User Guide and Reference Manual

Welcome to the Bot Libre User Guide and Reference Manual. This goal of this document is to provide an overview of the Bot Libre platform and its features.

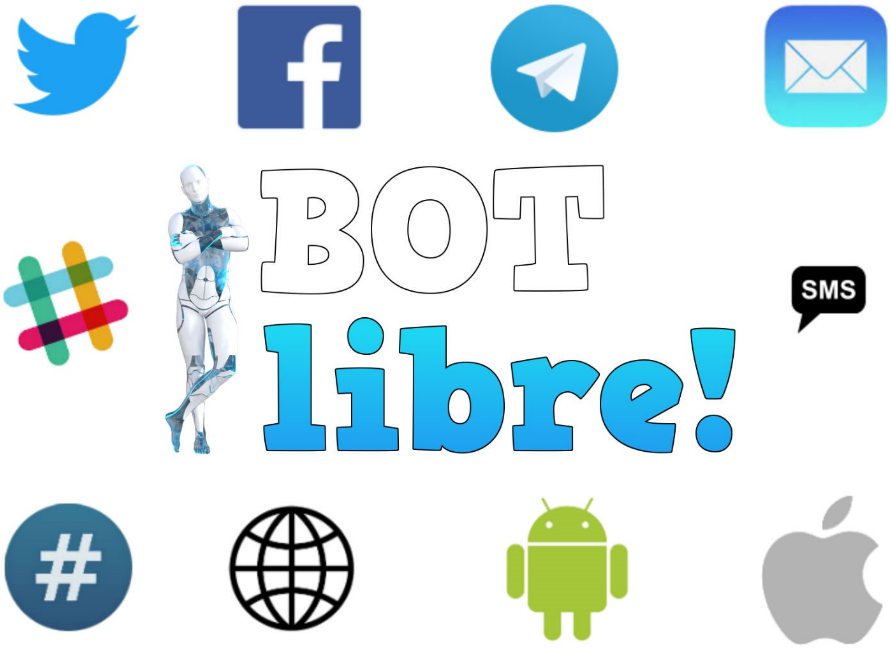


Table of Content

This user guide is split into several sections for Web, Mobile, Social Media, and Scripting.

- iOS
- SDK

Training & Scripting

- Response List
- Self
- AIML
- Regular Expressions (Regex)

Overview

About

Bot Libre is a free open platform for developing and hosting bots for the web, mobile, and social media. The Bot Libre platform also provides development and hosting for deep learning analytics, avatars, live chat, chat rooms, forums, scripts, and other content.

Bot Libre provides several components:

- Website - A website and web platform that lets you create and host bots and other content.
- Apps - Bot Libre provides several apps for Android and iOS , including bot, chat, and virtual assistant apps.
- SDK - The Bot Libre Software Development Kit (SDK) is an open source code library that you can use to develop your own website, mobile app, or desktop application.
- Web API - The Bot Libre web API provides an XML and JSON HTTP REST API that lets you access and integrate Bot Libre's services from any programming language or environment.
- AI Engine - The Bot Libre AI Engine is an open source project for bots and advanced artificial intelligence developed in Java and hosted on GitHub

Bot Libre is hosted by Paphus Solutions, a Canadian Corporation that specializes in intelligence automation products and services.Bot Libre is developed by Paphus, their partners, and the Bot Libre open source community.

Paphus Solutions also provides commercial bot hosting on Bot Libre for Business, as well as bot and app development services. Dedicated and private hosting is also available, and private licensing is provided through the Bot Libre Enterprise Bot Platform.

Architecture

Bot Libre provides a web based architecture that provides chat and bot services to many different clients. Bot Libre also implements many webhook APIs for integrating with other services, and can access web services and other online services.

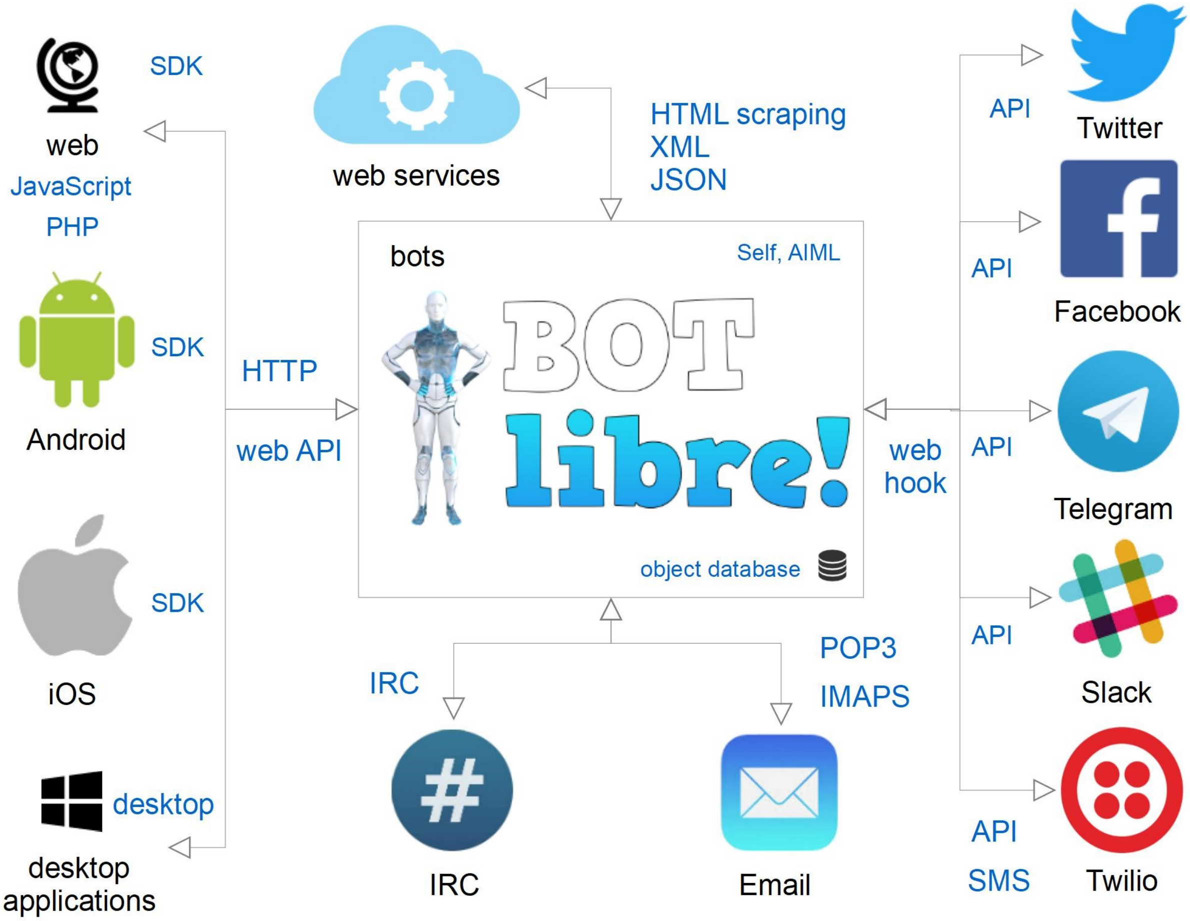
For the web, Bot Libre provides an embed tool that automatically generates the HTML and JavaScript code to embed your bot or live chat your website. Bot Libre also provides an open source JavaScript SDK that gives you complete control to customize your interface and user experience. Bot Libre provides a web API that enables you to access your bots and Bot Libre's services from any programming language and environment.

