NodeBB Documentation

Redaktsioon 0.4.1

The NodeBB Team (https://github.com/NodeBB/NodeBB/graphs/c

sept 27, 2017

Sisukord

NodeBB on järgmise generatsiooni foorumiplatvorm, mis kasutab oma süsteemis web socketeid reaalajas muudatuste ning teadete kuvamiseks. NodeBB pakub mitmeid modernseid võimalusi, näiteks sotsiaalvõrgustikega integratsioon ning voogedastuse arutelud.

NodeBB on avatud lähtekoodiga projekt, mis on saadaval GitHubis. Ärge unustage ka vaatamast võimalusi, kuidas saate ise kasulik olla selle projekti jaoks - tõlked, vigade teatamine ja sõpradele rääkimine.

PEATÜKK 1

NodeBB paigaldamine

NodeBB paigaldamine OSile

Siit leiad õpetused, mis aitavad sul NodeBB foorumitarkvara paigaldada erinevatele operatsioonisüsteemidele.

Märkus: Kui sinu operatsioonisüsteemi siin listis ei eksisteeri, siis küsige õpetust meie foorumist või koostage ise üks õpetus.

Ubuntu

Esmalt paigaldame põhitarkvara:

```
$ sudo apt-get install git nodejs redis-server imagemagick npm
```

Kui soovid kasutada MongoDB'd, LevelDB'd, või hoopiski mõnda muud andmebaasi peale Redis andmebaasi, siis uuri lähemalt Andmebaasi konfigureerimine sektsioonist.

Kui sinu Ubuntu package manager installis Node.js versiooni, mis on väiksem kui 0.8, tuleb manuaalselt uuendada uuemale versioonile. Esiteks tehke kindlaks Node.js versioon `node --version`:

```
$ sudo add-apt-repository ppa:chris-lea/node.js
$ sudo apt-get update && sudo apt-get dist-upgrade
```

Järgmise sammuna kloonime NodeBB Githubi repo sobivasse asukohta:

```
$ git clone git://github.com/NodeBB/NodeBB.git nodebb
```

Paigaldame NodeBB foorumitarkvara:

```
$ cd nodebb
$ npm install
```

Käivitame NodeBB seadete muutmise režiimi `setup` flagiga:

```
$ ./nodebb setup
```

Algseaded on mõledud local serveri jaoks, kus Redis andmebaas töötab samuti samas keskkonnas.

Lõpetuseks tuleb foorum käivitada:

```
$ ./nodebb start
```

NodeBB'd saab käivtada ka muude programmidega, näiteks `supervisor` ja `forever`. *Vaata lähemalt siit*.

Debian

The current Ubuntu guide is not completely compatible with Debian and there are some specificities and especially the NodeJS installation, and how to get latest Redis.

Requirements

NodeBB requires these software to be installed:

- Node.js at least 0.10 and greater
- Redis, version 2.6 or greater
- cURL installed, just do `sudo apt-get install curl` in order to install it

Node.js installation

Debian 7 and Debian 6 and older doesn't have *nodejs* packages included by default, but there are some solutions to install Node.js on your Debian distribution.

Wheezy Backport :

This solution is **ONLY for Debian 7**, simply run the following **as root** :

```
$ echo "deb http://ftp.us.debian.org/debian wheezy-backports main" >> /etc/apt/
↪sources.list
$ apt-get update
```

To install Node.js + NPM, run this :

```
$ apt-get install nodejs-legacy
$ curl --insecure https://www.npmjs.org/install.sh | bash
```

The following install a Node.js version who is greater than 0.8 (at 29 March 2014 : 0.10.21)

Compiling from the source :

This solution is for Debian 6 (Squeeze) and greater, in order to install NodeJS, run this **as root** :

```
$ sudo apt-get install python g++ make checkinstall
$ src=$(mktemp -d) && cd $src
$ wget -N http://nodejs.org/dist/node-latest.tar.gz
$ tar xzvf node-latest.tar.gz && cd node-v*
$ ./configure
$ fakeroot checkinstall -y --install=no --pkgversion $(echo $(pwd) | sed -n -re's/.
↪+node-v(.+)$/\1/p') make -j$(($(nproc)+1)) install
$ sudo dpkg -i node_*
```

Get latest Software via DotDeb

Dotdeb is a repository containing packages to turn your Debian boxes into powerful, stable and up-to-date LAMP servers.

- Nginx,
- PHP 5.4 and 5.3 (useful PHP extensions : APC, imagick, Pinba, xcache, Xdebug, XHpro..)
- MySQL 5.5,
- Percona toolkit,
- Redis,
- Zabbix,
- Passenger…

Dotdeb supports :

- Debian 6.0 “Squeeze“ and 7 “Wheezy“
- both amd64 and i386 architectures

Debian 7 (Wheezy) :

For the complete DotDeb repositories :

```
$ sudo echo 'deb http://packages.dotdeb.org wheezy all' >> /etc/apt/sources.list
$ sudo echo 'deb-src http://packages.dotdeb.org wheezy all' >> /etc/apt/sources.list
```

After this, add the following GPC keys :

```
$ wget http://www.dotdeb.org/dotdeb.gpg
$ sudo apt-key add dotdeb.gpg
```

And update your package source :

```
$ sudo apt-get update
```

Debian 6 (Squeeze)

For the complete DotDeb repositories :

```
$ sudo echo 'deb http://packages.dotdeb.org squeeze all' >> /etc/apt/sources.list
$ sudo echo 'deb-src http://packages.dotdeb.org squeeze all' >> /etc/apt/sources.list
```

After this, add the following GPC keys :

```
$ wget http://www.dotdeb.org/dotdeb.gpg
$ sudo apt-key add dotdeb.gpg
```

And update your package source :

```
$ sudo apt-get update
```

Installing NodeBB

Now, we have NodeJS installed and Redis ready to be installed, run this command for install the base software stack :

```
$ apt-get install redis-server imagemagick git
```

Next clone this repository :

```
$ cd /path/to/nodebb/install/location
$ git clone git://github.com/NodeBB/NodeBB.git nodebb
```

Now we are going to install all dependencies for NodeBB via NPM :

```
$ cd /path/to/nodebb/install/location/nodebb (or if you are on your install location␣
↪directory run : cd nodebb)
$ npm install
```

Install NodeBB by running the app with *–setup* flag :

```
$ ./nodebb setup
```

1. ***URL of this installation*** **is either your public ip address or your domain name pointing to that ip address. Example:** `http://0.0.0.0` or `http://example.org`
2. **`Port number of your NodeBB` is the port needed to access your site: Note:** If you do not proxy your port with something like nginx then port 80 is recommended for production.
3. If you used the above steps to setup your redis-server then use the default redis settings.

And after all.. let's run the NodeBB forum

```
$ ./nodebb start
```

Note: If you NodeBB or your server crash, your NodeBB instance will not reboot (snap), this is why you should take a look at the other way to start your NodeBB instance with helper programs such as `supervisor` and `forever`, just *take a look here* it's simple as a click!

Extras, tips and Advice

You should secure your NodeBB installation, take a look here.

You should use Nginx (or similar) in order to reverse proxy your NodeBB installation on the port 80, *take a look here*

SmartOS

Requirements

NodeBB requires the following software to be installed:

- A version of Node.js at least 0.8 or greater.
- Redis, version 2.6 or greater (steps to install from Joyent's package repository given below).
- nginx, version 1.3.13 or greater (**only if** intending to use nginx to proxy requests to a NodeBB server).

Server Access

1. Sign in your Joyent account: Joyent.com
2. Select: `Create Instance`
3. Create the newest `smartos nodejs` image.

   Note: The following steps have been tested with image: `smartos nodejs 13.1.0`
4. Wait for your instance to show *Running* then click on its name.
5. Find your `Login` and admin password. If the `Credentials` section is missing, refresh the webpage.

   Example: `ssh root@0.0.0.0 A#Ca{c1@3`
6. SSH into your server as the admin not root: `ssh admin@0.0.0.0`

   Note: For Windows users that do not have ssh installed, here is an option: Cygwin.com

Installation

1. Install NodeBB's software dependencies:

   ```
   $ sudo pkgin update
   $ sudo pkgin install scmgit nodejs build-essential ImageMagick redis
   ```

   If any of these failed:

   ```
   $ pkgin search *failed-name*
   $ sudo pkgin install *available-name*
   ```

2. **If needed setup a redis-server with default settings as a service (automatically starts and restarts): Note:** These steps quickly setup a redis server but does not fine-tuned it for production.

   Note: If you ran *redis-server* manually then exit out of it now.

   ```
   $ svcadm enable redis
   $ svcs
   ```

   - If *svcs* shows "/pkgsrc/redis:default" in maintenance mode then:

   ```
   $ scvadm clear redis
   ```

   - To shut down your redis-server and keep it from restarting:

   ```
   $ scvadm disable redis
   ```

   - To start up your redis-server and have it always running:

```
$ scvadm enable redis
```

3. Move to where you want to create the nodebb folder:

```
$ cd /parent/directory/of/nodebb/
```

4. Clone NodeBB's repository:

```
$ git clone git://github.com/NodeBB/NodeBB.git nodebb
```

5. Install NodeBB's npm dependencies:

```
$ cd nodebb/
$ npm install
```

6. Run NodeBB's setup script:

```
$ node app --setup
```

   (a) *URL of this installation* is either your public ip address from your ssh *Login* or your domain name pointing to that ip address.

