Overview

Casnode is a forum developed by Casbin community. The detailed architecture:

Name	Description	Tools	Source code
Frontend	Web frontend UI for Casnode	JavaScript and React	https://github.com/casbin/casnode/tree/master/web
Backend	RESTful API backend for Casnode	Golang + Beego + MySQL + Xorm	https://github.com/casbin/casnode/

Casbin community uses the forum as the official forum: Casnode.

Casnode is open source, you can get the code here.

Casnode is easy to use. It has detailed description in user interfaces so normal users can easily get start with Casnode. This document is prepared for administrators who want to get full use of Casnode.

Main Features

There special features distinguish Casnode from other forums. Here is a short introduction of these features. Read forward for more details.

📧 Mailing list

Casnode supports Google Groups well. By integrate google-group-crawler, after setting up a Google Group config in Casnode, a two-way synchronization will be started. For now, Casnode can only sync conversations from Google Groups, but can push conversations to any other mailing lists.

✨ Multi-platform

Casnode supports computer and mobile access. Frontend UI adapts to PC and mobile.

🔍 Built-in search

Casnode support built-in search, of course, also support search using various engines such as Google.

🖼️ Upload pictures and attachments

Casnode support drag and drop to upload pictures and attachments. At the same

time, each account has its own file library and quota. Files in the file library can also be shared using sharing links, making it more convenient to post pictures and pictures. Supports various OSS storages such as Alibaba Cloud and Tencent Cloud.

📢 Site advertisement

Support for setting up site advertisements, which can be delivered independently through the background.

🎯 Server side rendering

Casnode support server side rendering, friendly to search engine SEO.

🗄️ All kinds of databases

Casnode uses xorm to connect to databases. You can use MySQL, sqlite3, mymysql and Postgres for Casnode.

🌐 Multi-languages

Casnode uses i18next and Crowdin to support multi-language. Now Casnode supports Chinese, English, French, German, Russian, Japanese and Korean. Welcome to make a PR or file an issue to support your language! And any translation contribution is welcomed!

🚪 Use Casdoor to manage members

Casnode does not maintain a member table in the database. Casnode uses Casdoor to manage users. Casdoor is an SSO platform based on OAuth2.0. Through Casdoor, Casnode supports a variety of third-party login/registration methods. There are many registration methods, mobile phone, email, QQ, WeChat, GitHub, Facebook, Google, LinkedIn, DingTalk, Gitee, wecom and GitLab. It also supports graphic verification codes for Human-machine verification. If you use Casdoor for your organization management, your members can sign into Casnode directly, no need to signing up for Casnode again. To obtain more detailed features, please go to Casdoor.

Installation

These paragraphs will help you deploy Casnode on your server. If you want to install Casnode by BT Panel or Docker, please see the details at: BT Panel and Docker.

Please clone Casnode first:

```
git clone https://github.com/casbin/casnode
```

and follow these steps to easily setup your own forum!

Production Environment

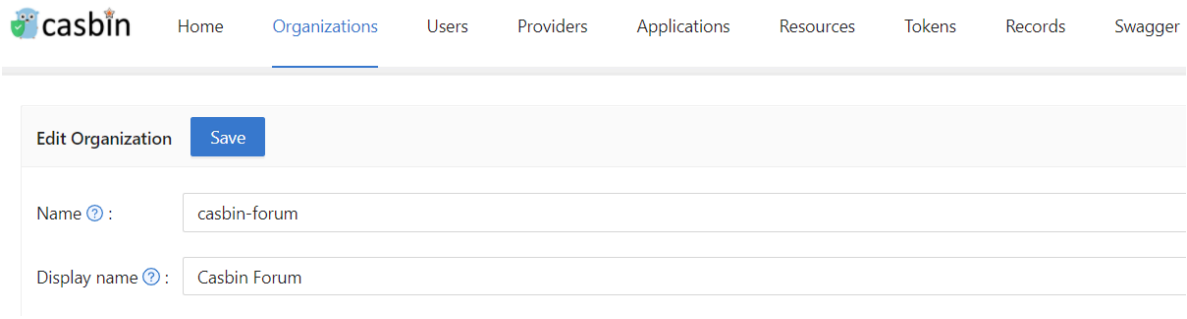
1. Setup Casdoor

Casnode uses Casdoor to manage members. So you need to create an organization and an application for Casnode in a Casdoor instance.