On mobile, Bot Libre provides an open source SDK for Android and iOS. The SDK includes example apps, and connection and GUI code to make building your own bot or chat app easy. The SDK provides virtual assistant commands for performing actions on phones and devices.

Bot Libre integrates with many different social media platforms include Twitter, Facebook, Telegram, Skype, Kik, WeChat, and Slack. Bot Libre implements their APIs and provides https webhooks to allow real-time messaging, you only need to authorise and connect your bot. Bot Libre also supports email automation, IRC, and SMS text messaging through Twilio.

Bot Libre bots can be trained without any programming, using text questions and answers, or learning from chat logs, live chat, or Twitter. Bot Libre supports tagging responses with keywords, topics, context, and other meta data to automatically match questions to responses. Bot Libre also supports patterns, templates, and scripting using Self (a JavaScript dialect), and AIML.

Bot Libre's scripting language Self allows access to XML and JSON web services, and HTML scraping. Bot Libre also provides several classes to access web services such as Google Calendar, WikiData, Wiktionary, Twitter, Facebook, Telegram, Email, Twilio, and more. Bot Libre bots each have their own object database to easily store data and information, and import JSON and XML data.

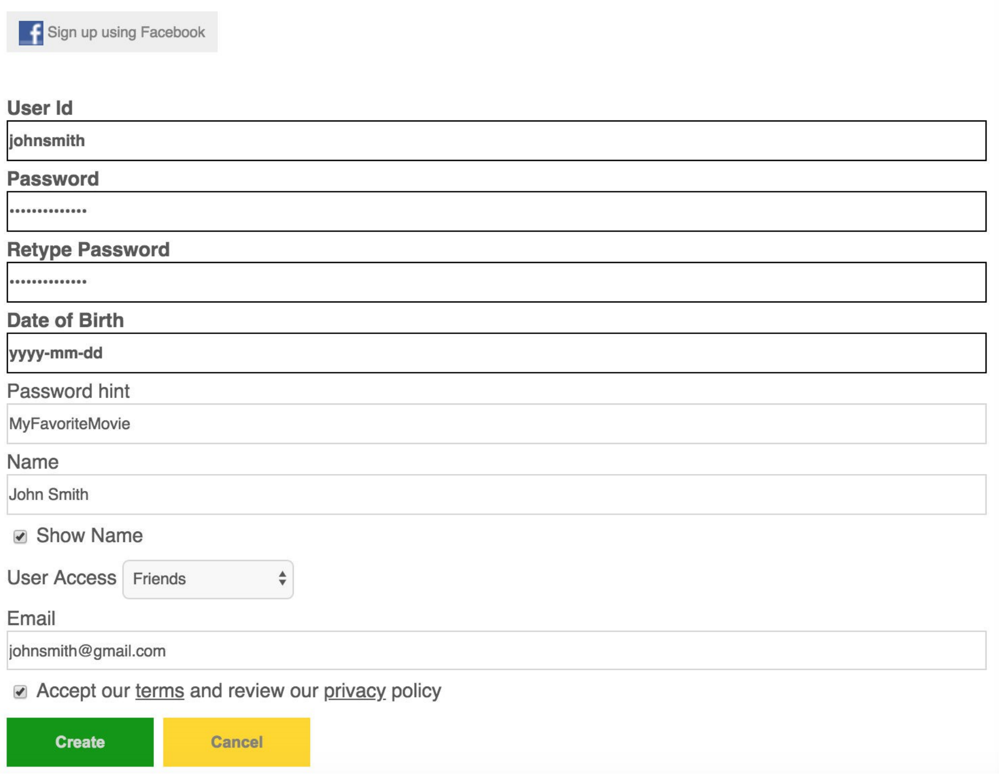

web services
HTML scraping
XML
JSON
SDK
web
JavaScript
PHP
Android
SDK
HTTP
web API
SDK
iOS
desktop
desktop applications
bots
Self, AIML
BOT
libre!
object database
web hook
API
Twitter
API
Facebook
API
Telegram
API
Slack
API
SMS
Twilio
IRC
IRC
POP3
IMAPS
Email

Sign Up

Overview

You can browse and chat with public bots anonymously, but to create your own bot you must create an account by clicking Sign Up. Creating an account is free and easy. To create an account you only need to enter a unique user ID and a password. Your name and email are optional, if you wish to remain anonymous, you may do so.

Your user id must be a unique name such as jim2017, do not user an email address, spaces, or special characters. It is recommended to enter an email address, then you can sign in also using the email address, and can reset your password if required.



User Id: johnsmith

Password: ••••••••••••••

Retype Password: ••••••••••••••

Date of Birth: yyyy-mm-dd

Password hint: MyFavoriteMovie

Name: John Smith

☑ Show Name

User Access: Friends

Email: johnsmith@gmail.com

☑ Accept our terms and review our privacy policy

Properties

Properties	Required	Description
User Id	☑	Enter a unique user id (no spaces, alpha numeric, visible to other users).
Password	☑	Enter a secure password.
Retype Password	☑	Confirm password.
Password Hint	☐	Enter a personal hint in case you forget your password.
Name	☐	Enter your real name (visible to other users if 'Show Name' is selected).
Email	☐	Enter your email addrress (not visible to other users, recommended, required for password reset).
Website	☐	Enter your business or personal website (visible to other users).
Bio	☐	Enter anything about you (HTML, visible to other users).
Accept Terms	☑	You must accept our terms of service.

Facebook

- You can also sign up using your Facebook account.

