
"keys"    => array ( "id" => "%%facebook.appid%%", "secret" => "%%facebook.secret%%"␣
↪),
....
```

Note: you need a Facebook, Xing or LinkedIn App, if you want to integrate the social networks . So take a look how to create an App with Facebook, Xing or LinkedIn.

Copy the *.dist files from the modules/*/config dir into the config/autoload directory. Don't forget to remove the "*.dist" suffix. Addjust the values and remove the cache/modules-* files.

Configuration

Configuration files are located in `config/autoload`. Config files are returning an associative array. All arrays are merged, so the order how the configuration files are processed is relevant.

Files with names ending in `*.global.php` are process first. As a second files ending in `*.{env}.php`. {env} can have at least the values `production`, and `development`. If the environment variable `APPLICATION_ENV` is set, and if files named `*. development.php` exist, then these configurations are processed. If no environment variable ist set, `production` is assumed.

At the end `*.local.php` files are processed.:

Modules are coming with there own `config` directory. Configuration files of modules can be named `*.config.php`. This allows you to split configurations into sections. E.g. a router.config.php file should contain an associative array defining routing specific things.

If the enviroment is set to `production`, all configurations are cached in `cahe/module-classmap-cache.module_map.php`. There is currently no way to invalidate the cache. You have to remove this file, if you alter files in `config/autoload`.

Authentication

to enable login via Facebook, Xing, LinkedIn or any other hybridauth adapter simply copy the module.auth.local.php.dist file to `config/autoload/module.auth.local.php` and adjust your keys and secrets.

```
<?php
return array(
     'hybridauth' => array(
     "Facebook" => array (
         "enabled" => true,
         "keys"    => array ( "id" => "", "secret" => "" ),
         "scope"       => 'email, user_about_me, user_birthday, user_hometown, user_
↪website',
     ),
     "LinkedIn" => array (
         "enabled" => true,
         "keys"    => array ( "key" => "", "secret" => "" ),
     ),
     "XING" => array (
         "enabled" => true,
         "keys"    => array ( "key" => "", "secret" => "" ),
     ),
     "Github" => array(
         "enabled" => true,
         'keys'    => array ( "id" => "", 'secret' => ""),
         "scope"   => ''
     ),
     "Google" => array(
```

```
23          "enabled" => true,
24          'keys'    => array ( "id" => 'xxxxxxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxxx.apps.
↪googleusercontent.com', 'secret' => ''),
25          "scope"   => 'https://www.googleapis.com/auth/userinfo.profile https://www.
↪googleapis.com/auth/userinfo.email',
26      ),
27  );
28  ?>
```

Debugging

you can enable the debugging Mode by setting the environment variable `APPLICATION_ENV=development`. This will increase the debug level, enable error messages on the screen and disables sending of mails to the recipients, stored in the database. You can overwrite the the all recipients (To, CC, Bcc) by setting `mail.develop.override_recipient=<your mail address>`

CHAPTER 3

Upgrade

1) backup your mongo data with the mongodump command. This will create a directory `dump` containing all your mongo databases. You can restore these databases with the mongorestore command.

YAWIK creates all needed mongo indexes automatically. But this only works, if an index is not already available. Since some indexes have changed in the past, it might be required to drop all indexes, so YAWIK will be able to create all needed indexes.

To drop all indexes, go to your mongo shell and type:

```
set1:PRIMARY> db.users.dropIndexes();
set1:PRIMARY> db.applications.dropIndexes();
set1:PRIMARY> db.jobs.dropIndexes();
```

2. Move your YAWIK Installation to a new location, so you are able to undo the upgrade any time.

3) Install the new Version. Eighter via `git` or unpack the latest ZIP/TGZ Package from sourceforge. In contrast to a fresh installation, you do not access your updated YAWIK via a Browser. Copy all `config/autoload/*` files of your moved old YAWIK installation into to `config/autoload` directory of your new installation.

4. Now you can access your new YAWIK via a Browser.

0.24 => 0.25

New users get a status. You can update old Users by

```
db.getCollection('users').update({'status': {$exists : false}}, {$set: {'status': {
    "name" : "active",
    "order" : NumberLong(50)
}}}, {multi: true})
```

Companynames ares searchable and sortable. If you want to make older companies searchable and sortable to, run the following query

```
db.getCollection('organizations.names').find().forEach(function(name) {
    db.getCollection('organizations').update({organizationName: name._id}, {$set: {_
↪organizationName: name.name}}, {multi: true});
})
```

CHAPTER 4

Configuration

Configuration files are located in `config/autoload`. Config files are returning an associative array. All arrays are merged, so the order how the configuration files are processed is relevant.

Files with names ending in `*.global.php` are process first. As a second files ending in `*.{env}.php`. {env} can have at least the values `production`, and `development`. If the environment variable `APPLICATION_ENV` is set, and if files named `*. development.php` exist, then these configurations are processed. If no environment variable ist set, `production` is assumed.

At the end `*.local.php` files are processed.

Modules are coming with there own `config` directory. Configuration files of modules can be named `*.config.php`. This allows you to split configurations into sections. E.g. a router.config.php file should contain an associative array defining routing specific things.

If the enviroment is set to `production`, all configurations are cached in `cache/module-classmap-cache.module_map.php` and `module-config-cache.production.php`. There is currently no way to invalidate the cache. You have to remove these files, if you modify files in `config/autoload`.

Authentication

to enable login via Facebook, Xing, LinkedIn or any other hybridauth adapter simply copy the module.auth.local.php.dist file to `config/autoload/module.auth.local.php` and adjust your keys and secrets.

```
    <?php
    return array(
         "Facebook" => array (
             "enabled" => false,
             "keys"    => array ( "id" => "your-consumer-key", "secret" => "your-consumer-secret" ),
             "scope"   => "email, user_about_me, user_birthday, user_hometown, user_work_history, user_education_history",// optional
             "display" => "popup"
         ),
```

```
        "LinkedIn" => array (
            "enabled" => true,
            "keys"    => array ( "key" => "your-consumer-key", "secret" => "your-consumer-secret" ),
            "scope"   => "r_fullprofile, r_emailaddress"
        ),
        "XING" => array(
            "enabled" => true,
            'keys'    => array ( "key" => 'your-consumer-key', 'secret' => 'your-consumer-secret'),
            "scope"   => ''
        ),
        "Github" => array(
            "enabled" => true,
            'keys'    => array ( "id" => 'your-consumer-key', 'secret' => 'your-consumer-secret'),
            "scope"   => ''
        ),
        "Google" => array(
             "enabled" => true,
             'keys'    => array ( "id" => 'your-consumer-key', 'secret' => 'your-consumer-secret'),
             "scope"   => 'https://www.googleapis.com/auth/userinfo.profile https://www.googleapis.com/auth/userinfo.email',

    ),
   );
   ?>
```

The configuration structure was simply taken from the hybridauth library. So the "enabled" field means enabled for the hybridauth library. It does not mean "enabled" for login. To enable a social network for login you have to ad the lowercased key to *ènableLogins* array. You have to copy the auth.options.global.php.dist to `config/autoload/auth.options.global.php` and adjust your values.

```
    $options = array(
            /*
            * default email address, which is used in FROM headers of system mails like "new registration",
            * "forgot password",..
            */
            'fromEmail' => 'email@example.com',

            /*
            * default name address, which is used in FROM headers of system mails like "new registration",
            * "forgot password",..
            */
            'fromName' => 'YAWIK Website',

            /*
            * Subject of your registration Mail
            */
            'mailSubjectRegistration' => 'your registration',

            /*
             * enable social networks for login and registration. The names must match the keys used in
             * in the 'hybridauth' section of you module.auth.global.php file
```

```
                */
                'enableLogins' => ['linkedin','github','xing','google','facebook'],

               /*
                * if true, users are allowed to register.
                */
                'enableRegistration' => true,

               /*
                * if true, users can reset their password.
                */
                'enableResetPassword' => true,
      );
```

Mail

To configure an SMTP Server, copy `Core/config/MailServiceOptions.config.local.php.dist` to your `config/` directory and adjust the values.

Setting the senders address

Setting Mail Texts

The mail texts are defined by the following templats. You can overwrite the mails by mapping the following keys

```
          'mail/job-created' => __DIR__ . '/../view/mails/job-created.phtml',
          'mail/job-pending' => __DIR__ . '/../view/mails/job-pending.phtml',
          'mail/job-accepted' => __DIR__ . '/../view/mails/job-accepted.phtml',
          'mail/job-rejected' => __DIR__ . '/../view/mails/job-rejected.phtml',
```

The mail texts can be translated by adding the languages to the mapping keys. The Logic is coded in: https://github.com/cross-solution/YAWIK/blob/develop/module/Core/src/Core/Mail/HTMLTemplateMessage.php#L246

```
               'mail/job-created.fr' => __DIR__ . '/../view/mails/job-created.fr.phtml',
               'mail/job-pending.fr' => __DIR__ . '/../view/mails/job-pending.fr.phtml',
               'mail/job-accepted.fr' => __DIR__ . '/../view/mails/job-accepted.fr.phtml
↪',
               'mail/job-rejected.fr' => __DIR__ . '/../view/mails/job-rejected.fr.phtml
↪',
```

Jobs

```
      $options = array(

               /**
               * If not set, the email address of the default user is used
               * @see Jobs\Options\ModulesOptionFactory
               */
               'multipostingApprovalMail' => '',

               /**
```

```
            * If a target Uri is set, a rest Request is sent to this target in case
            * a job posting was accepted.
            */
            'multipostingTargetUri' => '',

            /**
            * default Logo, if a company has no logo.
            */
            'default_logo' => '/Jobs/images/yawik-small.jpg',

            /**
            * Maximum size in bytes of a company Logo
            */
            'companyLogoMaxSize' => 100000,

            /**
            * Allowed Mime-Types for company Logos
            */
            'companyLogoMimeType' => array("image")
    );

    ### do not edit below ###

    return array('jobs_options' => $options);
```

Setting channels

Currently prices and channels are hard coded. The operator of YAWIK is responsible for publishing a jobposting to n ordered channel.