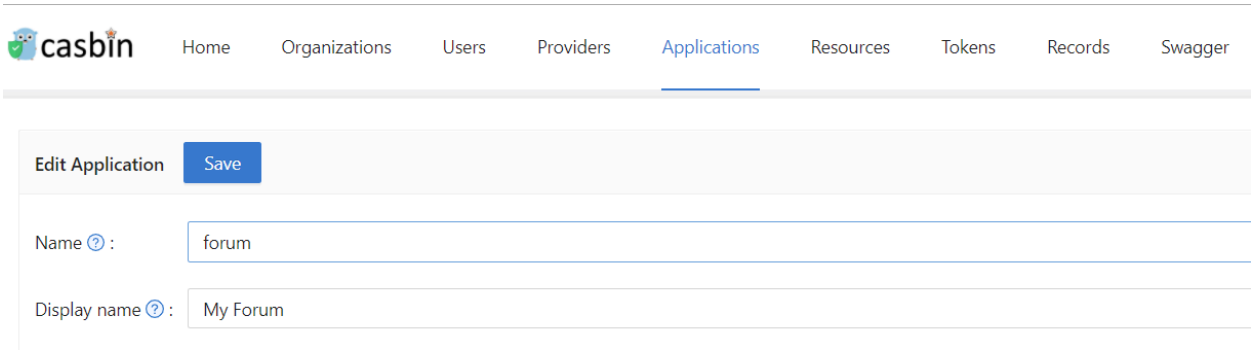
Follow these steps to setup Casdoor for Casnode:

- Navigate to Casdoor (Click here for details of Casdoor)
- Sign into the organization "built-in"
- Click **Organizations** in the top bar
- Click **add** button

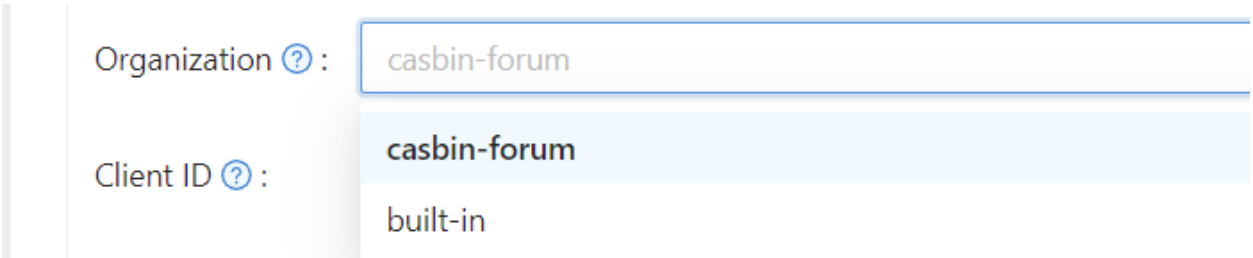
- Remember the Organization name, here I use **casbin-forum** as my organization name:



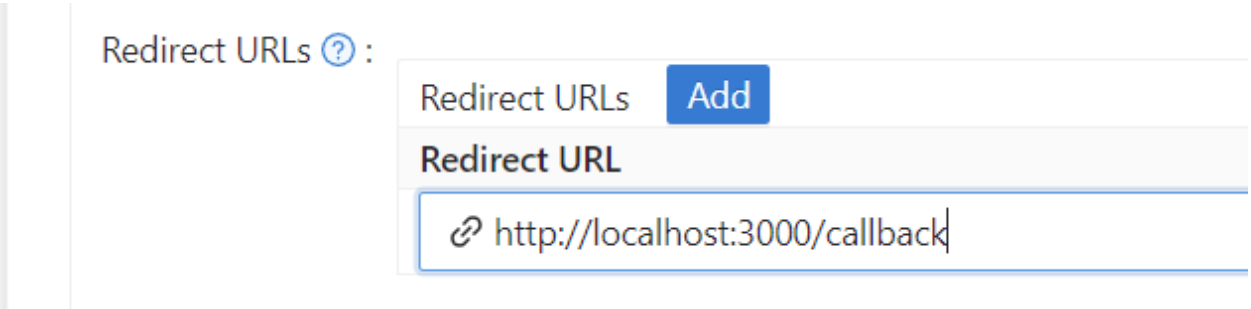
- Click **Applications** in the top bar
- Click **add** button
- Remember the Application name, here I use forum as my application name:
- Click **Edit**



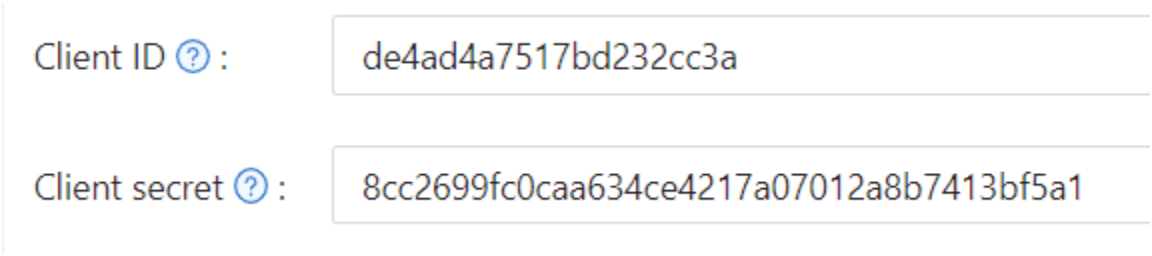
- Select the organization you just created as the application organization



- Modify the redirect URLs to the forum URL. If you are in a developing environment, your redirect URL is **http://localhost:3000/callback**. If you are in a production environment, your redirect URL is **http://yourip:7000/callback**



- Click **Save** and remember the `Client ID` and `Client Secret`



2. Modify `conf/app.conf`

Here is an explanation of the config items:

Database connection:

Casnode database

```
driverName = mysql
dataSourceName = root:123@tcp(localhost:3306)/
dbName = casnode
```

Casdoor database

```
casdoorDbName = casdoor
```

TIP

Casdoor's `driverName` and `dataSourceName` are the same as casnode by default. If your Casdoor and Casnode are not in the same database, you can set up the casdoor database in casdoor/adapter.go

Here we provide an example:

1. Add casdoor configuration in `conf/app.conf`:

```
casdoorDriverName =
casdoorDataSourceName =
```

2. Modify the `func InitCasdoorAdapter()`:

```
    adapter =
NewAdapter(beego.AppConfig.String("casdoorDriverName"),
beego.AppConfig.String("casdoorDriverName"),
beego.AppConfig.String("casdoorDbName"))
```

Object Storage Service (Casnode uses OSS to store resources):

```
OSSProvider = ""
accessKeyID     = ""
accessKeySecret = ""
OSSCustomDomain = ""
```

If you can not access Google in normal ways, you need to set up a http proxy here:

```
httpProxy = "127.0.0.1:10808"
```

Casdoor config:

```
# Your Casdoor endpoint in step 1
casdoorEndpoint = http://localhost:8000

# Client ID you copied in step 1
clientId = xxx

# Client Secret you copied in step 1
clientSecret = xxx

jwtSecret = CasdoorSecret

# Organization name in step 1
casdoorOrganization = "casbin-forum"
```

3. Modify `web/src/Conf.js`

```
export const AuthConfig = {
  // Your Casdoor endpoint in step 1
  serverUrl: "http://localhost:7001",

  // Client ID you copied in step 1
  clientId: "014ae4bd048734ca2dea",

  // Application name you copied in step 1
  appName: "app-casbin-forum",
```

4. Build front end

In folder `web`, run the following commands:

Yarn npm

```
yarn install && yarn run build
```

```
npm install && npm run build
```

5. Build back end

In repository root, run:

```
go build
./casnode
```

Then the Casnode app should run on port 7000. You can setup a nginx proxy pass to manage SSL or something else.