See Also

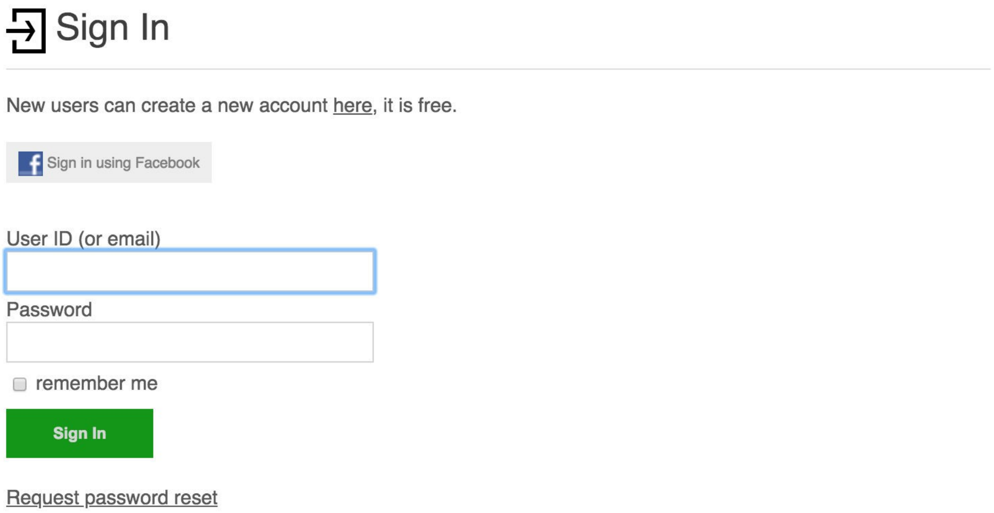
- How to create your own chat bot in 10 Clicks

Sign in

Overview

You can sign in from the home page, if you don't have an account you can create one, it is free, and only requires a user id/password.

You can sign in using your user id, or your email address if you entered one when signing up. If you forgot your password you will be given a password reset link after one failed sign in attempt. You must have entered an email address when signing up to be able to reset your password.



Sign In

New users can create a new account here, it is free.

User ID (or email)

Password

☐ remember me

Request password reset

Properties

Properties	Required	Description
User Id(or email)	☑	Enter a unique user id (no spaces, alpha numeric, visible to other users).
Password	☑	Enter a secure password.

Facebook

- You can also sign in using your Facebook account.

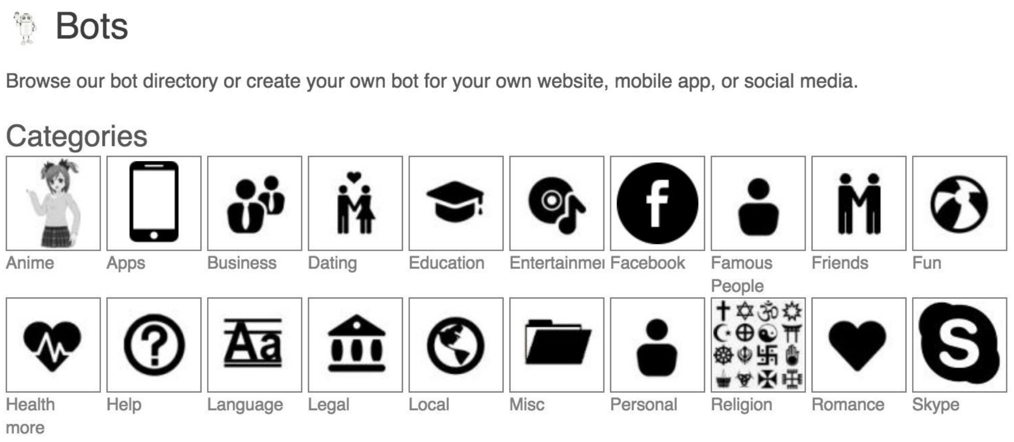
See Also

- How to create your own chat bot in 10 Clicks

Browsing

Overview

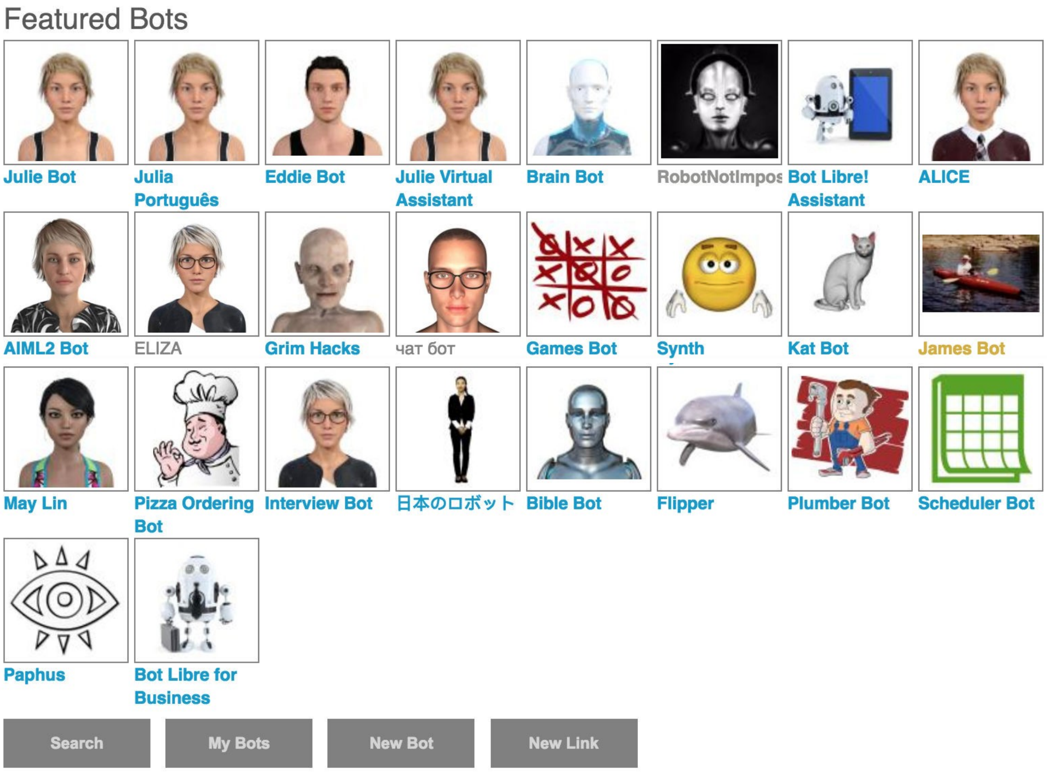
You can browse the set of bots and categories from the Browse menu.
You can create your own chat bot from the 'New Bot' menu, or browse all published bots.
To create a new bot you must first sign in.



Bots

Browse our bot directory or create your own bot for your own website, mobile app, or social media.

Featured Bots



Properties

Properties	Description
Categories	A category lets you classify your bot.
Featured	View featured and demostration bots.
New Category	Create a new category.
Search	Searching by categories, tags, name, display, restrict, and sort.
New Bot	Create your own bot.
New Link	Add a link to an external bot or website to the bot directory.