   Example: *http://0.0.0.0* or *http://example.org*

   (b) *Port number of your NodeBB* is the port needed to access your site:

   Note: If you do not proxy your port with something like nginx then port 80 is recommended for production.

   (c) If you used the above steps to setup your redis-server then use the default redis settings.

7. Start NodeBB process:

   Run NodeBB manually:

   Note: This should not be used for production.

```
$ node app
```

8. **Visit your app! Example:** With a port of 4567: `http://0.0.0.0:4567` or `http://example.org:4567`

   Note: With port 80 the *:80* does not need to be entered.

Note: If these instructions are unclear or if you run into trouble, please let us know by filing an issue.

Upgrading NodeBB

Note: Detailed upgrade instructions are listed in *Upgrading NodeBB*.

Windows 8

Required Software

First, install the following programs:

- https://windows.github.com/
- http://nodejs.org/

- http://sourceforge.net/projects/redis/files/redis-2.6.10/

You may have to restart your computer.

Running NodeBB

Start Redis Server

Märkus: The default location of Redis Server is

C:\Program Files (x86)\Redis\StartRedisServer.cmd

Open Git Shell, and type the following commands. Clone NodeBB repo:

```
git clone https://github.com/NodeBB/NodeBB.git
```

Enter directory:

```
cd NodeBB
```

Install dependencies:

```
npm install
```

Run interactive installation:

```
node app.js
```

You may leave all of the options as default.

And you're done! After the installation, run

```
node app.js
```

You can visit your forum at `http://127.0.0.1:4567/`

Developing on Windows

It's a bit of a pain to shutdown and restart NodeBB everytime you make changes. First install supervisor:

```
npm install -g supervisor
```

Open up bash:

```
bash
```

And run NodeBB on "watch" mode:

```
./nodebb watch
```

It will launch NodeBB in development mode, and watch files that change and automatically restart your forum.

- *Ubuntu*
- *Debian*
- *SmartOS*

- *Windows*
- CentOS (väline link)

NodeBB paigaldamine pilveserverile

Siit leiad õpetused, mis aitavad sul NodeBB foorumitarkvara paigaldada erinevatele pilveserveritele, mis põhinevad PaaS lahendusel.

Märkus: Kui sinu PaaS lahendusega pilveserveri teenust ei eksisteeri siin listis, siis küsige õpetust meie foorumist või koostage ise üks õpetus.

Heroku

Note: Installations to Heroku require a local machine with some flavour of unix, as NodeBB does not run on Windows.

1. Download and install Heroku Toolbelt for your operating system
2. Log into your Heroku account: `heroku login`
3. Verify your Heroku account by adding a credit card (at http://heroku.com/verify)
4. Clone the repository: `git clone https://github.com/NodeBB/NodeBB.git /path/to/repo/clone`
5. `cd /path/to/repo/clone`
6. Install dependencies locally `npm install`
7. Create the heroku app: `heroku create`
8. Enable WebSocket support (beta): `heroku labs:enable websockets -a {APP_NAME}`, where `{APP_NAME}` is provided by Heroku, and looks something like `adjective-noun-wxyz.herokuapp.com` (NOTE: See this doc): drop the *.herokuapp.com* when entering `{APP_NAME}` above.
9. Enable Redis To Go for your heroku account: `heroku addons:add redistogo:nano`
10. Run the NodeBB setup script: `node app --setup` (information for your Heroku server and Redis to Go instance can be found in your account page)
    - Your server name is found in your Heroku app's "settings" page, and looks something like `adjective-noun-wxyz.herokuapp.com`
    - Use any port number. It will be ignored.
    - Your redis server can be found as part of the redis url. For example, for the url: `redis://redistogo:h28h3wgh37fns7@crestfish.redistogo.com:12345/`
    - The server is `fishyfish.redistogo.com`
    - The port is `12345`
    - The password is `h28h3wgh37fns7`
12. Add the following two packages to the `dependencies` section of your `package.json`:

```
"dependencies": {
    ...
    "nodebb-plugin-dbsearch": "0.0.10",
    "redis": "~0.10.1",
    "connect-redis": "~2.0.0"
},
"devDependencies": {
```

13. Create a Procfile for Heroku: `echo "web: node app.js"> Procfile`
14. Commit the Procfile:

```
git add -f Procfile config.json package.json && git commit -am "adding Procfile and
↪configs for Heroku"
```

15. Push to heroku: `git push heroku master` * Ensure that a proper SSH key was added to your account, otherwise the push will not succeed!
16. Initialise a single dyno: `heroku ps:scale web=1`
17. Visit your app!

If these instructions are unclear or if you run into trouble, please let us know by filing an issue.

Keeping it up to date

If you wish to pull the latest changes from the git repository to your Heroku app:

1. Navigate to your repository at `/path/to/nodebb`
2. `git pull`
3. `npm install`
4. `node app --upgrade`
5. `git commit -am üpgrading to latest nodebb"`
6. `git push heroku master`

Cloud 9 IDE

The following are installation instructions for the Cloud 9 web based IDE.

Step 1: Clone NodeBB into a new workspace from GitHub. You can use the following command from the terminal:

```
git clone git://github.com/NodeBB/NodeBB.git nodebb
```

The nodebb command after the git url will create a file called nodebb so you have to CD into the file after you have cloned NodeBB.

Step 2: Install redis with Cloud9's package manager

```
nada-nix install redis
```

Step 3: Run your redis server on port 16379 - port 6379 tends to be already used on Cloud 9. The "&" makes the command run in the background. You can always terminate the process later. $IP is a Cloud 9 system variable containing the global ip of your server instance.

```
redis-server --port 16379 --bind $IP &
```

Step 4: Find out your instance's ip address so NodeBB can bind to it correctly. This is one of Cloud 9's demands and seems to be the only way it will work. You can't use $IP in your config.json either (which means you can't enter $IP in the node app –setup).

```
echo $IP
```

Step 5: Install NodeBB and it's dependencies:

```
npm install
```

Step 6: Run the nodebb setup utility:

```
node app --setup
```

URL of this installation should be set to 'http://workspace_name-c9-username.c9.io', replacing workspace_name with your workspace name and username with your username. Note that as NodeBB is currently using unsecure http for loading jQuery you will find it much easier using http:// instead of https:// for your base url. Otherwise jQuery won't load and NodeBB will break.

Port number isn't so important - Cloud9 may force you to use port 80 anyway. Just set it to 80. If this is another port, like 4567, that is also fine.

Use a port number to access NodeBB? Again, this doesn't seem to make a big difference. Set this to no. Either will work.

Host IP or address of your Redis instance: localhost (the output of the $IP Command is also acceptable)

IP or Hostname to bind to: Enter what your $IP value holds here found in step 4. It should look something like: 123.4.567.8

Host port of your Redis instance: 16379

Redis Password: Unless you have set one manually, Redis will be configured without a password. Leave this blank and press enter

First-time set-up will also require an Admin name, email address and password to be set.

And you're good to go! Don't use the Run button at the top if the IDE, it has been a little buggy for me. Besides, you're better off using the command line anyway. Run:

```
node app
```

And then open http://workspace_name-c9-username.c9.io in your browser.

Troubleshooting

A common problem is that the database hasn't been started. Make sure you have set Redis up correctly and ran

```
redis-server --port 16379 --bind $IP
```

- *Heroku*
- *Cloud9*
- Openshift
- Nitrous.IO

- Digital Ocean (väline link)

PEATÜKK 2

NodeBB konfigureerimine

Configuring Databases

NodeBB has a Database Abstraction Layer (DBAL) that allows one to write drivers for their database of choice. Currently we have the following options:

MongoDB

If you're afraid of running out of memory by using Redis, or want your forum to be more easily scalable, you can install NodeBB with MongoDB. This tutorial assumes you know how to SSH into your server and have root access.

These instructions are for Ubuntu. Adjust them accordingly for your distro.

Note: If you have to add `sudo` to any command, do so. No one is going to hold it against you ;)

Step 1: Install MongoDB

The latest and greatest MongoDB is required (or at least greater than the package manager). The instructions to install it can be found on the MongoDB manual).

Step 2: Install node.js

Like MongoDB, the latest and greatest node.js is required (or at least greater than the package manager), so I'm leaving this to the official wiki. The instructions to install can be found on Joyent.

Note: NPM is installed along with node.js, so there is no need to install it separately

Step 3: Install the Base Software Stack

Enter the following into the terminal to install the base software required to run NodeBB:

```
# apt-get install git build-essential imagemagick
```

Step 4: Clone the Repository

Enter the following into the terminal, replacing */path/to/nodebb/install/location* to where you would like NodeBB to be installed.

```
$ cd /path/to/nodebb/install/location
$ git clone git://github.com/NodeBB/NodeBB.git nodebb
```

Step 5: Install The Required NodeBB Dependencies

Go into the newly created *nodebb* directory and install the required dependencies by entering the following.

```
$ cd nodebb
$ npm install
```

Step 6: Adding a New Database With Users

To go into the MongoDB command line, type:

```
$ mongo
```

To add a new database called *nodebb*, type:

```
> use nodebb
```

To add a user to access the *nodebb* database, type:

```
> db.createUser( { user: "nodebb",
...       pwd: "<Enter in a secure password>",
...       roles: [ "readWrite" ] } )
```

Note: The role `readWrite` provides read or write any collection within a specific database to user.