For most of site owners who want to develop a forum using Casnode, steps above is enough. But if you are a developer, want to contribute to Casnode, or modify the code to suit your own environment, then you can run Casnode in the developing mode. Please follow these steps to start developing mode:

Dev Environment

1. Do step 1-3 above

4. Run back end

```
go run main.go
```

5. Run front end

In `web` folder:

Yarn npm

```
yarn install
yarn run start
```

```
npm install
npm run start
```

Now, Casnode runs its front end at port 3000 and runs it's back end at port 7000. You can modify the code and see what will happen.

> ⚠ CAUTION
>
> The front end uses these codes to determine whether it is a dev mode:

```
export function initServerUrl() {
  const hostname = window.location.hostname;
  if (hostname === "localhost") {
    ServerUrl = `http://${hostname}:7000`;
  }
}
```

It means if hostname is `localhost`, then you are in dev mode. If not, then you are in productive mode. Port of the back end is not same in dev mode and productive mode, so please do not use `127.0.0.1` instead of `localhost` in your browser in dev mode.

BT panel

> CAUTION
>
> The tutorial environment is Ubuntu 20.04

Install Casnode under the Linux BT panel

Prepare work

After installing the BT panel, the browser visits the BT panel, selects the software store, searches for and installs MySQL, and then searches for node, you can see that there is a PM2 manager, install the PM2 manager.

After the installation is complete, disconnect from the server or restart the server, node will be automatically written into the environment variable.

Enter `git --version` to make sure git is , if the prompts Command `git` not found, use `apt-get install git` to install git.

> TIP
>
> To access the Casnode successfully, you need to open the **7001** and **8000** port.

Install Golang

The root user executes the following commands to download and decompress the

Go binary file to the **/usr/local** directory.

```
wget -c https://dl.google.com/go/go1.16.5.linux-amd64.tar.gz -O
- | sudo tar -xz -C /usr/local
```

Then we need to add Golang to the environment variables, edit `/etc/profile`, add the following code in the last line of the file.

```
export GOROOT=/usr/local/go
export PATH=$PATH:$GOROOT/bin
```

Then use command `source /etc/profile` to make the newly added environment variables work.

Now, enter `go version`, you will see the go version, and we installed it successfully. If you can't connect to GitHub, you can set up the mirror. The command is

```
go env -w GOPROXY=https://goproxy.cn,direct
```

Git clone Casnode & Casdoor

Next, execute the following commands in the folder where you want to store the project.

```
git clone https://github.com/casbin/casdoor.git
git clone https://github.com/casbin/casnode.git
```

Now, you can see there are two folders, Casnode and Casdoor.

Configure Casdoor

Run Casdoor

We first configure Casdoor.

```
cd casdoor
go build main.go
```

Then edit `conf/app.conf`, find

```
dataSourceName = root:123@tcp(localhost:3306)/
```

Change MySQL password provided by the BT panel as **123**.

```
cd web
npm install
npm run build
cd ..
sudo nohup ./main &
```

Configure Casnode in Casdoor

Now that Casdoor has been configured, visit http://your-ip:8000 to configure Casnode.

The default administrator login account is `admin/123`.

Click Organization, then click Add, click Edit for the added organization, and change the name to the organization name you want. Here I set it to casbin-forum, and then click Save.

Click Applications, then click Add, for the application you just added, click Edit, change the name to the application name you want, I changed it to app-casbin-forum. Click on the organization, select the organization you just added, my organization is casbin- forum. Click Redirect URLs, modify the link in the box to http://your-ip:7000/callback.Finally, remember the Client ID and Client Secret, and click Save.