See Also

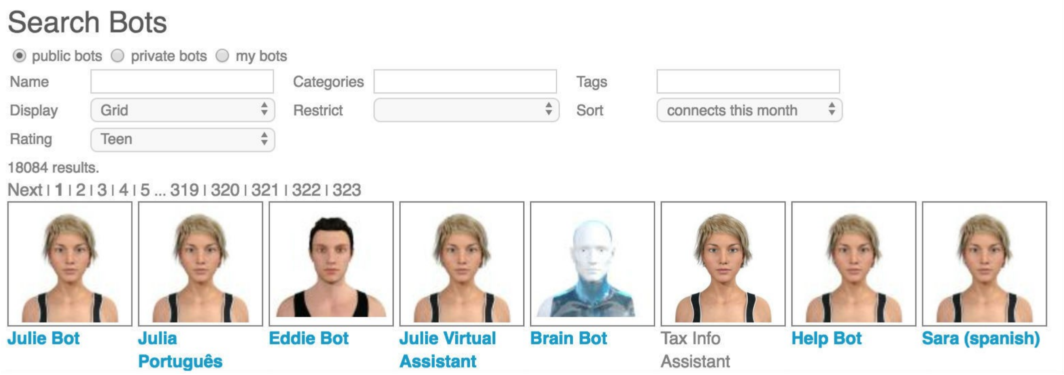
- Browse Bots

Searching

Overview

You can search the open bot directory from the Search menu, or search the website from the Google site search link in the top menu or footer.

Only public bots are listed in our browse directory. If your bot is private, you can find it from selecting 'my bots' or 'private' from search. You can also mark your bot as hidden to have it not searchable.



Search Bots

● public bots ○ private bots ○ my bots

Name:
Categories:
Tags:
Display: Grid
Restrict:
Sort: connects this month
Rating: Teen

18084 results.

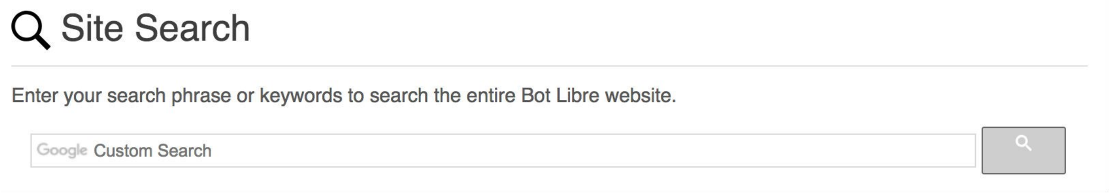
Next | 1 | 2 | 3 | 4 | 5 ... 319 | 320 | 321 | 322 | 323

Properties

Properties	Description
Public Bots	Show all public bots created by all users.
Private Bots	Show all private bots this user has access to.
My Bots	Show all bots you are the administrator for.
Categories	Filter by a comma separated list of category names.
Tags	Filter by a comma separated list of tag names.
Name	Filter by any name containing the text.

Site Search

The Site Search uses Google site search to search all webpages in the website.



Site Search

Enter your search phrase or keywords to search the entire Bot Libre website.

See Also

- Search Bots
- Site Search

Creating

Overview

To create a new bot click Create from the top banner. You can give your bot a name and description. The name must be unique and can contain spaces, special characters, and is case sensitive.

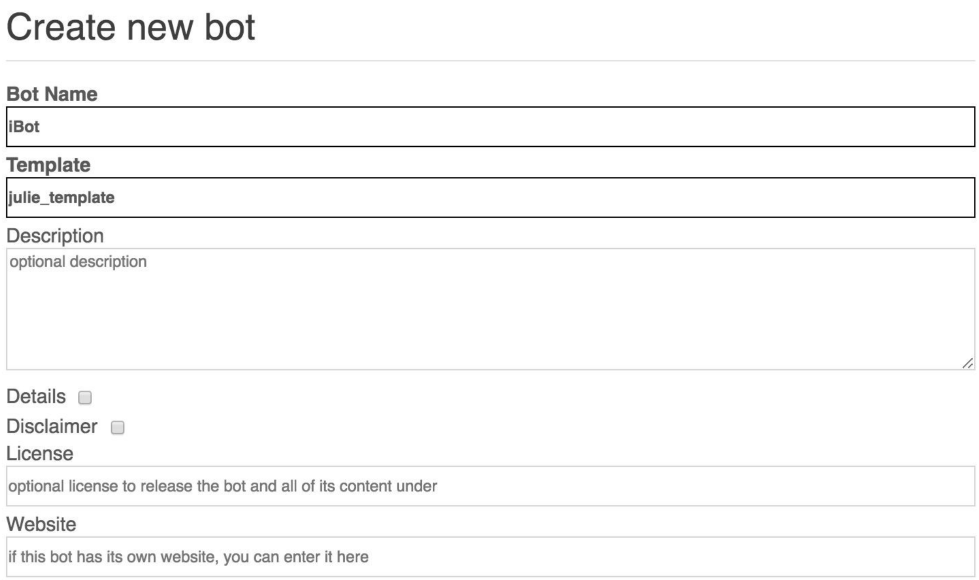
The bots you create on Bot Libre belong to you. When you create a bot you can specify the license you wish to share it under. If you allow other users to 'fork' your bot, you release your bots content to them under your bot's license. Also, if your bot learns from the users it interacts with, they must understand they are releasing their interactions under your bot's license.

You can create a bot from the default template, or select one of the other predefined templates. Your bot will start as a clone of the template bot, and replicate its entire memory (brain). You can also allow your bot to be 'forked' and used as a template to other bots, or you can browse other public bots and choose to fork your bot from them.

You can add tags to your bot, to help classify it, and help other users find it. Some relevant tags include 'fun', 'business', 'help', and 'avatar'. The tags are entered as a comma separated list.

You can create either a public bot that will be accessible by other users, or a private bot that will be accessible only by you. You can control the users that can access and administer your bot.

You can choose if you want your bot to learn as it interacts with other users. If you disable learning, then your bot will only learn from its administrator when you 'correct' it, from chat logs you upload from the 'Chat Logs' page under 'Admin', or from AIML and Self scripts you add from the 'Scripts' page. Caution should be used in allowing learning, as other users may teach your bot offensive responses. Bot Libre provides a profanity filter, but it cannot catch everything. If you are creating a bot for business, such as for customer service, then you should disable learning.