Step 7: Configure MongoDB

MongoDB needs text search enabled. Modify `/etc/mongodb.conf`.

```
# nano /etc/mongodb.conf
```

Add `setParameter=textSearchEnabled=true` to the end. Also, to enable authentication, uncomment `auth = true`. Restart MongoDB.

```
# service mongodb restart
```

Step 8: Configuring NodeBB

Make sure you are in your NodeBB root folder. If not, just type:

```
$ cd /path/to/nodebb
```

To setup the app, type:

```
$ node app --setup
```

- Change the hostname to your domain name.
- Accept the defaults by pressing enter until it asks you what database you want to use. Type `mongo` in that field.
- Accept the default port, unless you changed it in the previous steps.
- Change your username to `nodebb`, unless you set it to another username.
- Enter in the password you made in step 5.
- Change the database to `nodebb`, unless you named it something else.

Continue with the installation, following the instructions the installer provides you.

Step 9: Starting the App

To start the app, run:

```
$ node app
```

Now visit `yourdomainorip.com:4567` and your NodeBB installation should be running.

NodeBB can also be started with helper programs, such as *supervisor or forever*. You can also use `nginx` as a *reverse proxy*).

LevelDB

Follow the *installation instructions* for your particular OS but feel free to omit the Redis installation.

After cloning NodeBB, ensure that you run:

```
npm install levelup leveldown
```

Finally, set up a directory to store your LevelDB database, for example:

```
mkdir /var/level/
```

Run the NodeBB install, select `level` when it prompts you for your database. If you created the folder as above, you can leave the rest of the questions as default.

- Redis (default, see *installation guides*)
- *Mongo*
- *Level*

Märkus: If you would like to write your own database driver for NodeBB, please visit our community forum and we can point you in the right direction.

Running a Secondary Database

Hoiatus: This option is experimental and should not be used on a production environment.

Both databases **must** be flushed before beginning - there isn't a mechanism yet that detects an existing installation on one database but not another. Until fail-safe's such as these are implemented this option is hidden under the `--advanced` setup flag.

```
node app --setup --advanced
```

Consult the other database guides for instructions on how to set up each specific database. Once you select a secondary database's modules, there's no turning back - until somebody writes an exporter/importer.

Currently this setup is being tested with Redis as the primary store (sets, lists, and sorted sets, because Redis is super fast with these), and Mongo as the hash store (post and user data, because ideally we wouldn't want this in RAM).

Configuring Web Server / Proxies

Here a few options that you can use to proxy your NodeBB forum.

Configuring nginx as a proxy

NodeBB by default runs on port `4567`, meaning that builds are usually accessed using a port number in addition to their hostname:

```
http://example.org:4567
```

In order to allow NodeBB to be served without a port, nginx can be set up to proxy all requests to a particular hostname (or subdomain) to an upstream NodeBB build running on any port.

Requirements

- **NGINX version v1.3.13 or greater**
  - Package managers may not provide a new enough version. To get the latest version, compile it yourself, or if on Ubuntu, use the NGINX Stable or NGINX Development PPA builds, if you are on Debian, use DotDeb repository to get the latest version of Nginx.
  - To determine your nginx version, execute `nginx -V` in a shell

Configuration

NGINX-served sites are contained in a `server` block. This block of options goes in a specific place based on how nginx was installed and configured:

- `/path/to/nginx/sites-available/*` – files here must be aliased to `../sites-enabled`
- `/path/to/nginx/conf.d/*.conf` – filenames must end in `.conf`
- `/path/to/nginx/httpd.conf` – if all else fails

Below is the basic nginx configuration for a NodeBB build running on port `4567`:

```
server {
    listen 80;

    server_name forum.example.org;

    location / {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_set_header X-NginX-Proxy true;

        proxy_pass http://127.0.0.1:4567/;
        proxy_redirect off;

        # Socket.IO Support
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}
```

Notes

- Remember to also edit *config.json* and change *use_port* from *true* to *false*
- nginx must be on version 1.4.x to properly support websockets. Debian/Ubuntu use 1.2, although it will work there will be a reduction in functionality.
- The `proxy_pass` IP should be `127.0.0.1` if your NodeBB is hosted on the same physical server as your nginx server. Update the port to match your NodeBB, if necessary.
- This config sets up your nginx server to listen to requests for `forum.example.org`. It doesn't magically route the internet to it, though, so you also have to update your DNS server to send requests for `forum.example.org` to the machine with nginx on it!

Configuring apache as a proxy

Prerequisites to making this work: Apache 2.4.x

What if I'm on 2.2.x (Debian/Ubuntu)?

you need to manually compile and add the module "mod_proxy_wstunnel" to the Apache 2.2 branch. If you're running Ubuntu or Debian, you're likely on the 2.2 branch of code.

The following guide will assist with that if you're on Debian or Ubuntu. This is what I used to backport the mod_proxy_wstunnel module to the 2.2 code base of Apache;

http://www.amoss.me.uk/2013/06/apache-2-2-websocket-proxying-ubuntu-mod_proxy_wstunnel/

NOTE: On ubuntu, if you're missing the ./configure file

You need to first run ./buildconf. After this is complete, you will then be able to use ./configure.

automake & libtool package was needed too.

```
apt-get install automake libtool
```

Enable the necessary modules

1. sudo a2enmod proxy
2. sudo a2enmod proxy_http
3. sudo a2enmod proxy_wstunnel

Add the config to Apache

The next step is adding the configuration to your virtualhost.conf file, typically located in /etc/apache2/sites-available/. The below configuration assumes you've used 4567 (default) port for NobeBB installation. It also assumes you have the bind address set to 127.0.0.1.

```
ProxyRequests off

<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>
ProxyPass /socket.io/1/websocket ws://127.0.0.1:4567/socket.io/1/websocket
ProxyPassReverse /socket.io/1/websocket ws://127.0.0.1:4567/socket.io/1/websocket

ProxyPass /socket.io/ http://127.0.0.1:4567/socket.io/
ProxyPassReverse /socket.io/ http://127.0.0.1:4567/socket.io/

ProxyPass / http://127.0.0.1:4567/
ProxyPassReverse / http://127.0.0.1:4567/
```

The last thing you need to be sure of is that the config.json in the NodeBB folder has use_port: false. Otherwise some functionality will not work properly.

Example nodebb/config.json

```
{
    "base_url": "http://www.yoursite.com",
    "port": "4567",
    "use_port": false,
    "secret": "55sb254c-62e3-4e23-9407-8655147562763",
    "bind_address": "127.0.0.1",
    "database": "redis",
    "redis": {
        "host": "127.0.0.1",
        "port": "6379",
        "password": "",
        "database": "0"
    },
    "bcrypt_rounds": 12,
    "upload_path": "/public/uploads",
    "relative_path": ""
}
```

Change the domain and dont use the secret in the example above.

Configuring Varnish Cache

To be sure Varnish will work properly with NodeBB check that your configuration `/etc/varnish/default.vcl` is optimized for **websockets**.

```
backend nodebb {
  .host = "127.0.0.1"; # your nodebb host
  .port = "4567"; # your nodebb port
}

sub vcl_recv {

  # Pipe websocket connections directly to Node.js
  if (req.http.Upgrade ~ "(?i)websocket") {
    set req.backend = nodebb;
    return (pipe);
  }

  # NodeBB
  if (req.http.host == "forum.yourwebsite.com") { # change this to match your host
    if (req.url ~ "^/socket.io/") {
        set req.backend = nodebb;
        return (pipe); # return pass seems not working for websockets
    }
    return (pass); # don't cache
  }

}

sub vcl_pipe {
  # Need to copy the upgrade header
  if (req.http.upgrade) {
    set bereq.http.upgrade = req.http.upgrade;
  }
}
```

- *Nginx*
- *Apache*
- *Varnish Cache*

PEATÜKK 3

NodeBB jooksutamine

Running NodeBB

The preferred way to start and stop NodeBB is by invoking its executable:

- `./nodebb start` Starts the NodeBB server
- `./nodebb stop` Stops the NodeBB server
- Alternatively, you may use `npm start` and `npm stop` to do the same

The methods listed below are alternatives to starting NodeBB via the executable.

Simple Node.js Process

To start NodeBB, run it with `node` (some distributions use the executable `nodejs`, please adjust accordingly):

```
$ cd /path/to/nodebb/install
$ node app
```

However, bear in mind that crashes will cause the NodeBB process to halt, bringing down your forum. Consider some of the more reliable options, below:

Supervisor Process

Using the supervisor package, you can have NodeBB restart itself if it crashes:

```
$ npm install -g supervisor
$ supervisor app
```

As `supervisor` by default continues to pipe output to `stdout`, it is best suited to development builds.

Forever Daemon

Another way to keep NodeBB up is to use the forever package via the command line interface, which can monitor NodeBB and re-launch it if necessary:

```
$ npm install -g forever
$ forever start app.js
```

PEATÜKK 4

NodeBB uuendamine

Upgrading NodeBB

NodeBB's periodic releases are located in the Releases. These releases contain what is usually considered the most bug-free code, and is designed to be used on production-level instances of NodeBB.

You can utilise git to install a specific version of NodeBB, and upgrade periodically as new releases are made.