Click Users, click Add, then click Edit, modify the added user, click Organization, select casbin-forum, and click is admin. Finally click Save, now your organization has an administrator account.

Configure Casnode

Next we configure in Casnode.

```
cd casnode
go build main.go
```

Edit `conf/app.conf`, find

```
dataSourceName = root:123@tcp(localhost:3306)/
```

Change MySQL password provided by the BT panel to **123**, then find casdoorEndpoint, modify it to http://your-ip:8000 (Casdoor backend address), find **ClientId** and **ClientSecret**, and modify them to the previously remembered Application client id and client secret, find casdoorOrganization, modify the organization name to your set. Save and exit.

Edit `web/src/Conf.js`, modify `serverUrl` to http://your-ip:8000 (Casdoor front-

end access address), modify `ClientId` to the ClientId of the application just added, modify `appName` to the set application name, and modify `organizationName` to the set organization name.

```
npm install
npm run build
cd ..
nohup ./main &
```

Next visit **http://your-ip:7000**, click login, enter the account you added before, user_1/123, you have now successfully logged in to Casnode.

For more settings please see Casnode.

Docker

> ⚠ CAUTION
>
> The tutorial environment is Ubuntu 20.0.4 .

Install casnode through docker

Prepare work

Install docker and docker-compose

Install Docker and Docker-compose, you see docker and docker-compose

Clone casnode & casdoor

Next, clone Casdoor and Casndoe from GitHub.

```
git clone https://github.com/casbin/casdoor.git
git clone https://github.com/casbin/casnode.git
```

Now, you can see two folders, `casnode` and `casdoor`.

Configure casdoor

Run casdoor

We first configure casdoor.

Edit `conf/app.conf`, modify **dataSourceName = root:123@tcp(localhost:3306)/ to dataSourceName = root:123@tcp(db:3306)/**

```
docker-compose up
```

> TIP
>
> mysql and casdoor are in different docker containers.

Configure casnode in casdoor

Now that Casdoor has been configured, visit http://your-ip:8000 to configure Casnode.

The default administrator login account is `admin/123`.

Click Organization, then click Add, click Edit for the added organization, and change the name to the organization name you want. Here I set it to casbin-forum, and then click Save.

Click Applications, then click Add, for the application you just added, click Edit, change the name to the application name you want, I changed it to app-casbin-forum.Click on the organization, select the organization you just added, my organization is casbin- forum. Click Redirect URLs, modify the link in the box to `http://your-ip:7000/callback`.Finally, remember the Client ID and Client Secret, and click Save.

Click Users, click Add, then click Edit, modify the added user, click Organization, select casbin-forum, and click is admin. Finally click Save, now your organization has an administrator account.

Configure casnode

Next we configure in Casnode.

Edit `conf/app.conf`, modify **dataSourceName = root:123@tcp(localhost:3306)/** to **dataSourceName = root:123@tcp(db:3306)/** so that the data come from your database.

Then find `casdoorEndpoint`, modify it to `http://your-ip:8000` (Casdoor backend address), find `clientId` and `clientSecret`, and modify them to the previously remembered Application client id and client secret, find casdoorOrganization, modify the organization name to you set.

Edit `web/src/Conf.js` , modify `serverUrl` to http://your-ip:8000 (Casdoor front-end access address), modify `clientId` to the clientId of the application just added, modify `appName` to the set application name, and modify `organizationName` to the set organization name.

Next, run casnode with docker

```
docker-compose up
```

Next visit http://your-ip:7000, click login, enter the account you added before, user_1/123, you have now successfully logged in to Casnode.

More settings reference casnode.

Migration from DiscuzX

Casnode has provided a lot of Go scripts to help users migrate their forums from DiscuzX 3.x to Casnode. The scripts are located at: https://github.com/casbin/casnode/tree/master/discuzx

A Xorm-supported database (e.g., MySQL) is used by Cansode to store forum data like topics and replies. Object storage (as the form of Casdoor storage provider) is used by Casnode to store images and attachment files.