Create new bot

Bot Name
iBot

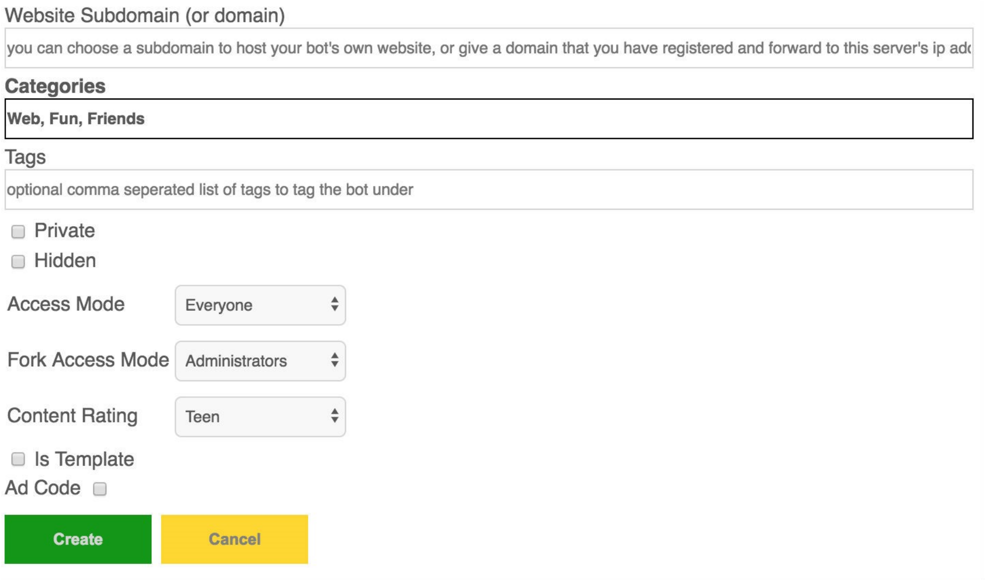
Template
julie_template

Description
optional description

Details □
Disclaimer □

License
optional license to release the bot and all of its content under

Website
if this bot has its own website, you can enter it here



Website Subdomain (or domain)

you can choose a subdomain to host your bot's own website, or give a domain that you have registered and forward to this server's ip add

Categories

Web, Fun, Friends

Tags

optional comma seperated list of tags to tag the bot under

□ Private

□ Hidden

Access Mode: Everyone

Fork Access Mode: Administrators

Content Rating: Teen

□ Is Template

Ad Code □

Properties

Properties	Description
Bot Name	Enter a bot name.
Template	Type of bot to create.
Description	Optional description.
Details	You can enter optional additional information.
Disclaimer	You can enter optional legal information.
License	Optional license to release the bot and all of its content under.
Website	If this bot has its own website, you can enter it here.
Website Subdomain (or domain)	You can choose a subdomain to host your bot's own website, or give a domain that you have registered and forward to this server.
Categories	Comma separated list of categories to categorize the bot under.
Tags	Optional comma separated list of tags to tag the bot under.
Private	A private bot is not visible to the public, only to the user and users grant access.
Hidden	A hidden bot is not displayed in the browse directory.
Access Mode	Define who can access this bot.
Fork Access Mode	Define who can fork (copy) this bot.
Content Rating	Rate the bot.
Ad Code	You can display ads on your bot's pages.

See Also

- How to create your own chat bot in 10 clicks

Bots

Overview

A **chat bot**, or **chatbot** is a software program that responds to questions in natural language (such as English, French, etc.).

Chat bots can be used for many purposes. Chat bots can be used for a business to help users, provider customer service, or promote a product. Chat bots can be used for fun, as a friend to chat with, even a girlfriend or boyfriend. Chats bot can also be used for research, knowledge and education.

There are many different types of chat bots, such as:

- Virtual Agents
- Chatterbots
- Knowledge bots
- Twitterbots
- AI bots

Most chat bots are web based, but there are also chat bots that run on Twitter, email, Facebook, Telegram, Slack, Skype, ICQ, IM, Second Life, and a growing number of Android, iOS and mobile chat bots.

Bot Libre's goal is to support chat bots running on as many different services as we can support. We currently have integrated support for connecting your bot to the web, Twitter, Facebook, Telegram, Slack, SMS, email, IRC, Android, iOS, and other platforms through our apps, API and SDK.

Chat bots can be developed using many different techniques and programming models.

Some common techniques include:

- Question/response matching
- Keyword and topic matching
- Sequential scripting
- Text search and processing (such as AIML)
- Language parsing and state machines (such as Self)
- Direct programming (PHP, Java, LISP)
- Machine learning (artificial intelligence)

Bot Libre understands that no single method or technique is best for all types of bots, or all types of situation. Bot Libre provides a heterogeneous environment of different techniques including, automatic question/responses matching, keyword and topic matching, AIML, Self scripting, and machine learning and comprehension.