To obtain the latest fixes and features, you can also `git clone` the latest version directly from the repository (`master` branch), although its stability cannot be guaranteed. Core developers will attempt to ensure that every commit results in a working client, even if individual features may not be 100% complete.

As always, the NodeBB team is not responsible for any misadventures, loss of data, data corruption, or any other bad things that may arise due to a botched upgrade - so please **don't forget to back up** before beginning!

Upgrade Path

NodeBB's upgrade path is designed so that upgrading between versions is straightforward. NodeBB will provide upgrade compatibility (via the `--upgrade` flag) between the latest version of a lower branch and the latest version of the higher branch. For example, if `v0.2.2` is the latest version in the `v0.2.x` branch, you can switch to the `v0.3.x` branch and suffer no ill effects. Upgrading from `v0.2.0` to `v0.3.x` is not supported, and NodeBB will warn you when attempting to upgrade that you are not upgrading cleanly.

Upgrading between patch revisions

e.g. v0.1.0 to v0.1.1

Patch revisions contain bugfixes and other minor changes. Updating to the latest version of code for your specific version branch is all that is usually required.

Execute steps 1 through 3.

Upgrading between minor revisions

e.g. v0.1.3 to v0.2.0

Minor revisions contain new features or substantial changes that are still backwards compatible. They may also contain dependent packages that require upgrading, and other features may be deprecated (but would ideally still be supported).

Execute steps 1 through 4.

Upgrade Steps

Note: After upgrading between revisions (i.e. v0.0.4 to v0.0.5), it may be necessary to run the following upgrade steps to ensure that any data schema changes are properly upgraded as well:

1. Shut down your forum

While it is possible to upgrade NodeBB while it is running, it is definitely not recommended, particularly if it is an active forum:

```
$ cd /path/to/nodebb
$ ./nodebb stop
```

2. Back up your data

Märkus: This section is incomplete, please take care to back up your files properly!

Backing up Redis

As with all upgrades, the first step is to **back up your data**! Nobody likes database corruption/misplacement.

All of the textual data stored in NodeBB is found in a `.rdb` file. On typical installs of Redis, the main database is found at `/var/lib/redis/dump.rdb`.

Store this file somewhere safe.

Backing up MongoDB

To run a backup of your complete MongoDB you can simply run

    mongodump

which will create a directory structure that can be restored with the *mongorestore* command.

It is recommended that you first shut down your database. On Debian / Ubuntu it's likely to be: *sudo service mongodb stop*

Backing up LevelDB

As LevelDB is simply a collection of flat files, just copy the database over to a safe location, ex.

```
cp -r /path/to/db /path/to/backups
```

Store this file somewhere safe.

Avatars

Uploaded images (avatars) are stored in /public/uploads. Feel free to back up this folder too:

```
cd /path/to/nodebb/public
tar -czf ~/nodebb_assets.tar.gz ./uploads
```

3. Grab the latest and greatest code

Navigate to your NodeBB: `$ cd /path/to/nodebb`.

If you are upgrading from a lower branch to a higher branch, switch branches as necessary. ***Make sure you are completely up-to-date on your current branch!***.

For example, if upgrading from `v0.3.2` to `v0.4.3`:

```
$ git fetch    # Grab the latest code from the NodeBB Repository
$ git checkout v0.4.x    # Type this as-is! Not v0.4.2 or v0.4.3, but "v0.4.x"!
$ git merge origin/v0.4.x
```

If not upgrading between branches, just run the following command:

```
$ git pull
```

This should retrieve the latest (and greatest) version of NodeBB from the repository.

Alternatively, download and extract the latest versioned copy of the code from the Releases Page. Overwrite any files as necessary. This method is not supported.

4. Run the NodeBB upgrade script

This script will install any missing dependencies, upgrade any plugins or themes (if an upgrade is available), and migrate the database if necessary.

```
$ ./nodebb upgrade
```

Note: `./nodebb upgrade` is only available after v0.3.0. If you are running an earlier version, run these instead:

- `npm install`
- `ls -d node_modules/nodebb* | xargs -n1 basename | xargs npm update`
- `node app --upgrade`

6. Start up NodeBB & Test!

You should now be running the latest version of NodeBB.

PEATÜKK 5

NodeBB haldamine

Administrative Functions

Märkus: These docs are out of date. If you are interested in updating these (preferably with screenshots) let us know on the community forum because we are planning on a design overhaul of the ACP soon.

To view the admin panel (if you are an admin): *http://your_nodebb_domain.com/admin*

Top Menu (all can be easily reached by other means)

- NodeBB ACP (Administrator Control Panel: this view) * *http://your_nodebb_domain.com/admin/index* (see Home below)
- Forum * *http://your_nodebb_domain.com* (your main forum)
- Home * *http://your_nodebb_domain.com/admin/index* (see Home below)
- Settings * *http://your_nodebb_domain.com/admin/settings* (see Settings below)

Side Menu: NodeBB

- Home * *http://your_nodebb_domain.com/admin/index* * Links all go to nodebb.com homepage
  - NOTE: should all the links go to same place?
  - Message reminder of what version this is and to check for updates (See *Upgrading NodeBB*) * NOTE: would a link to where the latest stable version is help?
  - Active Users * lists number users per page-path (?) * NOTE: not clear exactly what the paths mean or how to visit that path
- Categories * *http://your_nodebb_domain.com/admin/categories* * Filters: Active, Disabled, Unit Tests * List of Categories:
  - Icon, Name, Desc, Action: Disable
  - Actions: Save, Add New

- Users * *http://your_nodebb_domain.com/admin/users* * Filters: Latest Users, Top Posters, Most Reputation, Action: Search * List of Users:
  - Icon, Link: Name, Reputation(star), Number Posts(pencil), Action: Ban
  - Action: Load More
- Groups * *http://your_nodebb_domain.com/admin/groups*
  - List of Groups
  - Name, Desc, Icon
  - Action: Delete Group * NOTE: What exactly can Groups be set up to do, besides Admin?
- Topics * *http://your_nodebb_domain.com/admin/topics* * List of Topics
  - Name [link to topic], Posted When and By, Number posts (Topic+Replies), Thread Actions: Pin(pushpin), Lock(lock), Delete(trashcan)
  - Action: Load More Topics
  - Topic [from List of Topics link]
  - Normal View of Topic+Reply Posts but with: * Link, Edit, Delete Actions all enabled for each Post * Thread Tools:
    - Pin, Lock, Move, Delete
- Themes (See *Theming NodeBB*) * *http://your_nodebb_domain.com/admin/themes* * List of (Custom | Bootswatch) Themes
  - Actions: Use, Preview
  - Action: Revert (to base)
- Plugins (See *Writing Plugins for NodeBB*) * *http://your_nodebb_domain.com/admin/plugins* * List of Plugins
  - Action: De/activate
  - Info on making plugins
- Settings * *http://your_nodebb_domain.com/admin/settings*
  - General Settings * (textbox) Site Title * (textbox) Site Description * (textbox) Site Keywords * (textbox) Imgur Client ID
    - NOTE: How does this function?
    - (textbox) Maximum User Image Size
  - Privilege Thresholds (Use privilege thresholds to manage how much reputation a user must gain to receive moderator access.) * (textbox) Manage Thread * (textbox) Moderate Users * (textbox) Create Pinned Topics
  - Email Settings * (textbox) Email Address (The following email address refers to the email that the recipient will see in the "From" and "Reply To" fields.) * (textbox) SMTP Server Host (Default: 127.0.0.1) * (textbox) SMTP Server Port
  - User Settings * (textbox) Minimum Username Length * (textbox) Maximum Username Length * (textbox) Minimum Password Length
  - Post Settings * (textbox) Post Delay * (textbox) Minimum Title Length * (textbox) Minimum Post Length * (checkbox) Use Outgoing Links Warning Page
  - Action: Save

- Redis * *http://your_nodebb_domain.com/admin/redis* * Redis data storage stats
- Logger * *http://your_nodebb_domain.com/admin/logger* * (checkbox) Enable HTTP logging * (checkbox) Enable socket.io event logging * (textbox) Path to log file
- MOTD (Message of the Day) * *http://your_nodebb_domain.com/admin/motd* * (textarea) You can enter either full HTML or Markdown text. * (checkbox) Show the Message of the Day

Side Menu: Social Authentication (See *Enabling Social Network Logins*)

- Twitter
- *http://your_nodebb_domain.com/admin/twitter*
- Facebook
- *http://your_nodebb_domain.com/admin/facebook*
- Google+
- *http://your_nodebb_domain.com/admin/gplus*

Side Menu: Plugins (Shows installed plugins)

Side Menu: Unit Tests (Will run qunit tests)

Social Network SSOs

NodeBB supports integration for Facebook, Twitter, and Google through third party plugins:

- *npm install nodebb-plugin-sso-facebook*
- *npm install nodebb-plugin-sso-twitter*
- *npm install nodebb-plugin-sso-google*

Other SSO vendors are available, such as GitHub. Please check the plugin directory for a list of all SSO vendors.

After installing and activating them, they require an API key in order to function:

Facebook

Register an application via the Facebook Developers page. A credit card or mobile phone number may be required in order to create a Developer account.

Create a new application, and obtain an Application Key and Application Secret:

Ensure that "Website with Facebook Login" is checked, and that the URL to your NodeBB instance is specified in the "Site URL" box. Add that site's domain to the "App Domains" field.