```
    $channel['yawik'] = array(
            'label' => 'YAWIK',
            'prices' => [ 'base' => 99, 'list' => 99, 'min'  => 99, ],
            'headline' => /*@translate*/ 'publish your job on yawik.org for free',
            'description' => /*@translate*/ 'publish the job for 30 days on %s',
            'linktext' => /*@translate*/ 'yawik.org',
            'route' => 'lang/content',
            'publishDuration' => 60,
            'params' => array(
                'view' => 'jobs-publish-on-yawik'
            )
        );

    $channel['jobsintown'] = array(
            'label' => 'Jobsintown',
            'prices' => [ 'base' => 650, 'list' => 698, 'min'  => 499, ],
            'headline' => '30 Tage, incl. Karrierenetzwerk',
            'description' => 'publish the job for 30 days on %s',
            'linktext' => 'www.jobsintown.de',
            'logo' => '/Jobs/images/channels/jobsintown.png',
            'route' => 'lang/content',
            'publishDuration' => 30,
            'params' => array(
                'view' => 'jobs-publish-on-jobsintown'
            )
        );
```

```
    $channel['fazjob'] = array(
            'label' => 'FAZjob.NET',
            'prices' => [ 'base' => 1095, 'list' => 1095, 'min'  => 1095, ],
            'headline' => '30 Tage auf dem Karriereportal der FAZ',
            'description' => 'publish the job for 30 days on %s',
            'linktext' => 'FAZjob.net',
            'logo' => '/Jobs/images/channels/fazjob_net.png',
            'route' => 'lang/content',
            'publishDuration' => 60,
            'params' => array(
                'view' => 'jobs-publish-on-fazjob-net'
            )
        );

    $channel['homepage'] = array(
            'label' => /*@translate*/ 'Your Homepage',
            'prices' => [ 'base' => 0, 'list' => 0, 'min'  => 0, ],
            'headline' => /*@translate*/ 'enable integration of this job on your Homepage',
            'description' => /*@translate*/ 'enable %s of this job on your Homepage',
            'linktext' => /*@translate*/ 'integration',
            'route' => 'lang/content',
            'params' => array(
                'view' => 'jobs-publish-on-homepage'
            )
        );

    return array('multiposting'=> array('channels' => $channel));
```

Sitename

Apache

point the DocumentRoot of your Webserver to the `public` directory.

```
<VirtualHost *:80>
      ServerName YOUR.HOSTNAME
      DocumentRoot /YOUR/DIRECTORY/YAWIK/public

      <Directory /YOUR/DIRECTORY/YAWIK/public>
              DirectoryIndex index.php
              AllowOverride All
              Order allow,deny
              Allow from all
      </Directory>
</VirtualHost>
```

Note: you should `SetEnv APPLICATION_ENV development` in your VirtualHost section, if you plan do develop.

MongoDB

Debugging

you can enable the debugging Mode by setting the environment variable `APPLICATION_ENV=development`. This will increase the debug level, enable error messages on the screen and disables sending of mails to the recipients, stored in the database. You can overwrite the the all recipients (To, CC, Bcc) by setting `mail.develop.override_recipient=<your mail address>`

Debugging Mails

CHAPTER 5

Modules

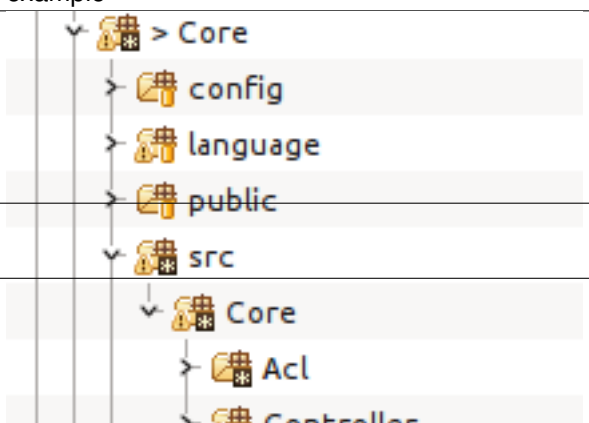
we use module system of the ZF2. Modules are configured in their `config` directory. You can use multiple configuration files by using the `\Core\ModuleManager\ModuleConfigLoader` utility. This way you can split up your configuration in smaller chunks (e.g. put all your configuration about routings into a `router.config.php` and about templating into a `template.congig.php`), which are easier to find, read and maintain.

Modules can simply be enabled by adding their names to an array in config/config.php.

```
<?php
$modules = array(
      'DoctrineModule',
      'DoctrineMongoODMModule',
      'Core',
      'Auth',
      'Cv',
      'Applications',
      'Jobs',
      'Organizations',
      'Settings',
      'Pdf',
      'Geo'
 );

...
?>
```

Directory Structure of a module

directory	description	example
language	contains gettext language files	
public	place for images, css or javascript	
config	place for configuration files	
test	place for unit tests	
src	Controllers, Models etc.	
view	view scripts	

A module can implement the following Features:

- Dashboard Widgets
- Configuration formulars
- Command line tools

currently the following modules exists:

Core

Contents:

Assets

Assets are common JS libraries like jquery, bootstrap or select2. It makes sense to manage these assets by npm. This means, all needed JS libraries are listed in *package.json*. If an additional library is needed, it can be added via `npm i --save-dev <packagename>`. This will update the *package.json* and download the package to the *node_modules* directory. Our *bin/install-assets.sh* copies all needed javascript, css, fonts, etc. files to the `public/assets` directory, which is accessable by the web server.

You can download all required JS files and copy them to their location in the assets dir with:

```
npm install
bin/install-assets.sh
```

You can allways remove and reinstall assets with

```
rm -R public/assets/*
npm install
bin/install-assets.sh
```

this will copy all needed JS files into `public/assets`.

Formular Handling

Author Mathias Gelhausen <gelhausen@cross-solution.de>

Forms are essential. YAWIK uses forms almost everywhere. The main goals of forms are:

- Consistent look&feel
- Binding to Entities

Build-In form classes and helpers

[TODO: Fill in..]

Form classes

\Core\Form\Form

This is the very base YAWIK form class, which extends \Zend\Form\Form.

It implements \Core\Form\DescriptionAwareFormInterface and \Core\Form\DisableElementsCapableInterface.

It sets the default hydrator to \Core\Entity\Hydrator\EntityHydrator (which allows the form to bind YAWIK entities.)

\Core\Form\BaseForm

An extension of \Core\Form\Form, which creates a form with a target fieldset and a default buttons fieldset.

It is meant to provide a ad-hoc solution for creating forms with a consistent look and handling.

\Core\Form\Container

The Container bundles several forms together, which work on one entity, enabling them to patch their behaviour together. Container have some specific methods for identifying or handle an explicit form. Most of these methods just pass some information to all subsequent forms

```
setParams(array $params)
```

is placing a hidden input filed in every subordinated forms. This comes in handy for the identification of an entitity.

```
setEntity(EntityInterface $entity)
```

Interfaces

View helpers

\Core\Form\View\Helper\FormContainer

\Core\Form\View\Helper\Form

\Core\Form\View\Helper\SummaryForm

Learning YAWIK forms

[TODO: Here must be some text....]

We try to make our forms' look and feel consistent across the application.

Therefor there are many form classes and view helpers available to help creating, handling and rendering forms.

Form

All YAWIK forms are handled by one javascript file, unless it has the html attribute `data-handle` set to a value other than `yk-forms`. This javascript bind on the submit event, makes a ajax call to the forms' action and takes care

of displaying the error messages, if any. It then triggers an own event called `yk.forms.done` and passes the ajax call result to all listeners.

BaseForm

Most of our forms share a common structure:

Some elements (inputs), grouped optionally in fieldsets and the "Submit" and "Cancel" buttons at the bottom.

To simplify creation of such forms, there's the \Core\Form\BaseForm class. You specify the fieldset with the elements, and the BaseForm adds a Button-Fieldset automatically.

The specified fieldset will be used as base fieldset of the form, so binding objects to the form will effectivly bind it to the fieldset. (see ZF-Doc)

Examples

Simplest BaseForm: Set the base fieldset to a form element manager service name.

```
class MyForm extends BaseForm
{
    protected $baseFieldset = 'MyFieldset';
}
```

Provide factory specification as base fieldset:

```
class MyForm extends BaseForm
{
    protected $baseFieldset = array(
        'type' => 'MyFieldset',
        'options' => array( /* ... */ ),
        /*...*/
    );
}
```

Overwrite parent methods to further customize:

```
class MyForm extends BaseForm
{
    protected function addBaseFieldset()
    {
        $fs = new Fieldset();
        /* configure your fieldset as you want */
        $fs->setUseAsBaseFieldset(true);
        $this->add($fs);
    }

    protected function addButtonsFieldset()
    {
        /* add the desired buttons fieldset here, e.g.: */
        $this->add(array(
            'type' => 'MyButtons',
        ));
    }
```

SummaryForm

The most used form will be the SummaryForm, which is an extension of BaseForm, that lets you render a form in two presentation modes:

- Form Mode (*Form Mode*)
- Summary Mode (*Summary Mode*)

Each presentation mode is rendered as a sub container in one html container, and one of these containers is hidden while the other is displayed.

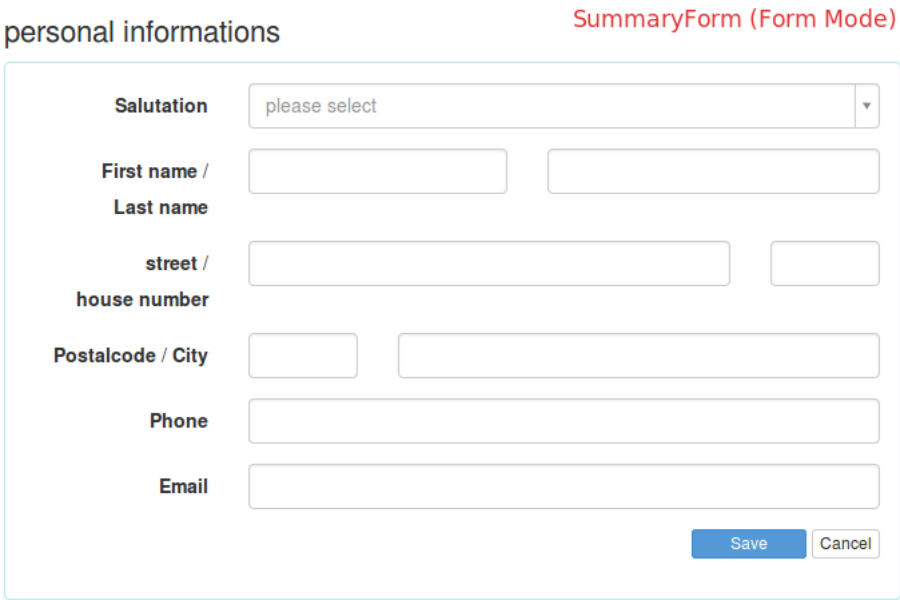


Fig. 5.1: Form Mode

The summary container gets a "edit"-button in the top right corner on hovering. Clicking this button will toggle the presentation modes. Clicking the submit or cancel button in the form will - in case of submit only if the results' valid field is true - toggle also the presentation modes.

The view helper SummaryForm takes care of rendering the form and includes the necessary Javascript files.

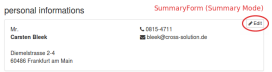


Fig. 5.2: Summary Mode

Note: The summary presentation renders only the base fieldset.

If the SummaryForms' base fieldset implements the \Core\Form\ViewPartialProviderInterface, it's possible to provide a view partial for the fieldsets' form view and summary view separately or provide one partial to render both modes. The view helper decides what to do on this criterions:

- To render the form part: It appends `.form` to the partial name and tries to resolve this template name using the ViewResolver. If this template name can be resolved, it is used to render the form, if not, the template with the original name is used.
- To render the summary part: It appends `.view` to the partial name and tries to resolve this template name using the ViewResolver. If this template name can be resolved, it is used to render the summary, if not, the template with the original name is used and a variable named "renderSummary" is passed with the boolean value "TRUE".

If no view partial is provided, it loops over the elements of the form and renders the elements as list of element labels, element values pairs.

Prior to rendering, the activated mode can be set. (Either `form` or `summary`)

The summary form javascript expects a field "content" in the ajax call result (json), which holds the rendered summary. This content then replaces the old summary content.

Examples

Create a summary form:

```
class MyForm extends SummaryForm
{
    protected $baseFieldset = 'MyFieldset';
}
```

render in view:

```
$this->summaryForm($form);
```

Handle in controller:

```
public function myFormAction()
{
    $services = $this->getServiceManager();
    $forms    = $services->get('FormElementManager');
    $form     = $forms->get('MyForm');
    $request  = $this->getRequest();

    if ($request->isPost()) {
        $form->setData($request->getPost());
        if ($form->isValid()) {
            $helper = $services->get('ViewHelperManager')->get('summaryform');
            return new JsonModel(array(
                'valid' => true,
                'content' => $helper($form)
            ));
        } else {
            return new JsonModel(array(
                'valid' => false,
                'errors' => $form->getMessages(),
            ));
        }
    }

    return array(
        'form' => $form
    );
```

To render Using ViewPartialProviderInterface in a SummaryForm (remember to use the base fieldset to set the partial...)

```
class MyForm extends SummaryForm
{
    protected $baseFieldset = 'MyFieldset';
}

class MyFieldset extends Fieldset implements ViewPartialProviderInterface
{
    protected $partial = 'my-form';

    public function getViewPartial()
    {
        return $this->partial;
    }
```

```
    public function setViewPartial($partial)
    {
        $this->partial = $partial;
        return $this;
    }
}
```

Render both presentation modes in one partial "my-form.phtml":

```
<?php if ($this->renderSummary): ?>
<!-- create the summary view, access the form with $this->form -->
<?php else: echo $this->summaryForm()->renderForm($this->form); ?>
```

Render the presentation modes in separate views "my-form.form.phtml" and "my-form.view.phtml"

```
<!-- my-form.form.phtml -->

<?php echo $this->summaryForm()->renderForm($this->form) ?>


<!-- my.form.view.phtml -->
<?php
    /* $this->renderSummary is NOT set, when using separate view scripts. */
    echo $this->summaryForm()->renderSummary($this->form)
?>
```

Container

Simple Form

Navigation

YAWIK uses Zend/Navigation. The following example shows, how you can modify the navigation. Our YawikDemoJobboard makes YAWIK running like a Jobboard. On a jobboard a navigation normally contains a public link to employers, who are offering jobads. No authentication is required to see the list of companies. This can be configured like:

```
'acl' => array(
       'rules' => array(
               // guests are allowed to see a list of companies.
               'guest' => array(
                       'allow' => array(
                               'route/lang/organizations',
                        ),
                ),
        ),
 ),
```

If YAWIK runs as an Applicant Tracking System (like in our YawikDemoSkin), a list of companies may contain the customers of a hr company. Such a list must not be show to the public. This can be configured like:

```
'acl' => array(
       'rules' => array(
               // guests must not see a list of companies.
```

```
                'guest' => array(
                        'deny' => array(
                                'route/lang/organizations',
                         ),
                 ),
                // recruiters see the link
                'recruiter' => array(
                        'allow' => array(
                                'route/lang/organizations',
                         ),
                 ),
        ),
 ),
```

Pagination

Author Mathias Gelhausen <gelhausen@cross-solution.de>

Pagination in the view layer

Author Mathias Gelhausen <gelhausen@cross-solution.de>

Syntax

The process described here assumes you render the actual content (the item list) of the pagination container in a separate view script which allows you to load subsequent pages with an ajax request.

Add a pagination container in the main view script:

```
<div class="pagination-container">
     <div class="pagination-content">
     <!-- The item list should be rendered in here -->
     </div>
</div>
```

This is the most basic container. The corresponding javascript will add a for the "empty" message and a for the loading indicator.

The actual HTML will then look like

```
<div class="pagination-container">
     <div style="position:absolute;
                 z-index:1000;
                 top: 0;
                 left: 0;
                 width: 100%;
                 height: 100%;
                 background-color: rgba(250,250,250,0.5);"
          id="jobs-list-container-loading-indicator"
          class="pagination-loading"
     >
         <i class="fa-spin yk-icon-spinner yk-icon fa-2x"
            style="position:absolute; top: 25%; left: 50%;">
         </i>
```

```
    </div>

    <div class="pagination-message alert alert-warning">
        <strong>Sorry</strong>, your search yields no results.
    </div>

    <div class="pagination-error alert alert-danger">
        <strong>Sorry</strong>, loading results failed.'
    </div>

    <div class="pagination-content">
    <!-- The item list should be rendered in here -->
    </div>
</div>
```

It is possible to alter any of the divs by simple render an element with the corresponding class name in the pagination container. If you want to simply alter the messages to be displayed you can also do this with data-* attributes on the container div

```
<div class="pagination-container"
     data-message="Any valid escaped html content"
     data-error="This message will be injected to the div.pagination-error"
>
```

The javascript will bind to the click events of any link inside an element with the class "pagination", as the PaginationControl view helper of the Zend Framework will do.

Once such a link is clicked, the loading div is displayed, an AJAX request is issued to the url of that links href attribute and the content of the div.pagination-content is replaced by the response. The javascript will bind to the click events of any link inside an element with the class "pagination".

If an empty string is returned from the AJAX request, the div.content will get hidden and the div.message will be displayed. On an error the div.error is displayed.

You can trigger a load programmatically with javascript

```
$('.pagination-container').paginationContainer('load', '/the/url/to/load/from?
↪with=parameter');
```

Example

This is taken from the Jobs Module and is the pagination container of the Jobboard. $jobs in this case is the paginator service passed along from the controller.

```
<?php //description: Renders the list of public jobs. ?>
<?php $this->headTitle($this->translate('Jobs'));
      $this->headScript()->appendFile($this->basepath('/Core/js/core.pagination-
↪container.js'))?>

<h1><?php echo $this->translate('Public Job Opportunities')?></h1>

<?php echo $this->flashMessenger()->render('default', array('alert', 'alert-success
↪')) ?>

<nav class="navbar yk-toolbar" id="jobs-list-filter-wrapper">
<?php echo $this->form($this->filterForm)  ?>
```

```
</nav>

<div id="jobs-list-container" class="pagination-container"
    data-message="<?php echo $this->escapeHtmlAttr(sprintf(
                        $this->translate('%sSorry%s, there are not any jobs matching␣
↪your search criteria.'),
                        '<strong>', '</strong>'
                ))?>">

    <div class="pagination-content">
    <?php echo $this->render('jobs/jobboard/index.ajax.phtml')?>
    </div>
</div>
```

and the script which renders the items:

```
<?php if (count($jobs)): // We only want to render something, if there are items.?>
<table class="pagination-content table table-striped table-bordered table-hover" id=
↪"jobs-list">
    <thead>
    <tr>
        <th><?php echo $this->translate('Title of the job')?> / <?php echo $this->
↪translate('Companyname')?></th>
        <th><?php echo $this->translate('Location')?> / <?php echo $this->translate(
↪'Date of receipt')?></th>
        <th><?php echo $this->translate('Apply')?></th>
    </tr>
    </thead>

<?php foreach ($jobs as $job):?>
<tr>
    <td>
        <?php if ($job->organization && $job->organization->image &&  $job->
↪organization->organizationName): ?>
            <div class="yk-logo-list">
                <img class="yk-logo-sm" src="<?php echo $this->basePath().$job->
↪organization->image->uri ?>">
            </div>
        <?php endif ?>
        <?php $href = $job->link ? $job->link : $this->url('lang/jobs/view', array(),␣
↪array('query' => array('id' => $job->id)), true); ?>
        <a href="<?php echo $href ?>" target="_blank"><?php echo strip_tags($job->
↪title)?></a>
        <br/><?php
            if (isset($job->organization) && isset($job->organization->
↪organizationName) && isset($job->organization->organizationName->name)) {
                echo $job->organization->organizationName->name;
            }
        ?>
    </td>
    <td>
        <div><?php echo $job->location?></div>
        <small>
            <?php
            if ($job->datePublishStart): echo $this->dateFormat($job->
↪datePublishStart, 'short', 'none');
            elseif ($job->dateCreated): echo $this->dateFormat($job->dateCreated,
↪'short', 'none');
```

```
            endif?>
        </small>
    </td>
    <td>
        <?php
            echo $this->applyUrl($job);
        ?>
     </td>
</tr>
<?php endforeach?>
    </table>

<?php echo $this->paginationControl($jobs, 'Sliding', 'pagination-control', array(
↪'lang' => $this->params('lang'), 'route' => 'lang/jobboard'));?>

<?php endif ?>
```

Notifications

YAWIK comes with a notification system to easily display notification messages to the user. These messages are persisted in the session and can be retrieved even after a redirection (e.g. Login)

Once a message is displayed (rendered), it is removed from the session.

Controller Plugin

Yawik provides the controller plugin "Notification" (service name "notification") to set notification messages in different namespaces.

It is merely a wrapper for Zend Framework's FlashMessenger. It provides own namespaces and shortcut methods to add notifications according to Twitter Bootstrap alert class names.

The plugin is registered in the ControllerPluginManager under the key "Notification"

Namespace	Class Constant	Meaning
success	Notification::NAMESPACE_SUCCESS	An action was successfull
warning	Notification::NAMESPACE_WARNING	Action was (partly) successfull
danger	Notification::NAMESPACE_DANGER	Action was not successfull (error)
info	Notification::NAMESPACE_INFO	General info notification w/o special meaning

In a controller action, simply call the plugin via the magic __call mechanism

```
$this->notification()->success('Updates successfully changed.');
return $this->redirect(...);
```

The plugin provides following methods:

- success($message)
- warning($message)
- danger($message)

- error($message) [alias for danger(), for convinience]
- info($message)
- addMessage($message, $namespace = 'info')
- __invoke($message = null, $namespace = 'info')

Rendering

To render notifications it is necessary to render the template which is registered under the key core/notifications in the view manager's template map.

The default view script provided renders all notifications in a div container with the class "yk-notifications" using the "Alert" view helper.

Notifications are rendered in the following order:

- Danger
- Warning
- Success
- Info

You can place notifications into your general layout by following these steps:

1. In the layout script, above the output of the headScript-Helper, render the notifications partial and capture to a variable. (Because the template injects a javascript to the headscript container)
2. Echo the capture variable at the position where the notifications should be.