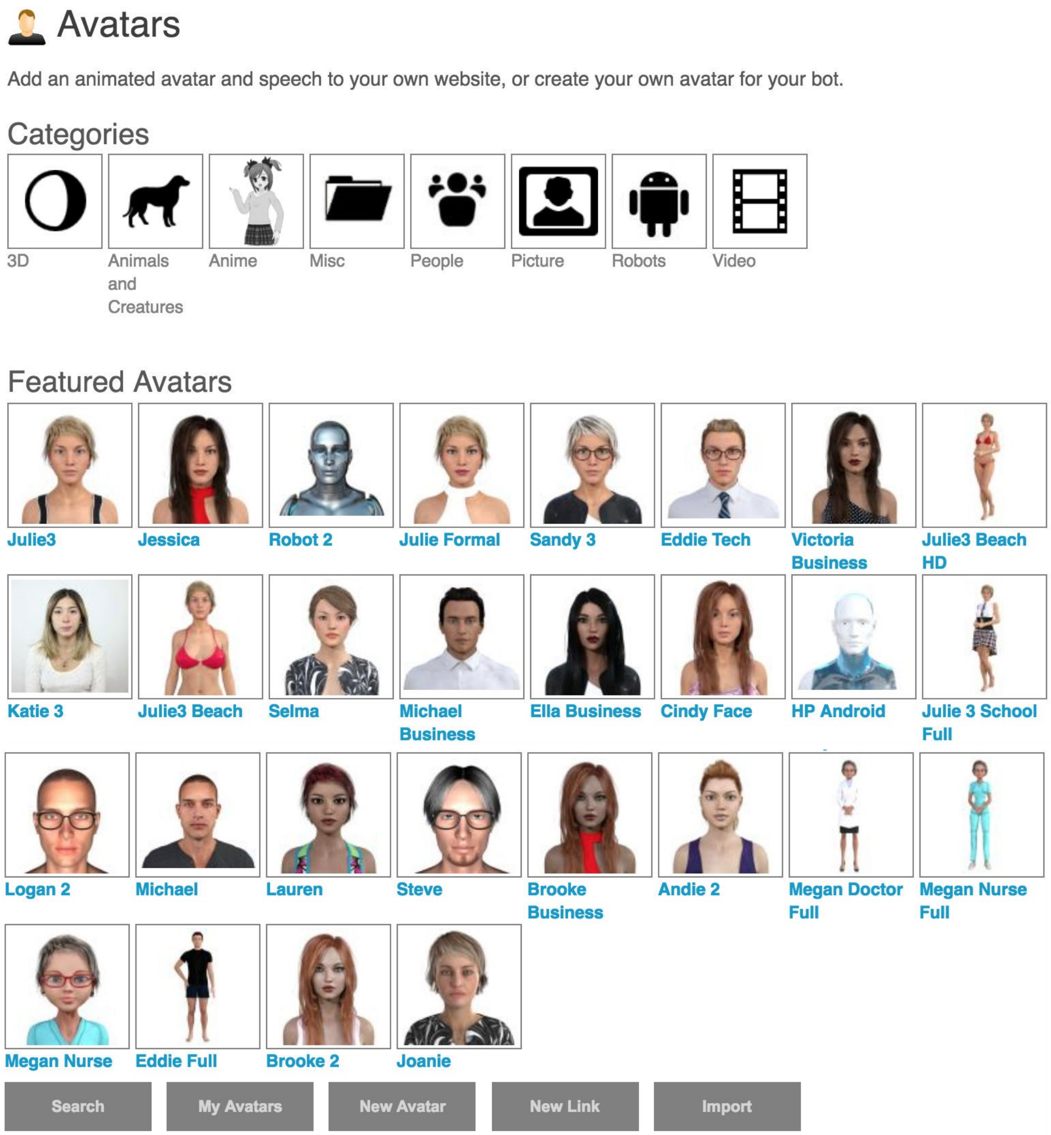
See Also

- How to create your own chat bot in 10 Clicks

Avatars

Overview

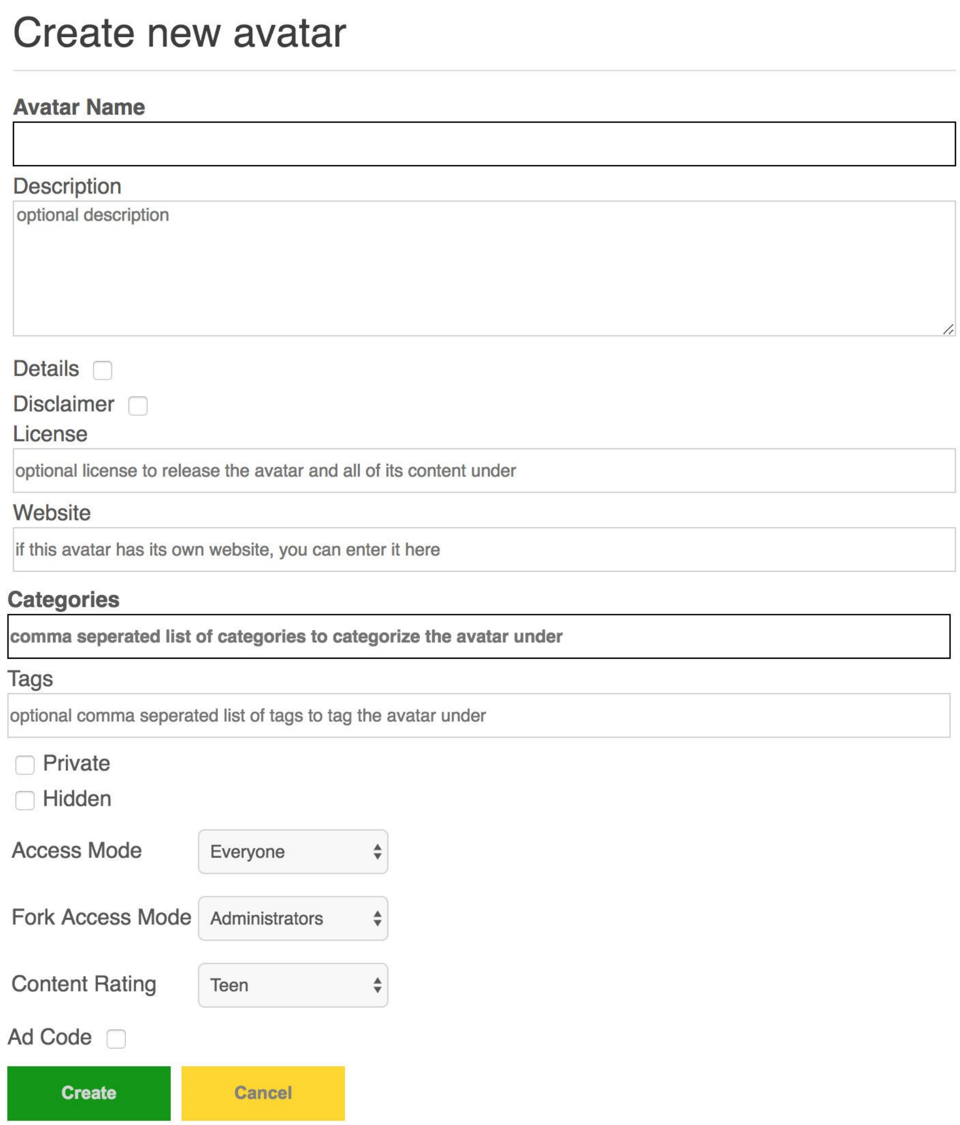
The avatar tab lets you choose your bot's avatar. An avatar is the physical representation of your bot, and can include images, video, audio, and animation. You can choose an avatar from our open avatar directory, or create your own.



Add an animated avatar and speech to your own website, or create your own avatar for your bot.

Creating

You can create your own avatar from the browse avatars page. To create an avatar, just give it a name, description, and category to categorize it under. You can either make the avatar private and choose who can access it, or make it public and accessible by anyone. You can also choose what license you wish to release its content under.