Paste this key and secret into the appropriate boxes in the NodeBB Administration Panel (accessible via /admin on your NodeBB install)

Twitter

Register an application at the Twitter Developers page. Create a new Application, and obtain the Access Token and Secret:

Important: While setting up your application, be sure to specify a Callback URL. It does not have to correspond to your installation, it just cannot be blank.

Paste this token and secret into the appropriate boxes in the NodeBB Administration Panel (accessible via /admin on your NodeBB install)

Google

Register an application at the Google API Console, and obtain a Client ID and Secret.

Paste this ID and secret into the appropriate boxes in the NodeBB Administration Panel (accessible via /admin on your NodeBB install)

Image Hosting APIs

Enabling Imgur Image Uploads

To enable post image attachments, first create an imgur app from :

https://api.imgur.com/oauth2/addclient

You can use : "Anonymous usage without user authorization"

After that you will get a "Client ID".

Then install nodebb-plugin-imgur:

```
npm install nodebb-plugin-imgur
```

Activate the plugin from the control panel and restart NodeBB.

You should see a Imgur menu item in the control panel. Paste the Client ID to the "Imgur Client ID" in the plugin page. Save and you should be able to upload images by dragging them into the composer window.

Uploading to Amazon S3

Märkus: No documentation for this yet! See the plugin thread for more information.

PEATÜKK 6

NodeBB'le lisade kirjutamine

NodeBB Style Guide

For the most part, NodeBB follows the Google Javascript Style Guide.

Code Formatting

Märkus: The existing codebase as of July 2013 does not adhere to this style guide 100%. If you see instances where the style guide is not adhered to, feel free to restyle and send off a pull request.

Indentation & Bracing

NodeBB uses tabbed indentation. Bracing should follow the One True Brace Style:

```
if (condition) {
    // code here ...
} else {
    // otherwise ...
}
```

Put conditionals and statements on separate lines and wrap with curly braces even if it's just one line:

```
if (leTired) {
    haveANap();
}
```

Errors

Most callbacks return an error as the first parameter. Handle this error first before processing further.

```
someFunction(parameters, function(err, data) {
    if(err) {
       return callback(err); // or handle error
    }
    // proceed as usual
});
```

Variables

Variables should always be prefaced with the *var* keyword:

```
var foo = 'bar';
```

Multiple declarations are to be included in the same *var* statement:

```
var foo = 'bar',
    bar = 'baz';
```

Semicolons

Use semicolons if at all possible

Nomenclature

CamelCase if at all possible:

```
functionNamesLikeThis, variableNamesLikeThis, ClassNamesLikeThis, EnumNamesLikeThis,␣
↪methodNamesLikeThis, CONSTANT_VALUES_LIKE_THIS, foo.namespaceNamesLikeThis.bar, and␣
↪filenameslikethis.js.
```

Core Modules

Märkus: This section is under construction. Have a look at the modules folder for more information, located at:

```
public/src/modules
```

Alerts

The alert module is a toaster notification that can be called via the following syntax:

```
app.alert({
    title: 'Success!',
    message: 'Here\'s an example of an alert!',
    location: 'left-bottom',
    timeout: 2500,
    type: 'success',
    image: 'https://i.imgur.com/dJBzcGT.jpg'
});
```

The above code will result in this notification (default styling):

To style this, have a look at the vanilla theme's `modules/alert.less` and `templates/alert.tpl`.

Parameters:

1. `title` - string, which can be a language string as well. Some core language strings that you can use here include: `[[global:alert.success]]` and `[[global:alert.error]]`
2. `message` - string, which can be a language string as well.
3. `location` (optional) - `right-top` (default), `left-top`, `right-bottom`, `left-bottom`
4. `timeout` (optional) - integer in milliseconds, default is permanent until closed.
5. `type` - error, success, info, warning/notify
6. `image` (optional) - string, URL to image.
7. `closefn` (optional) - function. This is called when the user closes the alert via the (X) button.
8. `clickfn` (optional) - function. This is called when the user clicks on the alert.

PEATÜKK 7

Plugin-süsteem

Writing Plugins for NodeBB

So you want to write a plugin for NodeBB, that's fantastic! There are a couple of things you need to know before starting that will help you out.

Like WordPress, NodeBB's plugins are built on top of a hook system in NodeBB. This system exposes parts of NodeBB to plugin creators in a controlled way, and allows them to alter content while it passes through, or execute certain behaviours when triggered.

See the full *list of hooks* for more information.

Filters and Actions

There are two types of hooks: **filters** and **actions**.

Filters act on content, and can be useful if you want to alter certain pieces of content as it passes through NodeBB. For example, a filter may be used to alter posts so that any occurrences of "apple" gets changed to "orange". Likewise, filters may be used to beautify content (i.e. code filters), or remove offensive words (profanity filters).

Actions are executed at certain points of NodeBB, and are useful if you'd like to *do* something after a certain trigger. For example, an action hook can be used to notify an admin if a certain user has posted. Other uses include analytics recording, or automatic welcome posts on new user registration.

When you are writing your plugin, make sure a hook exists where you'd like something to happen. If a hook isn't present, file an issue and we'll include it in the next version of NodeBB.

Configuration

Each plugin package contains a configuration file called `plugin.json`. Here is a sample:

```
{
    "id": "my-plugin",
    "name": "My Awesome Plugin",
    "description": "Your plugin's description",
    "url": "Absolute URL to your plugin or a Github repository",
    "library": "./my-plugin.js",
    "staticDirs": {
        "images": "public/images"
    },
    "less": [
        "assets/style.less"
    ],
    "hooks": [
        { "hook": "filter:post.save", "method": "filter" },
        { "hook": "action:post.save", "method": "emailme" }
    ]
}
```

The `id` property is a unique name that identifies the plugin.

The `library` property is a relative path to the library in your package. It is automatically loaded by NodeBB (if the plugin is activated).

The `staticDirs` property is an object hash that maps out paths (relative to your plugin's root) to a directory that NodeBB will expose to the public at the route `/plugins/{YOUR-PLUGIN-ID}`.

- e.g. The `staticDirs` hash in the sample configuration maps `/path/to/your/plugin/public/images` to `/plugins/my-plugin/images`

The `less` property contains an array of paths (relative to your plugin's directory), that will be precompiled into the CSS served by NodeBB.

The `hooks` property is an array containing objects that tell NodeBB which hooks are used by your plugin, and what method in your library to invoke when that hook is called. Each object contains the following properties (those with a * are required):

- `hook`, the name of the NodeBB hook
- `method`, the method called in your plugin
- `priority`, the relative priority of the method when it is eventually called (default: 10)

Writing the plugin library

The core of your plugin is your library file, which gets automatically included by NodeBB if your plugin is activated.

Each method you write into your library takes a certain number of arguments, depending on how it is called:

- Filters send a single argument through to your method, while asynchronous methods can also accept a callback.
- Actions send a number of arguments (the exact number depends how the hook is implemented). These arguments are listed in the *list of hooks*.

Example library method

If we were to write method that listened for the `action:post.save` hook, we'd add the following line to the `hooks` portion of our `plugin.json` file:

```
{ "hook": "action:post.save", "method": "myMethod" }
```

Our library would be written like so:

```
var MyPlugin = {
        myMethod: function(postData) {
            // do something with postData here
        }
    };
```

Using NodeBB libraries to enhance your plugin

Occasionally, you may need to use NodeBB's libraries. For example, to verify that a user exists, you would need to call the `exists` method in the `User` class. To allow your plugin to access these NodeBB classes, use `module.parent.require`:

```
var User = module.parent.require('./user');
User.exists('foobar', function(err, exists) {
    // ...
});
```

Installing the plugin

In almost all cases, your plugin should be published in npm, and your package's name should be prefixed "nodebb-plugin-". This will allow users to install plugins directly into their instances by running `npm install`.

When installed via npm, your plugin **must** be prefixed with "nodebb-plugin-", or else it will not be found by NodeBB.

As of v0.0.5, "installing" a plugin by placing it in the `/plugins` folder is still supported, but keep in mind that the package `id` and its folder name must match exactly, or else NodeBB will not be able to load the plugin. *This feature may be deprecated in later versions of NodeBB.*

Testing

Run NodeBB in development mode:

```
./nodebb dev
```

This will expose the plugin debug logs, allowing you to see if your plugin is loaded, and its hooks registered. Activate your plugin from the administration panel, and test it out.

Disabling Plugins

You can disable plugins from the ACP, but if your forum is crashing due to a broken plugin you can reset all plugins by executing

```
./nodebb reset plugins
```

Alternatively, you can disable one plugin by running

```
./nodebb reset plugin="nodebb-plugin-im-broken"
```

Available Hooks

The following is a list of all hooks present in NodeBB. This list is intended to guide developers who are looking to write plugins for NodeBB. For more information, please consult *Writing Plugins for NodeBB*.

There are two types of hooks, **filters**, and **actions**. Filters take an input (provided as a single argument), parse it in some way, and return the changed value. Actions take multiple inputs, and execute actions based on the inputs received. Actions do not return anything.

Important: This list is by no means exhaustive. Hooks are added on an as-needed basis (or if we can see a potential use case ahead of time), and all requests to add new hooks to NodeBB should be sent to us via the issue tracker.

Filters

`filter:admin.header_build`

Allows plugins to create new navigation links in the ACP

`filter:post.save`

Argument(s): A post's content (markdown text)

Executed whenever a post is created or edited, but before it is saved into the database.