```
<?php $notifications = $this->partial('core/notifications'); ?>

//...

<?php echo $this->headScript(); ?>

// ...

<?php echo $notifications; ?>
```

Alert View Helper

The alert view helper takes a message and renders it in the bootstrap markup for an dismissable alert box. It is registered in the view helper manager under the key "Alert".

```
<?php // capture content
$this->alert()->start('info'); ?>
<p>This is an info message</p>
<?php echo $this->alert()->end(); ?>

<?php // via __invoke
echo $this->alert('warning', 'This is a warning');

// via shortcut methods
echo $this->alert()->danger('This is an error message.');
```

The helper provides following methods

- __invoke($type = null, $content = null)
- start($type)
- end()
- info($content = true)
- warning($content = true)
- danger($content = true)
- success($content = true)

Passing "true" (or nothing) to a shortcut method is the same as starting capture with the according type.

```
<?php $this->alert()->info() ?>
<p> This is an info message </p>
<?php echo $this->alert()->end() ?>
```

The resulting html will look something like this:

```
<div class="alert alert-info alert-dismissable">
    <button type="button" class="close" data-dismiss="alert">&times;</button>
    <p>This is an info message</p>
</div>
```

Logging

all PHP errors are logged into log/yawik.log. This is configured in:

```
'log' => array(
      'Log/Core/Cam' => array(
          'writers' => array(
               array(
                   'name' => 'stream',
                  'priority' => 1000,
                  'options' => array(
                       'stream' => __DIR__ .'/../../../log/yawik.log',
                  ),
              ),
          ),
      ),
```

Headscripts

To inject script tags (with and without source) to the head section of the rendered output, YAWIK makes use of the Headscript view helper of Zend Framework.

Inject scripts from a view script

To inject a script tag from a view script:

```
<!-- append a file -->
<?php $this->headscript()->appendFile($this->basePath('path/to/script.js')) ?>
```

```
<!-- prepend a file -->
<?php $this->headscript()->appendFile($this->basePath('path/to/script.js')) ?>

<!-- or use another method of Zend Frameworks' helper. -->
```

Inject scripts via module.config.php

It is possible to inject head script tags using the module.config.php.

```
// inside module.config.php

return array(
    //...

    'view_helper_config' => array(
        // ...
        'headscript' => array(
            // append a script for all routes. (ommitting base path, it is added␣
↪automatically)
            'path/to/script.js',

            // append a script for a special route name (or child routes)
            // note: you need to wrap script path in an array due to ZFs' config␣
↪merging.
            'routename' => array('path/to/script.js'),

            // to prepend a script, you need to pass arguments to the headscript␣
↪helper:
            'routename' => array(array(Headscript::FILE, 'path/to/script.js', 'PREPEND
↪')),
        ),
    ),
    // ...

);
```

Note: The scripts from module.config.php are not included, if you use the default Headscript helper in your layout. You need to retrieve the ConfigHeadscript service from the view plugin manager, as its factory injects the scripts.

```
<?php echo $this->configHeadscript() ?>
```

Mails

Mails have two essential agents

- a message-service, which is liable for gathering and providing data and rendering the mail
- a mail-service, which is liable for sending the mail

when creating own mails, you usually extends the message

All related classes are in the Core-Modul, the interaction in in this diagram: http://www.gliffy.com/go/publish/7191865

Using the MailService

Mails can be used everywhere, where have access to the application-serviceLocator.

When you need to send a mail, there are four steps to do

1. call the mail-service
2. get a message-service from the mail-service (there are two distinguished types)
3. feed the message-service with informations
4. use the mail-service to send the message-service

The two types of message-service:

- **Templates, which uses scripts for the body and render them like usual views.** This is more preferred approach for mails with lots of text, and also with mails for different languages
- **Derived classes**, which is preferred when there is a lot of processing is involved.

Using a script as body

When using a script the message-service in some way behaves like a viewmodel, it takes in arbitrary variables, which can be accessed in the script. Also you can set a template, which is resolved by view-maps or view-pathes. In the scripts you can use PHP, and since the script is included into the message-service, you can set or change in the script mail-specific attributes like header oder subject. Scripts are an alike to views.

To use a script you have to instanciate a mail service and a htmltemplate service

```
$mailService              = $serviceManager->get('Core/MailService');
$mail                     = $mailService->get('htmltemplate');
$mail->entity             = $entity;
$mail->link               = $previewLink;
$mail->setTemplate('mail/myScript');
$mail->setSubject( /*translate*/ 'A Title');
$mail->setTo($email);
$mail->setFrom($userEmail, $userName);
$mailService->send($mail);
```

The script is set in the code, so there you can make the choice of the content, by simply choosing a script. But always remember to consign the location of the script in the template-map.

Using an own class

Own classes provide all information by methods. Own classes are the preferred choice when informations are volatile or special (like including pictures or other mimetypes). Look the classes in `Applications\Mail` for example. The own classes must be announced in the config like

```
'mails' => array(
    'invokables' => array(
        'myOwnClass' => 'xxx\Mail\myOwnClass',
```

With being announced, the mail-service can instantiate and initialize this class properly.

```
$mailService = $this->getServiceLocator()->get('Core/MailService');
$mail = $mailService->get('myOwnClass');
$mailService->send($mail);
```

Since most of the own classes are derived from `Zend\Mail\Message` (at least they should be derived from it), they will have a full pledge of all the methods, which are provided especially for mails, like `setEncoding`, `setFrom` etc...

Options

To modify the options, copy the module.core.options.local.php.dist to you `config/autoload` directory, remove the `.dist` prefix and adjust the values

Name	type	description
siteName	string	The siteName is used in Mails. Typically it's the name of your website
operator	array	Contact Data, which can be used in Mail signatures or the imprint page
supportedLanguages	array	supported frontend languages a user can switch to
defaultLanguages	string	default language to use, if no language is set. Default "en"
detectLanguage	bool	if enabled, YAWIK tries to detect the language from browser settings (if no language is set in the users settings)
defaultCurrency	string	default currency to use, if no currency is set. Default "USD"
defaultTaxRate	string	default tax rate to use, if no tax rate is set. Default "19"

supportedLanguages is an assoziative array. The key is used for routing. The value is used as the locale. The upper case part of the locale defines the regions.

```
protected $supportedLanguages = array(
        'de' => 'de_DE',
        'fr' => 'fr',
        'en' => 'en_US',
        'es' => 'es',
        'it' => 'it',
        'el' => 'el_GR'
    );
```

provides core functionality

- Sending Mails
- Pagination
- Error Handling
- Configuration Handling
- PDF Handling
- Attachment handling
- ACL for Attachments
- general Layout

Layout

Note: the following table is generated automatically. Descriptions are marked from the view scripts files mit `{{rtd: ...}}`

Module	Name	Description
Core	layout/layout	General layout. Includes the HTML Header
Core	error/404	File not found error page
Core	error/403	Forbidden error page
Core	error/index	Internal Server Error Page (500)
Core	main-navigation	Renders a horizontal navigation with drop downs
Core	pagination-control	Renders paginations
Core	core/loading-popup	Renders a simple loading box while ajax requests are proceeded
Core	core/notifications	Renders default notification boxes.
Core	form/core/buttons	Renders default 'save' and 'abort' buttons
not found (form/core/privacy)	WRONG CONFIGURATION	
Core	core/form/permissions-fieldset	Renders the group permission fieldset
Core	core/form/permissions-collection	Renders the group form
Core	core/form/container-view	Renders horizontal summary form
Auth	form/auth/contact.form	Renders the contact form within the application form and the per
Auth	form/auth/contact.view	renders the contact information within the application form and t
Auth	auth/form/user-info-container	Renders horizontal form for the contact and the user photo
Auth	auth/form/userselect	Renders form for adding Users to a Group
Auth	auth/form/social-profiles-fieldset	Renders the fieldset for adding Social Profiles to an Application
Auth	auth/form/social-profiles-button	Renders the selection boxes for adding social profiles to an appli
Auth	auth/sidebar/groups-menu	file exists
Applications	applications/error/not-found	Error Page for an Application form which references a non-existi
not found (layout/apply)	WRONG CONFIGURATION	
Applications	applications/sidebar/manage	currently not used
Applications	applications/mail/forward	Renders the email for forwarding an application
Applications	applications/detail/pdf	Renders a application as a simple HTML, used in the PDF gener
Applications	applications/index/disclaimer	Display the privacy policy disclaimer
Jobs	jobs/sidebar/index	Renders paginations
Jobs	jobs/form/list-filter	Renders the search formular for jobs used by recruiters
Jobs	jobs/form/apply-identifier	currently not used. Generates an reference number for jobs
Jobs	jobs-publish-on-yawik	displays short info about publishing on YAWIK
Jobs	jobs-publish-on-jobsintown	displays short info about publishing on Jobsintown.de
Jobs	jobs-publish-on-homepage	displays short info about publishing on the own homepage
Jobs	jobs-terms-and-conditions	display the terms and conditions, when publishing a job opening
Jobs	mail/jobCreatedMail	Mail is sent to the owner of yawik. The mail contains a link to ar
Pdf	pdf/application/details/button	Renders the download as PDF Button, in the Applications Modu
Geo	geo/form/GeoText	Renders the autocompletion for locations
Organizations	organizations/index/edit	Renders the formular for editing organizations
not found (piwik)	WRONG CONFIGURATION	

Mail Templates

Module	Name	Description
Auth	register	sends a confirmation link to the user, after registration
Auth	forgot-password	sends a confirmation link to the user, after using the forgot password feature
Auth	first-external-login	sends the user login data, after the user wa created by an external application
Auth	first-socialmedia-login	sends the user a welcom mail after the first login via a social network

Services

Module	Name	Description
Core	Core/Log	Logging service
Auth	HybridAuthAdapter	Login via Social Networks
Auth	AuthenticationService	Authentication Service

Events

Name	Description
core.create_paginator	is fired, when CreatePaginator plugins creating a paginator
job.created	is fired, when a user created a job opening.
job.accepted	is fired, when an admin accepts a new or modifications on an existing job opening

Notifications

Every notification or message, no matter how it will be displayed or returned, runs through an unified API. This API is implemented in the Controller-Plugin 'notification'. Notifications are session-persistent, that implies, they will pop up either on the current site, or on a following site. So unless you are sure of it, make no references to a current page, because the notification may pop up on a different page.

The common use is:

```
$this->notification('any text');
$this->notification()->success('any text');
$this->notification()->error('any text');
$this->notification()->info('any text');
```

To display notifications on a html-page, insert somewhere in the script or layout. In the standard-layout this partial is already included.