Create new avatar

Avatar Name

Description

optional description

Details □

Disclaimer □

License

optional license to release the avatar and all of its content under

Website

if this avatar has its own website, you can enter it here

Categories

comma seperated list of categories to categorize the avatar under

Tags

optional comma seperated list of tags to tag the avatar under

□ Private

□ Hidden

Access Mode: Everyone

Fork Access Mode: Administrators

Content Rating: Teen

Ad Code □

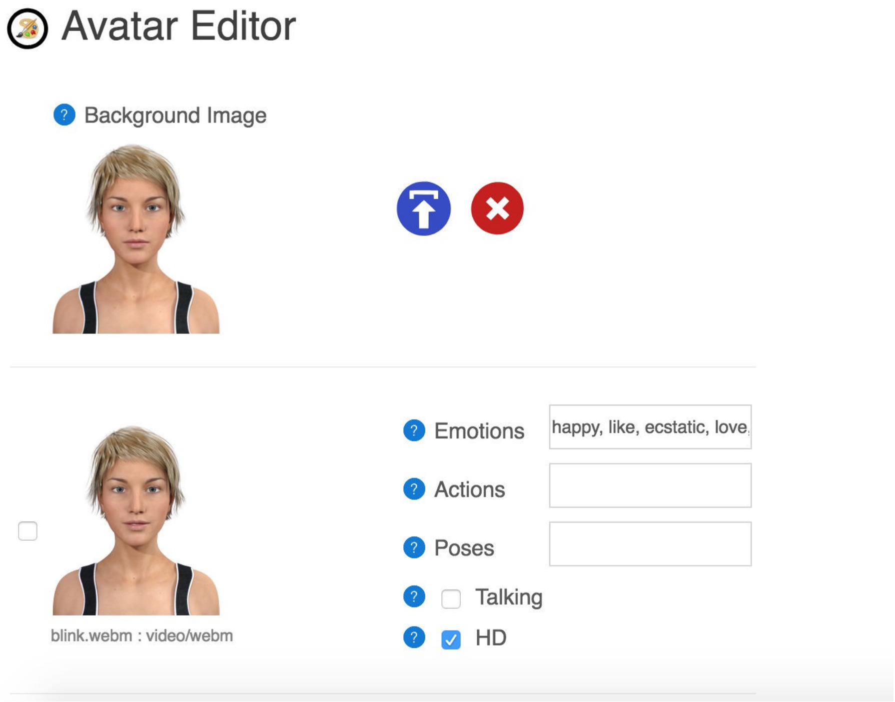
Properties

Properties	Description
Avatar Name	Enter an Avatar name.
Description	Optional description.
Details	You can enter optional additional information.
Disclaimer	You can enter optional legal information.
License	Optional license to release the avatar and all of its content under.
Website	If this avatar has its own website, you can enter it here.
Categories	Comma separated list of categories to categorize the avatar under.
Tags	Optional comma separated list of tags to tag the avatar under.
Private	A private avatar is not visible to the public, only to the user and users grant access.
Hidden	A hidden avatar is not displayed in the browse directory.
Access Mode	Define who can access this avatar.
Fork Access Mode	Define who can fork(copy) this avatar.
Content Rating	Rate the avatar.
Ad Code	You can display ads on your avatar's pages.

Avatar Editor

The avatar editor allows you to upload or import images, video, and audio files for your avatar, and tag them with emotions, actions, and poses. You can add as many images, videos, or audio as you want, and they will be randomly selected if not tagged. If tagged, they will only be used when the bot expresses the tagged emotion, action, or pose. You should have at least a 'talking' pose tagged, this will be displayed when the bot is talking.

To edit the avatar, click on the Editor button. The avatar editor lets you import video, audio, and image files. You can associate each media with an emotion, action, or pose. For example if you have a video that makes the avatar talk, you would associate the pose for the video to talking. The emotions let the avatar express different emotions, and will let the bot using the avatar express itself. There are a fixed set of emotions, but you can create any actions and poses that you wish. You can have multiple media for a emotions, action, or pose, and one will be chosen at random. For actions, and poses, you can have an audio element in addition to a video element. Audio can be used to make your avatar perform actions such as laugh, or used as background music or noise for a pose.



Avatar Editor

? Background Image

□

? Emotions happy, like, ecstatic, love

? Actions

? Poses

? □ Talking

? ☑ HD

blink.webm : video/webm

Toolbar Menu Section

Icon	Property Name	Description
💾	Save	Save the current changes.
☑	Select	Select all media.
✖	Delete	Delete the selected media.
⬆	Upload	Upload images, video, or sound files for the avatar.

Toolbar Background Section

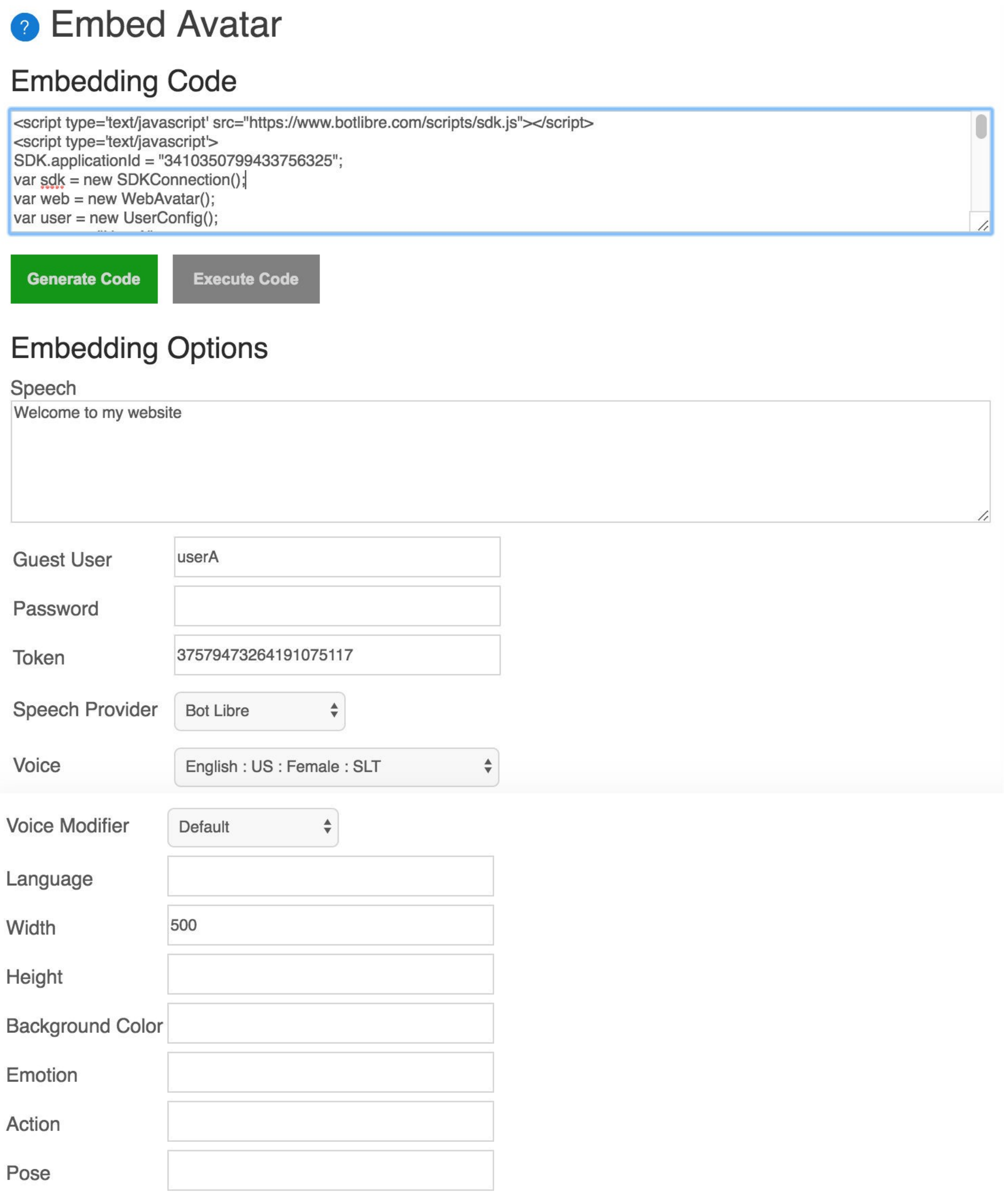
Icon	Property Name	Description
⬆	Upload	Upload a background image to overlay video on (this can reduce the video flicker).
✖	Clear	Clear the background image.