Preparation

You need to prepare the following environments before conducting the migration:

1. A cloud VM (better with above 4 cores and 8GB memory, 8 cores and 16GB is even better), better with Intranet connection with the database for higher speed. This VM is used to run the Go migration scripts.
2. A Casnode git repository with the Go migration scripts in the above VM (the Casnode instance can be running or stopped).
3. A running Casdoor instance (better in the same cloud VM for higher speed), with at least one object storage is configured as a Casdoor storage provider. This is used for uploading the images and attachment files of DiscuzX to the object storage.
4. Your DiscuzX instance is online.

Configuration

First configure the Casdoor and Casnode correctly based on the their installation guides, make sure they are working normally before migration.

> TIP
>
> Let Casdoor and Casnode connect to the DB's Intranet URL. Let Casdoor's storage provider's endpoint be the Intranet URL of the cloud object storage. It will be much faster.

Configure the Casdoor database in Casnode's app.conf, so Casnode can directly connect to Casdoor's DB and create users. This will be faster than calling Casdoor's RESTful API to create users.

```
casdoorDbName = casdoor
```

Configure the migration script at: https://github.com/casbin/casnode/blob/master/discuzx/conf.go .

1. dbName: your DiscuzX's DB name
2. discuzxDomain: your DiscuzX's public domain, with trailing slash
3. discuzxAttachmentBaseUrl: your DiscuzX's attachment base URL, with trailing slash (you can get it from a attachment file URL of your DiscuzX)
4. avatarPoolBaseUrl: don't change this

```
package discuzx

var dbName = "ultrax"
```

> TIP
>
> In Casnode, we assume you use the same DB username and password for all 3 DBs: Casnode's DB, Casdoor's DB and DiscuzX's DB. So make sure this DB user can access all 3 DBs.

Migration

You may see the `XXXConcurrency` is defined at the top of the scripts, this is the number of the concurrent threads to do the migration. You can tune this value based on your environment. If it is too small, the migration will be very slow. If it is too high, the DB connections will be more likely to report "Connections too many" errors.

```
var SyncAvatarsConcurrency = 20
```

> TIP
>
> The estimation time we provide in the following sections is measured in a machine with remote Internet connection with the DB and object storage. If your VM is in the same subnet with the DB and object storage, you will be much faster.

1. User migration

To migrate all your DiscuzX's UCenter members to Casdoor's users:

Run `TestAddUsers` in: https://github.com/casbin/casnode/blob/master/discuzx/user_test.go

This step will roughly take 2 minutes for 60,000 users.

2. User avatar migration

To migrate all your DiscuzX's UCenter members' avatars to Casdoor's resources (via Casdoor's storage provider, backed by the cloud object storage):

Run `TestSyncAvatars` in: https://github.com/casbin/casnode/blob/master/discuzx/avatar_test.go

This step will roughly take 10 minutes for 60,000 users.

3. Forum migration

To migrate all your DiscuzX's forums to Casdoor's tabs and nodes:

Run `TestAddForums` in: https://github.com/casbin/casnode/blob/master/discuzx/forum_test.go

This step will usually finish in 2 seconds.

4. Thread and post migration

To migrate all your DiscuzX's threads & posts to Casdoor's topics & replies:

Run `TestAddThreads` in: https://github.com/casbin/casnode/blob/master/discuzx/thread_test.go

This step will roughly take 7 minutes for 10,000 users.

Finalization

During the migration, you can keep the Casnode instance running at the same time, so you can see the effect immediately by pressing `F5` in Casnode's public homepage.

If you encounter panic when running the migration scripts, contact the Casnode authors.

API reference

API Overview

Casnode API

API Overview

We use the swagger to record all of our API, you can see the api docs here.

Architecture

Overview

Casnode's architecture

Main package

Casnode main packages

Routers

Filters in Casnode

Controllers

Use controllers to handle requests

📄 Server Side Rendering

Server Side Rendering

Overview

Casnode is an open source project. An active community is its vitality.