```
echo $this->partial('core/notifications');
```

If you have an ajax-request and expect back a JSON, the JSON-response should include information about notifications. You have to trigger an event with the whole response as data.

```
$.post(url, param, function(data) {
    $(this).trigger('ajax.ready', {'data': data});
})
```

Language Switcher

you can add a Language Switcher into you skin by:

```
<?=$this->languageSwitcher()?>
```

If you want to modify the Layout, edit the view script language-switcher.phtml

Auth

the auth module is based on hybridauth. The social networks Facebook, Xing and LinkedIn are ready to connect, just by configuring their API key and secret. Other Networks can be easily added.

User Data are stored in the `users` collection.

The Auth module offers the following features

- Register with Facebook, Xing, LinkedIn, Google or GitHub
- Register via registration form
- I forgot my Password
- Roles for applicants, recruiters and admin

By using the optional Module YawikCompanyRegistration, users can register as a company. The module provides a formular and creates a user and a company in one step.

Mails

template	purpose	triggered from
mail/register	contains a confirmation-link to ensure the email-address. Without this assurance the account will not be fully activated	
mail/forgot-password	Mail containing a link which enables the user to reset the password	
mail/first-socialmedia-login	contains username and password. Mail is sent to the user after the first social media login	
mauk/first-external-login	contains username and password. Mail is sent, after a user was created by an external application	

CV (Curriculum vitae)

The CV module offers the possibility to store CVs. A CV consists of a contact, information about the preferred jobs and job location of a candidate and a collection of education histories, work experiences, personal skills and attachments.

The following workflow can be offered.

- Job ist posted
- Candidate applies
- hiring organization must reject the applicant.
- The applicant is asked, if he agree to be added to a talent pool.
- If the applicant agrees, his application is copied to the CV module. He gets login data to the YAWIK installation
- If the applicant disagrees, the application is deleted.

In addition thr following features will be offered

- Recruiter can add multiple CVs.
- Recuiter/Applicant can import CVs from Europass

- Applicant can import CV from Social Network

Applications

the application module offers an application formular, a list of applications and a detail view of an application. Depending on the users privileges the detail view offers a way to invite or reject an applicant, to rate an application or to forward an application by email.

If the *PDF* Module is installed, the Application can be downloaded as a PDF document with attachments and social profiles embedded.

Options

You can configure the possible mime-types or the maximum size of attachments by copying the applicationOptions into your config/autoload directory. Remove the ".dist" extention and adjust the values.

```
$options = array(
      /*
      * maximum size in bytes of an uploaded attachment, default 5MB
      */
      'attachmentsMaxSize' => '5000000',

      /*
      * allowed Mime-Type of an uploaded Attachment, default images, *.PDF, *.DOC, *.
↪ODT
      */
      'attachmentsMimeType' => array('image','application/pdf', 'application/vnd.
↪oasis.opendocument.text', 'application/msword'),


      /*
      * maximum amount of uploaded attachments
      */
      'attachmentsCount' => 5,

      /*
      * maximum size in bytes of an uploaded contact photo. default 500kB
      */
      'contactImageMaxSize' => '500000',

      /*
      * allowed Mime-Type of an uploaded contact photo
      */
      'contactImageMimeType' => array('image'),

      /*
      * allowed Mime-Types, images, plain text, *.PDF, *.DOC, *.ODT
      */
      'allowedMimeTypes' => array('image','text','application/pdf', 'application/vnd.
↪oasis.opendocument.text', 'application/msword'),

  );
```

AbstractIdentifiableModificationDateAwareEntity

Events

you can attach Listeners to the following events

Name		description
EVENT_APPLICATION_POST_CREATE	application.post.create	Thrown, after an application was saved in the Database
EVENT_APPLICATION_PRE_DELETE	application.pre.delete	Thrown, befor an application is removed from the Database
EVENT_APPLICATION_STATUS_CHANGE	application.status.change	Thrown, befor an application is removed from the Database

Workflow

Organizations

The organizations module adds a storage for organizations. If this module is enabled, a user can create an organization. He can invite emploees via email to his organization. Employees can have the following roles

Role	Description
organization admin	Owner of an organization
recruiter	Default Role of an employee
department manager	Department managers can accept or reject applications
manager	currently unused

In addition the following permissions can be set

Permissions	Description
create jobs	User can create job postings
edit jobs	User can view and edit job postings
view jobs	User can view job postings
edit applications	User can update applications. Eg. rate, invite, reject, etc
view applications	User can view the application including attachments etc.

Applicants refer to Organization Names in their work history. Job Postings require an Organization Name. Either the name of the hiring Organization or the name of an agency. A Recruiter has to assign himself to an Organization.

Organization Names are just Names. They are public. Organization Names are assigned to various ratings. If an Organization Name is used as an hiring Organization for a job posting, or if a recruiter is using an Organization name for it's own company, the ranking is modified.

An Organization entity itself only contains a reference to an organization Name.

CompanyRegistration

CompanyRegistration

`Repository`	yawik/CompanyRegistration
`coverage`	
`buid`	

If you want to offer the registration for companies, this module might be helpful. It offers a registration form with additional fields. When a user registers, an user and an organization entity are created. This module requires the "Oganizations" Module.

Installation

to install the yawik/CompanyRegistration Module into a running YAWIK, change into the *YAWIK/modules* directory and clone the yawik/CompanyRegistration repository.

```
git clone https://github.com/yawik/CompanyRegistration
```

To activate the module create a php file named `WhateverYouWant.module.php` in your config autoload directory containing:

```
<?php
return ['CompanyRegistration'];
```

Another possibility to install YAWIK modules is using composer.

```
composer create-project cross-solution/yawik
cd yawik
composer require cross-solution/yawik-company-registration
```

This install the CompanyModule into the *module* directory of your YAWIK installation. You can uninstall the module via

```
composer remove cross-solution/yawik-company-registration
```

This removes the directory *CompanyRegistration* and all it content from your *module* directory of your YAWIK installation.

Configuration

The registration form contains by default the fields:

- gender
- name
- email
- organizationName
- postalCode
- city
- street
- houseNumber
- phone

You can configure the registration form. Copy the RegistrationFormOptions.config.local.php.dist into your *autoload* directory and adjust the values.

Orders

Orders

Repository	yawik/orders
coverage	
buid	

Requirements

a running YAWIK

Installation

to install the yawik/orders Module into a running YAWIK, change into the *YAWIK/modules* directory and clone the yawik/orders module.

```
git clone https://github.com/yawik/Orders
```

To activate the module create a php file named `WhateverYouWant.module.php` in your config autoload directory containing:

```
<?php
return ['Orders'];
```

Description

the oder module injects a billing address to to wizard for entering job postings. the order module adds a storage for orders. By submitting a job posting, an order is created. The order contains all relevant data needed for billings. In addition, the module adds an invoice formular, which can be added into the order process. Default values of the invoice formular can be set in Settings/Orders.

Technically, the oder module offers the feature to take a snapshot of an entity.

Jobs

The Jobs module allows to enter and manage job ads. In addition it generates a list of jobs. List of Jobs can be generated in a recruiter (*Recruiter Mode*) and a public search (*Public Search Mode*) mode

The entering process is defined at: http://www.gliffy.com/go/publish/6254781

in the recruiter mode the

Fig. 5.3: Recruiter Mode

recruiter can see active and inactive jobs. In addition the list contains informations like number of applications (total/new) or the recruiters name, who is responsible for the position.

in search mode the users only see published jobs. This is normally used as a list of current vacancies, which is often used on a corporate website.

Fig. 5.4: Public Search Mode

The list mode is defined by the users role.

It is also possible to configure YAWIK to run as a jobboard. There is a jobboard module which lets YAWIK act like a jobboard. This module is running on

http://jobs.yawik.org

Job Templates

you can create templates for entering job ads. All you need is an HTML version of your job opening. Simply replace the *requirements*, *qualifications* or *benefits* with a small piece of code. E.g.

```
<h4>Requirements:</h4>
<?php echo $this->requirements;?>
```

YAWIK replaces this code with an inline Wysiwyg HTML Editor if you want to modify your job opening. Otherwise the code is replaced by the HTML code, which was entered.

Modifications to the label fields `labelBenefits`, `labelQualifications` and `labelRequirements` are applied to all jobs of company, which are using the template.

You currently can use the following placeholders:

Name	Description
$this->benefits	Employee benefits
$this->city	City of the company
$this->description	description of the company
$this->descriptionEditable	editable description of the company
$this->jobId	ID of the job posting (since 0.29)
$this->qualifications	Needed qualifications
$this->location	Location of the job
$this->labelBenefits	Label of the Benefits Section
$this->labelQualifications	Label of the Qualifications Section
$this->labelRequirements	Label of the Requirements Section
$this->oraganizationName	Name of the company
$this->postalCode	postalCode of the company
$this->requirements	requirements of the job posting
$this->street	Street of the company
$this->title	editable title of the job posting
$this->headTitle	title of the job posting
$this->uriApply	URL a an application form
$this->uriJob	URL a the job posting
$this->uriLogo	URL of a company logo
$this->jobApplyButtons($this->applyData)	Apply Button
$this->jobApplyButtons($this->applyData)	Apply Button

Yawik comes with the example templates "default", "modern" and "classic". If you want to change the Templates within your Module, you can overwrite the template mapping adding the following configuration to your module config. Eg. you can put a file `templates.config.php` into your `MyModule/config` directory.