`filter:post.get`

Argument(s): A post object (javascript Object)

Executed whenever a post is retrieved, but before being sent to the client.

`filter:header.build`

Allows plugins to add new navigation links to NodeBB

`filter:register.build`

Argument(s):

- *req* the express request object (javascript Object)
- *res* the express response object (javascript Object)
- *data* the data passed to the template (javascript Object)

Allows plugins to add new elements to the registration form. At the moment, the only one supported is 'data.captcha'

`filter:post.parse`

Argument(s): A post or signature's raw text (String)

Executed when a post or signature needs to be parsed from raw text to HTML (for output to client). This is useful if you'd like to use a parser to prettify posts, such as Markdown, or BBCode.

filter:posts.custom_profile_info

Allows plugins to add custom profile information in the topic view's author post block

filter:register.check

Argument(s):

- *req* the express request object (javascript Object)
- *res* the express response object (javascript Object)
- *userData* the user data parsed from the form

Allows plugins to run checks on information and deny registration if necessary.

filter:scripts.get

Allows to add client-side JS to the header and queue up for minification on production

filter:uploadImage

filter:uploadFile

filter:widgets.getAreas

filter:widgets.getWidgets

filter:search.query

filter:post.parse

filter:messaging.save

filter:messaging.parse

filter:sounds.get

filter:post.getPosts

filter:post.getFields

filter:auth.init

filter:composer.help

filter:topic.thread_tools

filter:user.create

filter:user.delete

filter:user.profileLinks

filter:user.verify.code

Parameters: confirm_code

Ability to modify the generated verification code (ex. for using a shorter verification code instead for SMS verification)

filter:user.custom_fields

Parameters: userData

Allows you to append custom fields to the newly created user, ex. mobileNumber

filter:register.complete

Parameters: uid, destination

Set the post-registration destination, or do post-register tasks here.

filter:widget.render

Actions

action:app.load

Argument(s): None

Executed when NodeBB is loaded, used to kickstart scripts in plugins (i.e. cron jobs, etc)

action:page.load

Argument(s): An object containing the following properties:

- `template` - The template loaded
- `url` - Path to the page (relative to the site's base url)

action:plugin.activate

Argument(s): A String containing the plugin's `id` (e.g. `nodebb-plugin-markdown`)

Executed whenever a plugin is activated via the admin panel.

Important: Be sure to check the `id` that is sent in with this hook, otherwise your plugin will fire its registered hook method, even if your plugin was not the one that was activated.

action:plugin.deactivate

Argument(s): A String containing the plugin's `id` (e.g. `nodebb-plugin-markdown`)

Executed whenever a plugin is deactivated via the admin panel.

Important: Be sure to check the `id` that is sent in with this hook, otherwise your plugin will fire its registered hook method, even if your plugin was not the one that was deactivated.

action:post.save

Argument(s): A post object (javascript Object)

Executed whenever a post is created or edited, after it is saved into the database.

action:email.send

action:post.setField

action:topic.edit

action:post.edit

action:post.delete

action:post.restore

action:notification.pushed

Argument(s): A notification object (javascript Object)

Executed whenever a notification is pushed to a user.

action:config.set

action:topic.save

action:user.create

action:topic.delete

action:user.verify

Parameters: uid; a hash of confirmation data (ex. confirm_link, confirm_code) Useful for overriding the verification system. Currently if this hook is set, the email verification system is disabled outright.

action:user.set

Parameters: field (str), value, type ('set', 'increment', or 'decrement') Useful for things like awarding badges or achievements after a user has reached some value (ex. 100 posts)

action:settings.set

Parameters: hash (str), object (obj) Useful if your plugins want to cache settings instead of pulling from DB everytime a method is called. Listen to this and refresh accordingly.

Client Side Hooks

filter:categories.new_topic

action:popstate

action:ajaxify.start

action:ajaxify.loadingTemplates

action:ajaxify.loadingData

action:ajaxify.contentLoaded

action:ajaxify.end

action:reconnected

action:connected

action:disconnected

action:categories.loading

action:categories.loaded

action:categories.new_topic.loaded

action:topic.loading

action:topic.loaded

action:composer.loaded

action:widgets.loaded

Settings Framework

If you want to make your plugin customizable you may use the Settings Framework NodeBB offers.

Server-Side Access

First you need some default settings, just create a new object for this:

```
var defaultSettings = {
        booleans: {
            someBool: true,
            moreBools: [false, false, true]
        },
        strings: {
            someString: 'hello world',
            multiLineString: 'some\nlong\ntext',
```

```
            arrayOfStrings: ['some\nlong\ntexts', 'and another one']
        },
        numbers: {
            multiArrayDimensions: [[42,42],[21,21]],
            multiArrayDimensions2: [[42,42],[]],
            justSomeNumbers: [],
            oneNumber: 3,
            anotherNumber: 2
        },
        someKeys: ['C+S+#13'] // Ctrl+Shift+Enter
    };
```

Now you can use the server-side settings-module to access the saved settings like this:

```
var Settings = module.parent.require('./settings');
var mySettings = new Settings('myPlugin', '0.1', defaultSettings, function() {
    // the settings are ready and can accessed.
    console.log(mySettings === this); // true
    console.log(this.get('strings.someString') === mySettings.get().strings.
↪someString); // true
});
```

The second parameter should change at least every time the structure of default settings changes. Because of this it's recommended to use your plugins version.

To use the settings client-side you need to create a WebSocket that delivers the result of `mySettings.get()`.

The mySettings-object will cache the settings, so be sure to use methods like `mySettings.sync(callback)` when the settings got changed from somewhere else and `mySettings.persist(callback)` when you finished `mySettings.set(key, value)` calls.

You need to create a socket-listener like following to allow the admin to initiate a synchronization with the settings stored within database:

```
var SocketAdmin = module.parent.require('./socket.io/admin');
SocketAdmin.settings.syncMyPlugin = function() {
    mySettings.sync();
};
```

If you want to add a reset-functionality you need to create another socket-listener:

```
SocketAdmin.settings.getMyPluginDefaults = function (socket, data, callback) {
    callback(null, mySettings.createDefaultWrapper());
};
```

The methods of the `mySettings` object you probably want to use:

- `constructor()`
- **`sync([callback])`** Reloads the settings from database, overrides local changes.
- **`persist([callback])`** Saves the local changes within database.
- **`get([key])`** Returns the setting(s) identified by given key. If no key is provided the whole settings-object gets returned. If no such setting is saved the default value gets returned.
- **`set([key, ]value)`** Sets the setting of given key to given value. Remember that it's just a local change, you need to call `persist` in order to save the changes.
- **`reset([callback])`** Persists the default settings.

- **`getWrapper()`** Returns the local object as it would get saved within database.
- **`createWrapper(version, settings)`** Creates an object like it would get saved within database containing given information and settings.
- **`createDefaultWrapper()`** Creates an object like it would get saved within database containing the default settings.

Client-Side Access

The next step is making the settings available to the admin.

You need to use the *hooks* `filter:admin.header.build` (to display a link to your page within ACP) and `action:app.load` (to create the needed route).

Within your page you can access the client-side Settings API via

```
require(['settings'], function (settings) {
    var wrapper = $('#my_form_id');
    // [1]
    settings.sync('myPlugin', wrapper);
    // [2]
});
```

To make a button with the id `save` actually save the settings you can add the following at `[2]`:

```
$('#save').click(function(event) {
    event.preventDefault();
    settings.persist('myPlugin', wrapper, function(){
        socket.emit('admin.settings.syncMyPlugin');
    });
});
```

As said before the server-side settings-object caches the settings, so we emit a WebSocket to notify the server to synchronize the settings after they got persisted.

To use a reset-button you can add the following at `[2]`:

```
$('#reset').click(function(event) {
    event.preventDefault();
    socket.emit('admin.settings.getMyPluginDefaults', null, function (err, data) {
        settings.set('myPlugin', data, wrapper, function(){
            socket.emit('admin.settings.syncMyPlugin');
        });
    });
});
```

There you go, the basic structure is done. Now you need to add the form-fields.

Each field needs an attribute `data-key` to reference its position within the settings. The Framework does support any fields whose jQuery-object provides the value via the `val()` method.

The plugin to use for a field gets determined by its `data-type`, `type` or tag-name in this order.

Additionally the following plugins are registered by default:

- **array (types: div, array)** An Array of any other fields. Uses the object within `data-attributes` to define the array-elements. Uses `data-new` to define the value of new created elements.
- **key (types: key)** A field to input keyboard-combinations.

- **checkbox, number, select, textarea** Handle appropriate fields.

A full list of all attributes that may influence the behavior of the default Framework:

- data-key: the key to save/load the value within configuration-object
- data-type: highest priority type-definition to determine what kind of element it is or which plugin to associate
- type: normal priority type-definition
- data-empty: if `false` or `0` then values that are assumed as empty turn into null. data-empty of arrays affect their child-elements
- data-trim: if not `false` or `0` then values will get trimmed as defined by the elements type
- data-split: if set and the element doesn't belong to any plugin, it's value will get split and joined by its value into the field
- **array-elements:**
  - data-split: separator (HTML allowed) between the elements, defaults to `', '`
  - data-new: value to insert into new created elements
  - data-attributes: an object to set the attributes of the child HTML-elements. tagName as special key will set the tag-name of the child HTML-elements
- **key-fields:**
  - data-trim: if `false` or `0` then the value will get saved as string else as object providing following properties: `ctrl`, `alt`, `shift`, `meta`, `code`, `char`
  - data-split: separator between different modifiers and the key-code of the value that gets saved (only takes effect if trimming)
  - data-short: if not `false` or `0` then modifier-keys get saved as first uppercase character (only takes effect if trimming)
- **select:**
  - data-options: an array of objects containing `text` and `value` attributes.