This chapter is for developers who want to contribute to Casnode. Here you will learn the architecture of Casnode, and how it works.

Architecture

Casnode has 2 parts: the frontend and the backend.

Name	Description	Tools	Source code
Frontend	Web frontend UI for Casnode	JavaScript + React + Ant-Design	https://github.com/casbin/casnode/tree/master/web
Backend	RESTful API backend for Casnode	Golang + Beego + MySQL + Xorm	https://github.com/casbin/casnode

As we mentioned before, in product environments, the frontend of Casnode is built and served by the backend. In develop environments, the frontend is served by Nodejs.

Main package

There are several packages in the backend of Casnode. The main function and the Beego framework call these packages when the program starts.

main

The main package is the entrance of Casnode. We do some basic setup steps in the main package:

- **Setup some global variables**: Database adapter, http Client, OSS adapter, Segmenter and Forum basic info (forum version, online number, Google Groups Crawlers)
- **Setup filters:** API requests filter, search engine bot filter
- **Session data:** Casnode uses Beego session to store user info. In the main function, wrote these lines to use a file based session:

```
beego.BConfig.WebConfig.Session.SessionProvider = "file"
beego.BConfig.WebConfig.Session.SessionProviderConfig = "./tmp"
beego.BConfig.WebConfig.Session.SessionGCMaxLifetime = 3600 *
24 * 365
```

Please refer to Beego session if you want to use another session adapter in Beego.

Routers

Actually, `routers` is a built-in package of Beego. `routers.init()` is run by the framework when program starts. However, we added some filter functions to the package.

We are not going to talk about the `routers/router.go` here, Beego has already had a good document for it. We are going to introduce our filters here.

routers/filter.go

TransparentStatic: This filter is to serve static files. If a request path is not starts with `/api/`, then the filter will serve a static file to the browser. The filter will find the requested file in `web/build/`, and serve the file if exist.

FreshAccountActiveStatus: Update users' online states when they send a request.

routers/filter_ssr.go

We use Chromedp to render pages for search engine bots. If Chrome is installed, functions in this file will return a rendered page to bots.

We use a Regular expression `bot|slurp|bing|crawler` to match the User Agent of the request. If matched, we think the request is sent by a bot.

Controllers

In `routers/router.go`, you can find lots of lines like

```
beego.Router("/api/get-topics", &controllers.ApiController{},
"GET:GetTopics")
```

Please pay attention to the third parameter `"GET:GetTopics"`. This string is a combination of HTTP request method, and the function name of the request handler. You can find a function called `GetTopics` in the package `controllers`, and it is the handler of the request `/api/get-topics`.

You can find the corresponding function of the request this way.

> TIP
>
> If you are using **Goland**, you can press `Ctrl+Shift+F`(vscode use `Ctrl+F` instead), and search `func (c *ApiController) FunctionName` to locate to the function quickly.

Server Side Rendering

Casnode support SSR. SSR(Server-side rendering) is a popular technique for rendering a client-side single page application (SPA) on the server and then sending a fully rendered page to the client.

This allows for dynamic components to be served as static HTML markup.This approach can be useful for search engine optimization (SEO) when indexing does not handle JavaScript properly. It may also be beneficial in situations where downloading a large JavaScript bundle is impaired by a slow network.

Implementation

You can see the implementation of server side rendering in `routers/filter_ssr.go`.We use Chromedp to render pages for search engine bots. If Chrome is installed, functions in this file will return a rendered page to bots.

We use a Regular expression `bot|slurp|bing|crawler` to match the User Agent of the request. If matched, we think the request is sent by a bot.

Internationalization

Casnode supports multi-languages, by deploying the translations to Crowdin, we support Chinese, French, German, Russian, Japanese and Korean.

Casnode uses the official Crowdin cli to sync translations from Crowdin, if you want to add more languages supports, please propose in our community, and if you want to help us speed up the translating work, please help us translate on Crowdin.