```
<?php
return [
   'view_manager' => [
       'template_map' => [
           'templates/default/index' => __DIR__ . '/../view/yourTemplate1/index.phtml
↪',
           'templates/modern/index' => __DIR__ . '/../view/yourTemplate2/index.phtml',
           'templates/classic/index' => __DIR__ . '/../view/yourTemplate3/index.phtml
↪',
       ]
   ]
];
```

If you want to modify the selection of the templates (iframe) add the following mapping

```
'iframe/iFrame.phtml' => __DIR__ . '/../view/YourTemplateSelection.phtml',
```

Mails

you can translate mails by adding the language to the template name. example: https://github.com/cross-solution/YAWIK/tree/develop/module/Auth/view/mail

Name	Description
mail/job-created	mail is sent to th approval team
mail/job-pending	mail is sent to the person, who created the job.
mail/job-accepted	mail informs the person, who created the job, that the job is going to be published
mail/job-rejected	mail informs the person, who created the job, that the job was rejected

Options

To modify the options, copy the module.jobs.options.local.php.dist to you `config/autoload` directory, remove the `.dist` prefix and adjust the values

Name	type	description
multipostingApprovalMail	string	recipient email of the approval team
multipostingTargetUri	string	Send a Rest Request to this target on status changes of a job opening. The URI can contain username/password. eg: `http://user:pass@host/location?query`
defaultLogo	string	The default Logo is shown in a job opening and in the application form
companyLogoMaxSize	int	Maximum size in bytes of a company Logo. Default 200kB
companyLogoMimeType	array	Allowed Mime-Types for company Logos

Channel Options

The Channel Options contain information about publishing channels, a user can select to publish a job posting. To modify the options, copy the channel.options.local.php.dist to you `config/autoload` directory, remove the `.dist` prefix and adjust the values

Name	type	description
externalKey	string	external key of a channel. Eg. a provider offers the channel "MyJobboard" with the key "123". YAWIK provides a channel "MyJobboard" using the key "myJobborad". Set externalKey to "123", if the job is published to the provider.
prices	array	[base,list,min] You can define 3 prices which you can use in your *price-calculation*
currency	string	currency of the price. Default: CoreOptions::defaultCurrency
tax	int	tax rate of the channel. Default: CoreOptions::defaultTaxRate
label	string	label of the channel
publishDuration	int	number of days a job opening can be published
category	string	Category of the channel. Default: "General"
headline	string	Headline of the channel
description	string	Description of the channel
linktext	string	Linktext of a link to further information of the channel
linkTarget	string	Link target of a link to further information of the channel
route	string	Route to a content page with details about the channel
params	array	Parameter, which can be used for linking the detail page about the channel

ATS Mode

The ATS (Applicant Tracking System) Mode defines, how applications should be processed. The following modes exist:

Name	description
intern	Applications are stored within the local YAWIK instance
uri	Application Form is pointed to en external ATS System
email	Application Form is forwarded via Email
none	The Application Formular is deactivated

By using the ATS Mote `intern`, you can enable the One-Click-Apply Feature. This will add an additional Apply Button per selected social network into the job opening.

Widget

by using the folloging Javascript Widget you can add your jobs into your personal homepage.

```
<script>
   (function (window, document) {
       var loader = function () {
           var script = document.createElement("script"), tag = document.
↪getElementsByTagName("script")[0];
           script.src = "view-source:https://yawik.org/YawikWidget/yawik.min.js";
           tag.parentNode.insertBefore(script, tag);
       };
       window.addEventListener ? window.addEventListener("load", loader, false) :␣
↪window.attachEvent("onload", loader);
   })(window, document);
</script>
```

The javascript renders a joblist inside a container with the id `YawikWidget`

```
<div id="YawikWidget"
    data-organization="55ae775c6b10f8f05b8b457f"
    data-yawik="https://yawik.org/">
</div>
```

The attribute data-organizations takes an organization id, provided by your used yawik. The attribute data-yawik takes the location of the used yawik.

Source Code of the Widget: https://github.com/cbleek/YawikWidget

Price Calculation

The price calculations can be overridden by creating a MyCalculation.php. You can start by coping the ChannelPrices.php to MyCalculation.php. Adjust the namespace and implement your logic within the filter function.

To use your MyCalculation.php, you have to copy the ChannelPricesFactory.php into YourModule. Adjust the namespace and the $filterClass value.

To use your filter, you have to put the following config into your modules.config.php

```
'filters' => [
  'factories'=> [
     'Jobs/ChannelPrices'  => 'YourModule\Factory\Filter\MyCalculation',
     ...
    ]
 ]
```

One-Click-Apply

Since 0.25

You can simply add an apply button to you job opening by putting the following code into your job template.

```
<?=$this->jobApplyButtons($this->applyData)?>
```

This will use the ATS Mode settings and render the button. In addition to the ATS Mode settings you can set options to the Apply buttons. These options can be used to modify the layout and the behaviour of the Apply button.

```
<?=$this->jobApplyButtons(
    $this->applyData,
    [
        'sendImmediately' => true,
        'oneClickOnly' => false,
        'defaultLabel' => 'Click here to apply',
        'oneClickLabel' => null
    ]
)?>
```

name	value	description
sendImmediately	bool	true = Application is send immediatly. Privacy policy are accepted by clicking on the button
oneClickOnly	bool	true = normal button, which refers to the form is hidden
defaultLabel	string or NULL	label of the normal button.
oneClickLabel	string or NULL	label of the OneClickApply Button

By modifying the labels, you normally loose the translations. Feature was sponsored by http://stellenmarkt.de

XML Feeds

Since 0.28 each job channel can be exported as an XML Feed. A lot of existing jobboards does not provide an API to publish jobs. Job publishing is often done via XML Feeds. Therefore a default XML Structure is provided, which can be imported by external jobboards. The XML Structure can be easily modified and my vary between different jobboards.

Example of the default XML:

https://yawik.org/demo/en/export/xml

Search

the search formular contains the following fields

name	Description
q	fulltext search
l	location
d	distance
c	category

search formulars can be prefilled by using the formular field names. Example: https://yawik.org/demo/de/jobboard?d=20&l=Frankfurt+am+Main&q=bla

Will prefill the fullext field with “Bla”, The distance field with “20” and the location with “Frankfurt am Main”

The formular field for the professions has the name "c". You can use the name with a separator prefix. Example https://yawik.org/demo/de/jobboard?:c=it:sales will prefill categories with "it" and "sales".

PDF

the PDF modules enables to download an application as an PDF document

Geo

The Core module provides a form-field of the type "Location". This form-field should be used whenever a Location is entered. In the Core the type "location" is an alias to a standard form field type "text". So if the Geo module is inactive, a normal "text" field is used and nothing else happens.

But the Geo Module does a little bit more than just autocomplete the location. It uses a geo location service to enrich the entered data with geo coordinates and informations about the country and the region. If you enter "Frankfurt am Main", the location will be defined as:

```
city = Frankfurt am Main
region = Hessen
Country = Germany
corrdinates = [8.6820934,50.1106529], type:"Point"
```

This makes it possible to use the distance feature, when searching e.g. for jobs. The Geo module currently ca use two different geo location services.

1. the photon service
2. the geo service

What's the differences between those services.

Feature	photon	geo
multilingual	yes	no (only german is supported)
countries	worldwide	DE, AT, CH
search for postal codes	no	yes
Ranking	nearest by	population
needed requests	1	2
Ranking	nearest by	population
synchronized with OSM	yes	no
search for streets	yes	no
sources available	yes	no
free service available	http://photon.yawik.org/api	http://api.cross-solution.de

The Geo module can be easily configured to use one of the geo services by copying and modifying the Geo/config/Geo.options.local.php to the autoload directory of you YAWIK installation

```
<?php
/**
 * Name of the used geo coder plugin. You can use 'photon' or 'geo'. Photon is␣
↪recommended.
 */
$plugin = 'photon';

/**
```

```
 * Location of your geo coder server. If unsure, leave it unchanged. Possible values␣
↪are:
 * - http://photon.yawik.org/api
 * - http://api.cross-solution.de/geo
 */
$geoCoderUrl = 'http://photon.yawik.org/api';

//
// Do not change below this line!
//

return [
    'options' => [
        'Geo/Options' => [
            'options' => [
                'plugin' => $plugin,
                'geoCoderUrl' => $geoCoderUrl,
            ],
        ],
    ],
];
```

It it possible to configure

Solr

Solr

Repository	yawik/solr
coverage	
buid	

Requirements

current development is using:

- php5-solr (pecl >= 2.4.0)
- apache solr (6.1)

Note: Debian 8 ships with php5-solr 1.0.2. You can build your solr extension by:

```
aptitude install php5-dev libcurl4-openssl-dev libxml2-dev
pecl install solr
echo "extension=solr.so" > /etc/php5/mods-available/solr.ini
php5enmod solr
php -m| grep solr # should show the activated solr extension
```

Good resources on how to install solr:

- https://cwiki.apache.org/confluence/display/solr/Installing+Solr

- http://nl3.php.net/manual/en/solr.installation.php

Installation

to install the yawik/solr Modul into a running YAWIK, change into the *YAWIK/modules* directory and clone the yawik/solr module.

```
git clone https://github.com/yawik/Solr
```

To activate the module create a php file named `WhateverYouWant.module.php` in your config autoload directory containing:

```
<?php
return ['Solr'];
```

To configure the solr connection copy the Solr options file into you autoload directory and adjust the values.

```
cp module/Solr/config/solr.moduleoptions.local.php.dist config/autoload/solr.
↪moduleoptions.local.php
```

Note: Solr needs a schema. The schema is currently a work in progress. You can use the schema in Solr/contrib.

you can initially index all active jobs by:

```
bin/console solr index job
```

Description

YAWIK entities are searchable using the fulltext feature offered by mongodb. These features are great and normally sufficient, to offer e.g. jobs on a career page. If you want to use YAWIK as a jobboard, the requirements increase. A jobboard normally offers millions of jobs to millions of visitors. At first you need a scaling search engine. Currently Solr is supported.

Using the solr module, you'll get full featured search engine offering the following features:

- facet searches (e.g. list of categories showing possible matches)
- highlight matches in the search result

YawikXingVendorApi

Requirements

You'll need a Xing Account for publishing jobs. More infos: https://www.helpify.de/xing-posting-api-en/2937/how-can-i-as-a-developer-use-the-xing-posting-api

Installation

```
composer create-project cross-solution/yawik
cd yawik
composer require cross-solution/yawik-xing-vendor-api
```

This install the YawikXingVendorApi module into the *module* directory of your YAWIK installation. You can uninstall the module via

```
composer remove cross-solution/yawik-xing-vendor-api
```

This removes the directory *YawikXingVendorApi* and all it content from your *module* directory of your YAWIK installation.

Settings

The settings module takes settings of other modules.

Eg. The order modules adds the possibility to configure an invoice address. This is simply done by defining a Settings Entity and a Settings Fieldset. The Settings module ensures that forms are rendered, values are stored and the navigation is extended by the corresponding sections.

Eg:

https://github.com/cross-solution/YAWIK/blob/develop/module/Orders/src/Entity/SettingsContainer.php

https://github.com/cross-solution/YAWIK/blob/develop/module/Orders/src/Form/InvoiceAddressSettingsFieldset.php

JobsByMail (Work in Progress)

JobsByMail

Repository	yawik/JobsByMail
coverage	
buid	

The JobsByMail module offers a simple Formular to sign up to get the latest jobs by email. By activating the module you'll be able to the Form into the search result <?= *$this*->

Features

- view script for the formular and a result page.
- form ist pre-filled with the latest search parameters
- form can be used as an authenticated and as an anonymous user

If the formular is used by an anonymous users, a confirmation mail is send to the subscriber. Search profiles width confirmed email adresses will receive new jobs my mail.

The Infomation Mail about new Jobs contains an unsubscribe Link.

Installation

to install the yawik/JobsByMail Modul into a running YAWIK, change into the *YAWIK/modules* directory and clone the yawik/solr module.