The methods of the `settings` module:

- **`registerPlugin(plugin[, types])`** Registers the given plugin and associates it to the given types if any, otherwise the plugins default types will get used.
- **`get()`** Returns the saved object.
- **`set(hash, settings[, wrapper[, callback[, notify]]])`** Refills the fields with given settings and persists them. `hash` Identifies your plugins settings. `settings` The object to save in database (settings-wrapper if you use server-side Settings Framework). `wrapper` (default: 'form') The DOM-Element that contains all fields to fill. `callback` (default: null) Gets called when done. `notify` (default: true) Whether to display saved- and fail-notifications.
- **`sync(hash[, wrapper[, callback]])`** Resets the settings to saved ones and refills the fields.
- **`persist(hash[, wrapper[, callback[, notify]]])`** Reads the settings from given wrapper (default: 'form') and saves them within database.

For Settings 2.0 support the methods `load` and `save` are still available but not recommended.

Client-Side Example Template

An example template-file to use the same settings we already used server-side:

```
<h1>My Plugin</h1>
<hr />

<form id="my_form_id">
    <div class="row">
        <p>
            <h2>Settings</h2>
            A boolean: <input type="checkbox" data-key="booleans.someBool"></input>
↪<br>
            An array of checkboxes that are selected by default:
            <div data-key="booleans.moreBools" data-attributes='{"data-type":"checkbox
↪"}' data-new='true'></div><br>

            A simple input-field of any common type: <input type="password" data-key=
↪"strings.someString"></input><br>
            A simple textarea: <textarea data-key="strings.multiLineString"></
↪textarea><br>
            Array of textareas:
            <div data-key="strings.arrayOfStrings" data-attributes='{"data-type":
↪"textarea"}' data-new='Hello Kitty, ahem... World!'></div><br>

            2D-Array of numbers that persist even when empty (but not empty rows):
            <div data-key="numbers.multiArrayDimensions" data-split="<br>"
                data-attributes='{"data-type":"array","data-attributes":{"type":
↪"number"}}' data-new='[42,21]'></div><br>
            Same with persisting empty rows, but not empty numbers, if no row is␣
↪given null will get saved:
            <div data-key="numbers.multiArrayDimensions2" data-split="<br>" data-
↪empty="false"
                data-attributes='{"data-type":"array","data-empty":true,"data-
↪attributes":{"type":"number","data-empty":false}}' data-new='[42,21]'></div><br>
            Array of numbers (new: 42, step: 21):
            <div data-key="numbers.justSomeNumbers" data-attributes='{"data-type":
↪"number","step":21}' data-new='42'></div><br>
            Select with dynamic options:
            <select data-key="numbers.oneNumber" data-options='[{"value":"2","text":"2
↪"},{"value":"3","text":"3"}]'></select><br>
            Select that loads faster:
            <select data-key="numbers.anotherNumber"><br>
                <option value="2">2</option>
                <option value="3">3</option>
            </select>

            Array of Key-shortcuts (new: Ctrl+Shift+7):
            <div data-key="someKeys" data-attributes='{"data-type":"key"}' data-new=
↪'Ctrl+Shift+#55'></div><br>
        </p>
    </div>
    <button class="btn btn-lg btn-warning" id="reset">Reset</button>
    <button class="btn btn-lg btn-primary" id="save">Save</button>
</form>

<script>
    require(['settings'], function (settings) {
```

```
        var wrapper = $('#my_form_id');
        // [1]
        settings.sync('myPlugin', wrapper);
        $('#save').click(function(event) {
            event.preventDefault();
            settings.persist('myPlugin', wrapper, function(){
                socket.emit('admin.settings.syncMyPlugin');
            });
        });
        $('#reset').click(function(event) {
            event.preventDefault();
            socket.emit('admin.settings.getMyPluginDefaults', null, function (err,␣
↪data) {
                settings.set('myPlugin', data, wrapper, function(){
                    socket.emit('admin.settings.syncMyPlugin');
                });
            });
        });
      });
</script>
```

Custom Settings-Elements

If you want do define your own element-structure you can create a **plugin** for the Settings Framework.

This allows you to use a whole object like a single field which - besides comfort in using multiple similar objects - allows you to use them within arrays.

A plugin is basically an object that contains at least an attribute `types` that contains an array of strings that associate DOM-elements with your plugin.

You can add a plugin at `[1]` using the method `settings.registerPlugin`.

To customize the way the associated fields get interpreted you may add the following methods to your plugin-object:

All given elements are instances of JQuery.

All methods get called within Settings-scope.

- **`use()`** Gets called when the plugin gets registered.
- **`[HTML-Element|JQuery] create(type, tagName, data)`** Gets called when a new element should get created (eg. by expansion of an array).
- **`destruct(element)`** Gets called when the given element got removed from DOM (eg. by array-splice).
- **`init(element)`** Gets called when an element should get initialized (eg. after creation).
- **`[value] get(element, trim, empty)`** Gets called whenever the value of the given element is requested. `trim` Whether the result should get trimmed. `empty` Whether considered as empty values should get saved too.
- **`set(element, value, trim)`** Gets called whenever the value of the given element should be set to given one. `trim` Whether the value is assumed as trimmed.

For further impression take a look at the default plugins.

You should also take a look at the helper-functions within Settings in order to create your own plugins. There are a few methods that take response to call the methods of other plugins when fittingly.

PEATÜKK 8

Widget-süsteem

Writing Widgets for NodeBB

See the original blog post for a high level overview and screenshots of the widget system.

Embedding HTML and JavaScript

You don't need to be a developer to figure this out. Head over to the Themes control panel and click on the Widgets tab. Create a new HTML widget by dragging and dropping the widget onto whatever template you want.

Copy and paste HTML or JavaScript into the widget and hit save - you're done!

You can optionally give your widget a container by dragging and dropping from the containers section onto your selected widget.

If you're looking for some sample scripts, head over to our plugins section and look for any topic labelled `nodebb-script-xyz`. Don't forget to submit your scripts and ideas as well!

Creating Widgets

You can define widgets in both plugins and themes. If you're building a plugin which simply delivers a widget (or collection of widgets), we strongly suggest you follow the `nodebb-widget-xyz` nomenclature instead when publishing.

Registering your widget

Listen to this hook to register your widget:

```
"hook": "filter:widgets.getWidgets", "method": "defineWidgets", "callbacked": true
```

Pass this back in the array:

`Content` defines the form that is displayed to customize your widget in the admin panel.

Listening to your widget

NodeBB core will call your widget on the appropriate page load by way of the hooks system. The hook will be named after your widget's namespace (see previous example) - like so: `filter:widget.render:widget_namespace`

This will pass in an object with the following useful properties:

- `obj.area` - will have `location`, `template`, `url`
- `obj.data` - will have your admin-defined data; in the example from the previous section you will be exposed an `obj.data.myKey`

Defining Widget Areas in Themes

A Widget Area is characterized by a template and a location. Themes can share widgets if they define the same Widget Areas. If an admin switches themes, widgets that were previously defined in a Widget Area incompatible with the new theme are saved.

Listen to this hook to register your Widget Area:

```
"hook": "filter:widgets.getAreas", "method": "defineWidgetAreas", "callbacked": true
```

Pass this back in the array:

```
{
        name: "Category Sidebar",
        template: "category.tpl",
        location: "sidebar"
}
```

And that's all. You can define as many Widget Areas in your theme as you wish. If you're still stuck, have a look at this commit which upgraded the Cerulean theme to use the widget system.

PEATÜKK 9

Teemade mootor

Creating a new NodeBB Theme

NodeBB is built on Twitter Bootstrap, which makes theming incredibly simple.

Packaging for NodeBB

NodeBB expects any installed themes to be installed via `npm`. Each individual theme is an npm package, and users can install themes through the command line, ex.:

```
npm install nodebb-theme-modern-ui
```

The theme's folder must contain at least two files for it to be a valid theme:

1. `theme.json`
2. `theme.less`

`theme.less` is where your theme's styles will reside. NodeBB expects LESS to be present in this file, and will precompile it down to CSS on-demand. For more information regarding LESS, take a look at the project homepage.

Note: A *suggested* organization for `theme.less` is to `@import` multiple smaller files instead of placing all of the styles in the main `theme.less` file.

Configuration

The theme configuration file is a simple JSON string containing all appropriate meta data regarding the theme. Please take note of the following properties:

- `id`: A unique id for a theme (e.g. "my-theme")
- `name`: A user-friendly name for the theme (e.g. "My Theme")

- `description`: A one/two line description about the theme (e.g. "This is the theme I made for my personal NodeBB")
- `screenshot`: A filename (in the same folder) that is a preview image (ideally, 370x250, or an aspect ratio of 1.48:1)
- `url`: A fully qualified URL linking back to the theme's homepage/project

Child Themes

If your theme is based off of another theme, simply modify your LESS files to point to the other theme as a base:

topic.less

As `topic.less` from the theme `nodebb-theme-vanilla` was imported, those styles are automatically incorporated into your theme.

Important: If you depend on another theme, make sure that your theme specifically states this in its `package.json`. For example, for the above theme, as we depend on `nodebb-theme-vanilla`, we would explicitly state this by adding a new section into the `package.json` file:

```
"peerDependencies": {
    "nodebb-theme-vanilla": "~0.0.1"
}
```

Rendering Engine

How it works

Every page has an associated API call, Template file, and Language File.

For example, if you navigate to /topic/351/nodebb-wiki, the application will load three resources. The API return /api/topic/351/nodebb-wiki and the template, in this example, "topic.tpl", and the appropriate language file "topic.json"*.

Just prepend api/ to the URL's path name to discover the JSON return. Any value in that return can be utilized in your template.

*A page's name corresponds to the template and language's filename (ex. `http://domain.com/topic/xyz` correlates to `topic.tpl`). Sometimes this is not the case - ex. `/user/xyz` loads `account.tpl`. Have a look at the `custom_mapping` section in `public/templates/config.json` for more details.

Templating Basics

Using the API return as your guide, you can utilize any of those values in your template/logic. Using the above API call as an example, for anything in the root level of the return you can do something like:

```
{topic_name}
```

To access values in objects:

```
{privileges.read}
```

And finally you can loop through arrays and create blocks like so:

```
<!-- BEGIN posts -->
{posts.content}
<!-- END posts -->
```

The above will create X copies of the above block, for each item in the posts array.

Templating Logic

NodeBB's templating system implements some basic logic. Using the same API call as above for our example. You can write IF conditionals like so:

```
<!-- IF unreplied -->
This thread is unreplied!
<!-- ENDIF unreplied -->
```

Another example:

```
<!-- IF !disableSocialButtons -->
<button>Share on Facebook</button>
<!-- ELSE -->
Sharing has been disabled.
<!-- ENDIF !disableSocialButtons -->
```

We can check for the length of an array like so:

```
<!-- IF posts.length -->
There be some posts
<!-- ENDIF posts.length -->
```

While looping through an array, we can check if our current index is the @first or @last like so:

```
<!-- BEGIN posts -->
  <!-- IF @first -->
    <h1>Main Author: {posts.username}</h1>
  <!-- ENDIF @first -->
  {posts.content}
  <!-- IF @last -->
    End of posts. Click here to scroll to the top.
  <!-- ENDIF @last -->
<!-- END posts -->
```

For more advanced documentation, have a look at the templates.js repository

Exposing template variables to client-side JavaScript

There are two ways of letting our JS know about data from the server-side, apart from WebSockets (TODO: will be covered in a different article).

Via jQuery.get

If we require data from a different page we can make a `$.get` call to any other API call. For example, if we wanted to know more about a specific user we could make a call like so:

```
$.get(RELATIVE_PATH + '/api/user/psychobunny', {}, function(user) {
    console.log(user)
});
```

See this API call in action: http://community.nodebb.org/api/user/psychobunny

Via Template Variables

In topic.tpl for example, we can add a hidden input like so:

```
<input type="hidden" template-variable="pageCount" value="{pageCount}" />
```

The template system will immediately parse all of these and expose them via the following method:

```
ajaxify.variables.get('pageCount');
```

This is the ideal method of letting JS know about important variables within the template.

Internationalization

The template engine interfaces with the internationalization system as well. We can embed variables into language strings. Let's use this API call as well as this language file as an example. We can now do something like the following:

```
[[register:help.username_restrictions, {minimumUsernameLength},
↪{maximumUsernameLength}]]
```

Which will translate this string:

```
A unique username between %1 and %2 characters
```

to

```
A unique username between 2 and 16 characters
```

Advanced Topics

Dynamically requiring and rendering a template file from client-side JavaScript

The template engine lazy loads templates on an as-needed basis and caches them. If your code requires a template or partial on-demand then you can :

```
ajaxify.loadTemplate('myTemplate', function(myTemplate) {
        var html = templates.parse(myTemplate, myData);
});
```

You can also access the invidual blocks inside each template, which is handy for doing things like (for example) rendering a new post's `<li>` and dynamically sticking it in an already loaded `<ul>`

```
Some stuff here...
<!-- BEGIN posts -->
We just want to pull this block only.
<!-- END posts -->
... some stuff here
```

```
ajaxify.loadTemplate('myTemplate', function(myTemplate) {
        var block = templates.getBlock(myTemplate, 'posts');
        var html = templates.parse(block, myData);
});
```

Rendering templates on server-side Node.js

The templating system hooks into Express just like most other templating frameworks. Just use either `app.render` or `res.render` to parse the appropriate template.

```
res.render('myTemplate', myData);
```

```
app.render('myTemplate', myData, function(err, parsedTemplate) {
        console.log(parsedTemplate);
});
```

PEATÜKK 10

Arendajate vahendid

Developer's Resources

Märkus: This section is under construction.

Core

- Building a new Admin Page (Out of date)

Plugins

- Developing plugins with Grunt and Coffeescript
- Writing your first NodeBB plugin

Themes

Widgets

PEATÜKK 11

NodeBB projektile kaasa aitamine

Helping out the NodeBB Project

NodeBB is an open source project, and will forever remain free. Here's a number of ways you can help us, even if you aren't a programmer.

- Like and share our content on Facebook
- Follow us on Twitter and perhaps tweet **#NodeBB is most awesome forum software @NodeBB**
- Update our wiki! ;) We need everything from development/design tutorials to user friendly how-to guides.
- Tell everybody about NodeBB, including your grandma and her cats.
- Submit a pull request, or two, or three..
- Build a new theme
- Write a plugin
- Keep the link back to us on the footer of your own NodeBB :)
- Blog about us! Give the gift of SEO juice this Christmas
- Help Translate NodeBB - It's a really simple translation tool and you don't need to know how to code.
- Join our community and give us a hard time about bugs and missing features

Translating NodeBB to another language

NodeBB uses Transifex, which is a user friendly visual tool which allows any individual to translate text into a language of their choice. You don't need to be a programmer to do this, so what are you waiting for? Join the translation team now :)

Writing Documentation

These docs were written using Sphinx and published using rtfd.org.

You can edit these docs directly on GitHub, or by clicking on "View page source" on the top right of any page.

If you wish, you can clone the repository and compile the documentation yourself. Check out the Getting Started section for more info on how to accomplish the latter.

Documentation are auto-compiled and pushed to rtfd.org after every commit.

PEATÜKK 12

Vigade otsimine / Abi

Need Help?

Frequently Asked Questions

If you experience difficulties setting up a NodeBB instance, perhaps one of the following may help.

How do I start/stop/restart NodeBB?

You can call the `./nodebb` executable to start and stop NodeBB:

```
$ ./nodebb
    Welcome to NodeBB
    Usage: ./nodebb {start|stop|reload|restart|log|setup|reset|upgrade|dev|watch}

    start    Start the NodeBB server
    stop     Stops the NodeBB server
    reload   Restarts NodeBB
    restart  Restarts NodeBB
    log      Opens the logging interface (useful for debugging)
    setup    Runs the NodeBB setup script
    reset    Disables all plugins, restores the default theme.
    upgrade  Run NodeBB upgrade scripts, ensure packages are up-to-date
    dev      Start NodeBB in interactive development mode
    watch    Start NodeBB in development mode and watch for changes
```

How do I upgrade my NodeBB?

Please consult *Upgrading NodeBB*

I upgraded NodeBB and now X isn't working properly!

Please consult *Upgrading NodeBB*

I installed an incompatible plugin, and now my forum won't start!

If you know which plugin caused problems, disable it by running: `./nodebb reset plugin="nodebb-plugin-pluginName"`

Otherwise, disable all plugins by running: `./nodebb reset plugins`

Is it possible to install NodeBB via FTP?

It is possible to transfer the files to your remote server using FTP, but you do require shell access to the server in order to actually "start" NodeBB. Here is a handy guide for installing NodeBB on DigitalOcean

I'm getting an "npm ERR!" error

For the most part, errors involving `npm` are due to Node.js being outdated. If you see an error similar to this one while running `npm install`:

```
npm ERR! Unsupported
npm ERR! Not compatible with your version of node/npm: connect@2.7.11
```

You'll need to update your Node.js version to 0.8 or higher.

To do this on Ubuntu:

```
# add-apt-repository ppa:chris-lea/node.js
# apt-get update && apt-get dist-upgrade -y
```

If successful, running the following command should show a version higher than 0.8

```
# apt-cache policy nodejs
```

URLs on my NodeBB (or emails) still have the port number in them!

If you are using *nginx* or *Apache* as a reverse proxy, you don't need the port to be shown. Simply run *./nodebb setup* and specify the base URL without a port number.

Alternatively, edit the `config.json` file using your favourite text editor and change `use_port` to `false`.

Submit Bugs on our Issue Tracker

Before reporting bugs, please ensure that the issue has not already been filed on our tracker, or has already been resolved on our support forum. If it has not been filed, feel free to create an account on GitHub and create a new issue.

Ask the NodeBB Community

Having trouble installing NodeBB? Or did something break? Don't hesitate to join our forum and ask for help. Hopefully one day you'll be able to help others too :)

PEATÜKK 13

Indeksid ja tabelid

- genindex
- modindex
- search