```
git clone https://github.com/yawik/JobsByMail
```

To activate the module create a php file named `WhateverYouWant.module.php` in your config autoload directory containing:

```
<?php
return ['JobsByMail'];
```

Usage

CHAPTER 6

Guidelines

Programming Guidelines related to code-maintenance

- **the content of the output is completely determined by the Controllers**
  - Viewscripts do provide output, but they can be replaced by other viewscripts in the Controller, so the Controller is still in charge
  - a majority of viewhelper also do provide an output, but viewhelpers are still only active on demand.
  - a **nogo** are ubiquitous listener which are defined somewhere in the bootstrap-process and alter or extend the content, there is no point in trying to make them smart, this is just another cause for irratic behaviour
  - it is ok to offer some defaults for output, as long as these defaults can be altered in the Controller
- **avoid distributed addressing by name-strings, it is just awkward to search for errors related to run-time definitions hidden**
  - keep the defining of the behaviour for an entity in a small scope and use abstract handling of the behaviour anywhere else
  - throw exceptions if elements are addressed, which don't exist. Don't rely on purpose if something is missing - if something is optional, flag it as optional
- apply the rule of three
- prefer interfaces to constants for distinguishing code behaviour - interfaces may give you a runtime error if you have done something wrong
- allways use getter und setter, it is not just a principle of object-orientation (hiding), it is also easier to track down a call when debugging
- don't use exceptions as a regular programming flow, exceptions are exceptions - that implies something went wrong in the code, not by the use of the code

Common JavaScript Trigger

Trigger are used to broadcast certain events

ajax.ready Is triggered on the container, which is altered by the ajax.request. Remember, this event bubbles, so all listener on elements above will spring into action, too.

Naming Conventions

We are following the Zend Framework Coding Standard for PHP

- Variables: lowerCaseStartingCamelCase like ....
- Modules and Classes: UpperCaseStartingCamelCase
- Array keys (options arrays): underscore_separated ('option_key')
- Service names Module/[SubCategory/]Service ok.
- Configuration Keys invocable form element: UpperCaseStartingCamelCase, <Module>/<Element> like 'Applications/Mail'
- Configuration Keys invocable controllers: UpperCaseStartingCamelCase, <Module>/<Element> like 'Applications/Mail'
- Configuration Keys view scripts: lowercase, dash-separated, like 'applications/index/disclaimer'

CHAPTER 7

Customize

You can activate/deactivate Modules in config/config.php

```
$modules = array(
      'DoctrineModule',
      'DoctrineMongoODMModule',
      'Core',
      'Auth',
      'Cv',
      'Applications',
      'Jobs',
      'Settings',
      'Pdf',
 );
```

If you want to customize the layout, you can do so by writing a plugin. The easiest way is to clone the YawikDemoSkin into your `modules` directory.

```
cd modules
git clone https://github.com/cbleek/YawikDemoSkin
```

To activate the plugin you can either simply add `'YawikDemoSkin'` to your modules array in `config/config.php`, or if you don't want to touch any code from git at all, simply put a file named eg. `config/autoload/MyModule.module.php` in your autoload directory. Files named *.module.* are read to include additional Modules.

```
<?php
return array("YawikDemoSkin");
```

This will add the module dynamically.

If modules contain data like images, javasript or css, which should be directly accessable by the Webserver, these data should be placed into a directory named `public`. To make this directory accessible to the Webserver place a symbolic link into the `YAWIK/public` directory, pointing to the modules public directory.

```
cd YAWIK/public
ln -s ../modules/YawikDemoSkin/public YawikDemoSkin
```

It is a good practice to name the link with the modules name. This way, you can reference objects of the module by using the ModulesName within the URL.

Example: The YawikDemoSkin references its css in the layout.phtml

```
$this->headLink()->prependStylesheet($this->basePath() . '/YawikDemoSkin/
↪YawikDemoSkin.css');
```

Next thing you propably want is to change the name of the Module. Search and replace all "YawikDemoSkin" with "MyModule" in the sources and rename the Directory "YawikDemoSkin" into "MyModule". Do not forget to change the name in your "autoload/MyModule.module.php" file.

Now you have a module which you can use as a starting point for modifications.

customize your Skin by mapping more *templates* to your own views scripts.

If you want a completely own customized startpage, add a 'startpage' to your viewmap. It will be automatically picked, when you enter the name of the domain and have no session. But be aware, there is no login-box, unless you integrate it yourself.

CSS

YAWIK comes with bootstrap. Glyphicons are replaced by awesome fonts. The sources for for the main CSS is currently build with lessc. Bootstrap and awesome font sources are symlinked to the `vendor` directory'. The global CSS file is build with make-css.sh

you can install lessc on ubuntu by

```
sudo apt-get install npm
sudo npm install -g less
```

Our YawikDemoSkin can be seen as an example, how to modify the CSS. The Skin needs a different height for the fixed footer. This is achieved by creating a new less file, which can import our `yawik/yawik.less` (a symlink is pointing to it). You can overwrite all less variables.

```
@import "yawik/yawik.less";
@footer-height:                    39px;
```

Your customized CSS can be compiled with lessc like:

```
lessc YawikDemoSkin.less ../public/YawikDemoSkin.css
```

Formular Fields

Name	description
Editor	Editor element
FileUpload	FileUpload Form element
InfoCheckbox	InfoCheckbox Form element. Adds a Link like to the description Text.
Location	autocomplete a location and adds additional Geo data, see: *Once Click Apply*
Phone	adds Validation for a phone number
Rating	Star rating Element
SpinnerSubmit	a spinner icon is added during form validation. While sending data, the submit button is inactivated

View Helper Scripts

Name	description
Alert	displays notification like error or success
Services	can access Services within view view scripts
jobUrl	displays the link to a job posting.
applyUrl	displays the link to an application form of a job posting.
applyButton	displays application buttons. see: *Geo Module*
languageSwitcher	renders a language switcher select box. see: *Language Switcher*

CHAPTER 8

API

YAWIK currently offers a simple API to create Jobs. If you are able to create HTML formatted job ads and if you can put application links to the HTML source of your jobs, you might find the API usefull to put your jobs into YAWIK, which enables you the use the application forms.

We'll later replace this simple API with a full featured. Maybe build with https://apigility.org/.

YAWIK allows an external application to post jobs via http POST requests. At first you have to authenticate. This can be done by:

Example:

```
curl -c "/tmp/cookie" -d "appKey=SecretYawikDemoKey&user=demo&pass=demo" http://yawik.
↪org/demo/login/extern?format=json
```

The following parameters can be passed:

Param	Value	Description
appKey	string	pre-shared key between your app and your YAWIK
user	string	user name of an YAWIK Account
pass	string	password of an YAWIK Account
email	email adress	If an YAWIK Account is created, the password is sent to this email address
role	recruiter or user	If an YAWIK Account is created, this role is used

The appKey authenticates your external application. The to "top secret" pre-shared key of the YAWIK demo is `SecretYawikDemoKey`. Normally this key is only known by YAWIK and the external App.

Success:

```
{"status":"success","token":"em7ke40ec5sskqce5jgtt912m5"}
```

Failure:

```
{"status":"failure","code":0,"messages":["Invalid application key"]}
```

Once you've logged in, the cookie returned by YAWIK has to be stored on the external application site. In case you're testing it with the above curl statement, the cookie is stored in /tmp/cookie.

If the application Key is valid and the user is unknown, a user is simply created. The password of the user is sent via email (if an email-value is provided in the curl-call). Authentification via CURL and the normal login are traded different. Every authentification with an external application uses a separate key, therefore the external application is liable to provide privacy. An user can root out any external application by simply revoking it's key without affecting any other authentification.

These parameters are available or must be set to transmit a job:

Param	Value	example	mandatory	Description
applyId	string	AMS79j	yes	the id is an unique key to adress your job
company	string		yes	name of the company
companyId	string		no	if an id is provided, the company is stored in the YAWIK-DB
contactEmail	email adress		no	for automatic informations like new applicants
title	string		yes	for tabular overview
location	string		no	for overview, will be later prone for indexing
link	http adress		yes	the job offer weblink
datePublishStart	YYYY-MM-DD		yes	
status	string		no	Possible values
reference	string		no	an internal reference from the publisher
logoRef	http adress		no	Logo for the Company
uriPublisher	http adress		no	who get the credit for any application
atsEnabled	boolean		no	set to true shows up a link to an application form
uriApply	http adress		no	adress to an external application form

some remarks:

applyId The applyId must be unique just to the provider, this key along with your authentification is the only access to your data. Consistently there is no key provided by YAWIK.

atsEnabled enables the Applicant Tracking System for the job opening.

company, companyId companies can managed alongside the job if a companyId is passed, the companyId is an assurance for yawik, that different jobs with the same companyId belong to the same company. The name is for that a to weak criteria.

contactEmail Although a contact-email is not obligatory, it is a crucial enhancement of service. Whenever something happens to your job, you get an update. This includes new applicants for a job.

link This link should be an appealing presentation of the job. YAWIK can not (up to now) display Jobs on it own, so this link is mandatory.

uriPublisher One of the basic ideas of YAWIK is to distribute jobs automatically. Even though, every job may have an owner who wants to administer the job. The Adress of uriPublisher must provide and own rest-service for updates or feedback of informations.

uriApply As joboffers can be distributed, the application could directly linked back to the source. The distributing system can add a signature to the application to indicate where the applicant has first seen the job offer.

Current States of job openings are defined in Jobs/Entity/StatusInterface

Status	Description
CREATED	job opening was created
WAITING_FOR_APPROVAL	entering of a job opening was finished
REJECTED	the job was rejected
PUBLISH	job was accepted an is going to be published
ACTIVE	the job is online
INACTIVE	job should go offline
EXPIRED	the job is expired

```
curl -b /tmp/cookie -d "applyId=1234" 'http://yawik.org/demo/de/saveJob?format=json'
{
    "token":"903rgbrs1j6p5gb2586tdci833",
    "isSaved":false,
    "post":{"applyId":"1234"},
    "valid Error":
        {
        "job":
            {
                "company":{"isEmpty":"Es wird ein Eingabewert ben\u00f6tigt. Dieser␣
↪darf nicht leer sein"},
                "title":{"isEmpty":"Es wird ein Eingabewert ben\u00f6tigt. Dieser␣
↪darf nicht leer sein"},
                "link":{"isEmpty":"Es wird ein Eingabewert ben\u00f6tigt. Dieser darf␣
↪nicht leer sein"},
                "datePublishStart":{"isEmpty":"Es wird ein Eingabewert ben\u00f6tigt.␣
↪Dieser darf nicht leer sein"
            }
        }
    }
}
```

A successfull request returns:

```
curl -b /tmp/cookie -d "applyId=1234&title=this%20is%20a%20test%20job&
↪company=MyCompany&datePublishStart=2014-09-15&link=http://example.com/myjob.html" \
       'http://yawik.org/demo/de/saveJob?format=json'
{
    "token":"903rgbrs1j6p5gb2586tdci833",
    "isSaved":true,
    "post":{
        "applyId":"1234",
        "title":"this is a test job",
        "company":"MyCompany",
        "datePublishStart":"2014-09-15",
        "link":"http:\/\/example.com\/myjob.html"
    }
}
```

Transfering Jobs

Jobs are transferred as complete JSON-Objects, there are no RESTful API HTTP methods like `get`, `put`, `post`, `delete`. That's why this is considered a pure

```
curl name:password@server/report/job
    -H 'Accept: application/json'
```

```
    -H 'Content-Type: application/json'
    -d '{
        "applyId": "ref_from_yawik_123",
        "title": "lorem ipsum title",
        "description": "lorem ipsum body",
        "company": "tcomp",
        "contactEmail": "weitz@cross-solution",
        "location": "35510 Butzbach",
        "link": ".",
        "datePublishStart": "2013-08-20T08:19:12.000Z",
        "status": "active"
    }'
```

applyId is a textfield is always the ID of the sending system, it is within the duty of the receiving system to classify the ID to the system, it is recommended to use information of the send-header, like referer-host. The apply-id is mandatory.

title is a textfield title is for a fast human readable classification. The title is mandatory.

description is (propably) a textfield is all information displayed on the target. The content is not specified on purpose, so it's up to the user how to define the description. Though it is recommended to use HTML or at least XML. The description is optional, especially if you use an external link to the job offer.

company is a textfield a company-name is mandatory, for just the reason it is mandatory in every job-offer in the system. It should provide an allocation for accounting

contactEmail is a textfield in case of question or feedback although it is not settled, if this concerns just the content of job or the provider of jobs, this is mandatory

location is a textfield about the location of the job-offer, Since a lot of jobs have no specific location, this is optional.

link is a textfield, when the job-offer should redirect or link to an external page. This is always recommended for high glossy jobs.offers. But - buyers aware - you have to be cognizant that external linking can be disabled for some reason. Anyway, this is optional.

datePublishStart is a textfield in the format YYYY-MM-DDTHH:II:SS.FRACZ (Uni-Format), It can be easely interpreted with PHP and is the time-format in MONGO. This field is just to facilitate the process in transferring jobs in advance. Nonetheless it is a matter of personal agreements for ending a job-offer. As we can not ensure a notice of a premature ending of a job-offer this is not stringent. Therefore this is optional.

status is a textfield this is for putting a job on passiv by external command, or switch it back to active. Default is always active. This field is optional.

CHAPTER 9

Frequently Asked Questions

FAQ: Mails

Is it possible to configure an SMTP Server

Yes. Copy the `Core/config/MailServiceOptions.config.local.php.dist` into you autoload directory and adjects the values.

How can I translate Mails?

translating mails with gettext is possible, but you will propably like to translate mails a little different. You can code the language into the name of the view script.

example:

your're creating a mail using the viewscript `MyModule/view/myMail/mailtext.phtml`. If you wannt to translate this mail into the french lanuage, you simple copy it to `MyModule/view/myMail/mailtext.fr.phtml`.

FAQ: Navigation

How can I add a Link to the Navigation?

Place the following configuration into your `autoload/my-navigation.local.php`

```
<?php
return [
    'navigation' => [
        'default' => [
            'My-Identifier' => [
                'label' => 'Bewerbungsverwaltung',
```

```
                'route' => 'lang/applications',
                'order' => 100,
                'pages' => [
                    'Identifier-Page-1' => [
                        'label' => 'Page 1',
                        'route' => 'lang/applications',
                    ],
                    'Identifier-Page-2' => [
                        'label' => 'Page 2',
                        'route' => 'lang/applications',
                    ],
                ],
            ],
        ],
    ],
];
```

How can I remove Mait templates from the settings menu?

Modules can implement SettingEntities. If they do so, they will be automatically inserted into the navigation. If you want to disable this feature, you can unset Modules Settings in your configuration. Place the following configuration into your `autoload/my-navigation.local.php`

```
<?php
return [
    'Applications' => [
        'settings' => null,
    ],
    'Core' => [
        'settings' => null,
    ]
];
```

How can I add a Link only for Recruiters?

Place the following configuration into your `autoload/my-navigation.local.php`

```
<?php
return [
    'acl' => [
        'rules' => [
            'recruiter' => [
                'deny' => [
                    'route/lang/applications',
                ],
            ],
        ],
    ]
]
```

How can I hide the navigation on the application form?

In the YawikDemoSkin Module, you can see how this can be done.

https://github.com/cbleek/YawikDemoSkin/blob/master/Module.php#L65

FAQ: Translation

How can I translate my module?

we use gettext for translation as default. Gettext by default scans the sources for `translate()` and `setLabel()` function calls. In addition, we've defined, that strings following the annotation `/*@translate*/` should be translated as well. This is done by the little script translate. It scans .php and .phtml files for `/*@translate*/` annotations and puts all following strings into the `module/MyModule/language/_annotated_trings.php` file.

Note: This mechanism has the limitation, that the string which follows the annotation must be in one line.

Example

```
/*@translate*/ 'this will be found by gettext'

/*@translate*/
'this will not be found by gettext'
```

executing `bin/translate module/MyModule` will scan all .php and .phtml files for texts to translate. This will create .po file in the `module/MyModule/language` directory, which you may translate with poedit

User Feedback:

- "bin/translate module/MyModule" ausführen
- poedit (unter kubuntu, Version 1.5.4, deutsche Version) starten
- die de_DE.po öffnen
- im Menü 'Katalog' -> 'Aus POT-Datei aktualisieren ...' wählen
- 'messages.pot' wählen

Nach Änderungen dann die de_DE.po speichern und das translate - Skript nochmal ausführen.

FAQ: General

File Upload does not work?

when trying to upload a file, the status wheel turns forever. Uploaded file is not stored.

This happens, if a javascript error occurse. You can only debug such a problem by using firebug or comparable developer tools. The MimeType of uploaded files is checked by default using the libmagic. Please make sure that:

- the fileinfo extention exists. On FreeBSD, this extension has to be installed. On Linux, this extention is normally included by default. Check it by: `php -m | grep fileinfo`
- Make sure, your Webserver can access `/usr/share/misc/magic*`. These files are referenced by YAWIK/vendor/zendframework/zend-validator/src/File/MimeType.php
- make sure the access is not restricted by an open_basedir setting

FAQ: Documentation

How can I improve the documentation?

The recommended way is to send us pull requests. If you don't know, how to do this, send us documentation via mail to contact@yawik.org.

FAQ: XML Feeds

Can I export jobs via xml feeds?

Yes. YAWIK offers since 0.28 a default xml feed for all public job openings via the route `lang/export/xml`. Means for our demo https://yawik.org/demo/en/export/xml.

If you want do modify the xml structure to fit you needs, override the view `jobs/export/feed`. The view script gets injected a paginator containing jobs as `jobs`.

If you need different XML formats for different channels (in case you are offering multiposting) you can access feeds for different channels by extending the route name with the channel name. Example: our demo offers a channel `yawik`. The feed for this channel can be accessed via https://yawik.org/demo/en/export/xml . If YAWIK finds a view script named `feed.yawik.xml.phtml`, it uses it to render the xml. Otherwise it uses the default structure defined by `feed.xml.phtml`. You can access a channel feed via: https://yawik.org/demo/en/export/xml/yawik

Channel feeds only contain jobs, which are public on a certain feed. The URLs to job openings and application forms are containing the channel name, which makes counting easy.

If you use the `YawikSolr` module, Solr results are injected into the feeds automatically. Data, which are not available in solr are lazy loaded from mongo.

FAQ: Customize Formulars

Is it possible to use a certain list of locations?

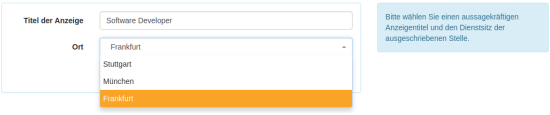
Yes. Since 0.29 it is possible to customize forms.

The *Geo* module offers to Form Elements. `LocationSelect`, which creates an autocomplete search fields for a location and `SimpleLocationSelect`, which can be used to create a select field with a certain list of locations.

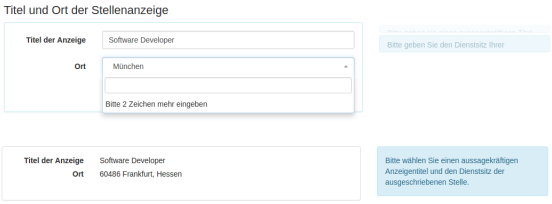
Example

Company XY has branch offices in *Frankfurt*, *München* and *Stuttgart*. The HR People want to simply select one of the location, if a job posting is created. This can be done by copying the Jobs/config/BaseFieldsetOptions.config.local.php.dist to the *config/autoload* directory.

When using `SimpleLocationSelect`, the form element comes with a search element. By entering a location, matching locations are fetched from a geo service.



When using `LocationSelect` the form looks like. There is a fixed list of locations. You can enrich your locations with all attributes of a Location entity


Titel und Ort der Stellenanzeige
Titel der Anzeige
Software Developer
Ort
München
Bitte 2 Zeichen mehr eingeben
Bitte geben Sie den Dienstsitz Ihrer

Titel der Anzeige
Software Developer
Ort
60486 Frankfurt, Hessen
Bitte wählen Sie einen aussagekräftigen Anzeigentitel und den Dienstsitz der ausgeschriebenen Stelle.

CHAPTER 10

Indices and tables

- genindex

Index

A

Applications, 40
Auth, 38

C

CompanyRegistration, 41
Core, 22
    Assets, 22
    Forms, 22
        Forms, 22, 23
    Headscripts, 33
    Logging, 33
    Mails, 34
    Navigation, 27
    Notifications, 31
    Options, 36
    Pagination, 28
        Views, 28

G

Geo, 49

J

Jobs, 43
JobsByMail, 52

M

Mail, 39

O

One-Click-Apply, 47
Orders, 42
Organizations, 41

P

PDF, 49

S

Settings, 52
Solr, 50

W

Widget, 47

Y

YawikXingVendorApi, 51