Media Section

Property Name	Description
Emotions	Comma separated list of emotions to trigger the media.
Actions	Comma separated list of actions to trigger the media.
Poses	Comma separated list of poses to trigger the media.

Embed Avatar

The avatar embed page allows you test the avatar and generate embedding code to add the avatar to your own website.



Embed Avatar

Embedding Code

```
<script type='text/javascript' src="https://www.botlibre.com/scripts/sdk.js"></script>
<script type='text/javascript'>
SDK.applicationId = "3410350799433756325";
var sdk = new SDKConnection();
var web = new WebAvatar();
var user = new UserConfig();
```

Embedding Options

Speech

Welcome to my website

Guest User	userA
Password	
Token	37579473264191075117
Speech Provider	Bot Libre
Voice	English : US : Female : SLT
Voice Modifier	Default
Language	
Width	500
Height	
Background Color	
Emotion	
Action	
Pose	

Properties

Properties	Description
Speech	The text for the avatar to speak.
Guest User	Guest user to connect as, you need to add this user to your avatar's users (do not use your own user id).
Password	Password for guest user (not secure).
Token	Token for guest user (this will expire, use a guest user/password instead).

Properties	Description
Server Voice	More than 30 server voices.
Voice Modifier	Several voice modifier.
Native Voice	The name of the native voice. A native browser voice can be used on Chrome or browses that support the HTML Speech API.
Responsive Voice	Use the third party voice API from ResponsiveVoice.
Language	The language code for the native voice.
Width	The width (in pixels) of the avatar box.
Height	The height (in pixels) of the avatar box.
Background Color	The background color to use.
Emotion	Emotion for avatar to express.
Action	Action for avatar to perform.
Pose	Pose for avatar to hold.

See Also

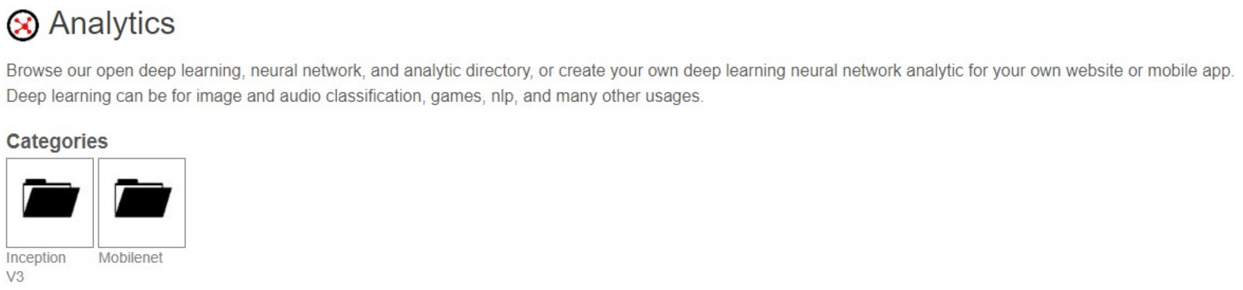
- How to add a 3D avatar to your website or blog

Deep Learning & Analytics

Overview

An analytic is a neural network, deep learning model, or analytical model that can be used to predict or classfiy data such as images, audio, text, and more.

The analytics library provides many open source examples of analytics for image classifcation and other usages. You can create your own analytic and upload a binary network file such as a python .pb file that represents your model.



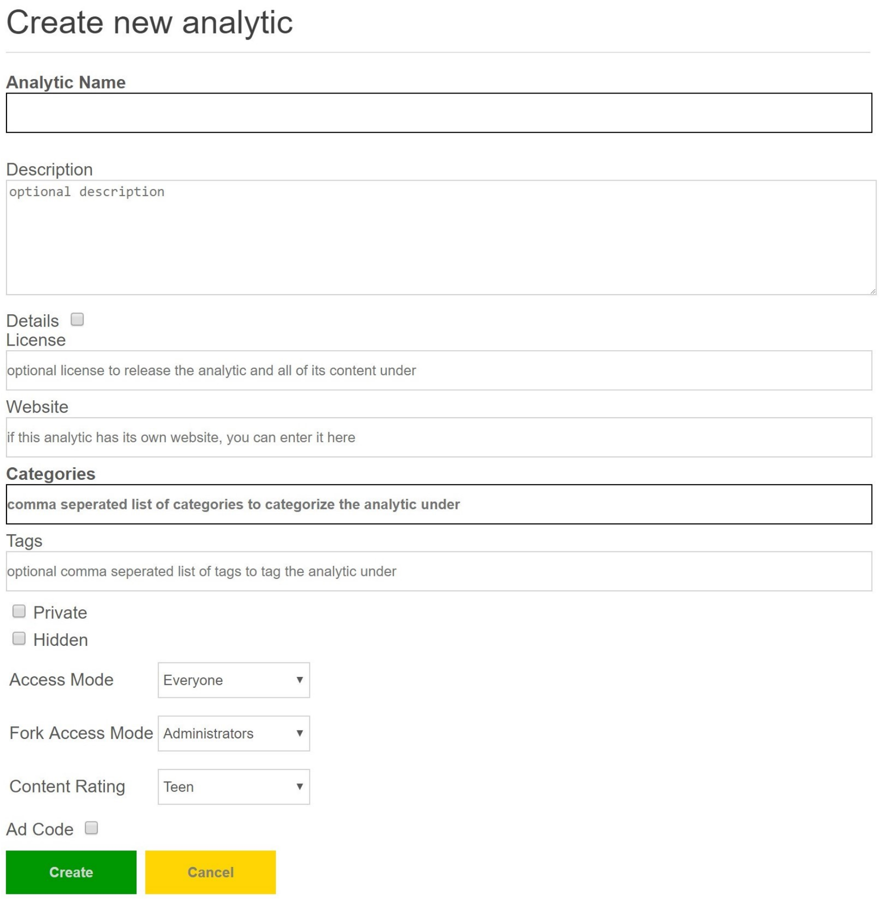
Analytics

Browse our open deep learning, neural network, and analytic directory, or create your own deep learning neural network analytic for your own website or mobile app. Deep learning can be for image and audio classification, games, nlp, and many other usages.

Categories

Creating

You can create your own analytic from the browse analytics page. To create a analytic, just give it a name, description, and category to categorize it under. You can either make the analytic private and choose who can access it, or make it public and accessible by anyone.



Create new analytic

Analytic Name

Description: optional description

Details □

License: optional license to release the analytic and all of its content under

Website: if this analytic has its own website, you can enter it here

Categories: comma seperated list of categories to categorize the analytic under

Tags: optional comma seperated list of tags to tag the analytic under

□ Private

□ Hidden

Access Mode: Everyone

Fork Access Mode: Administrators

Content Rating: Teen

Ad Code □

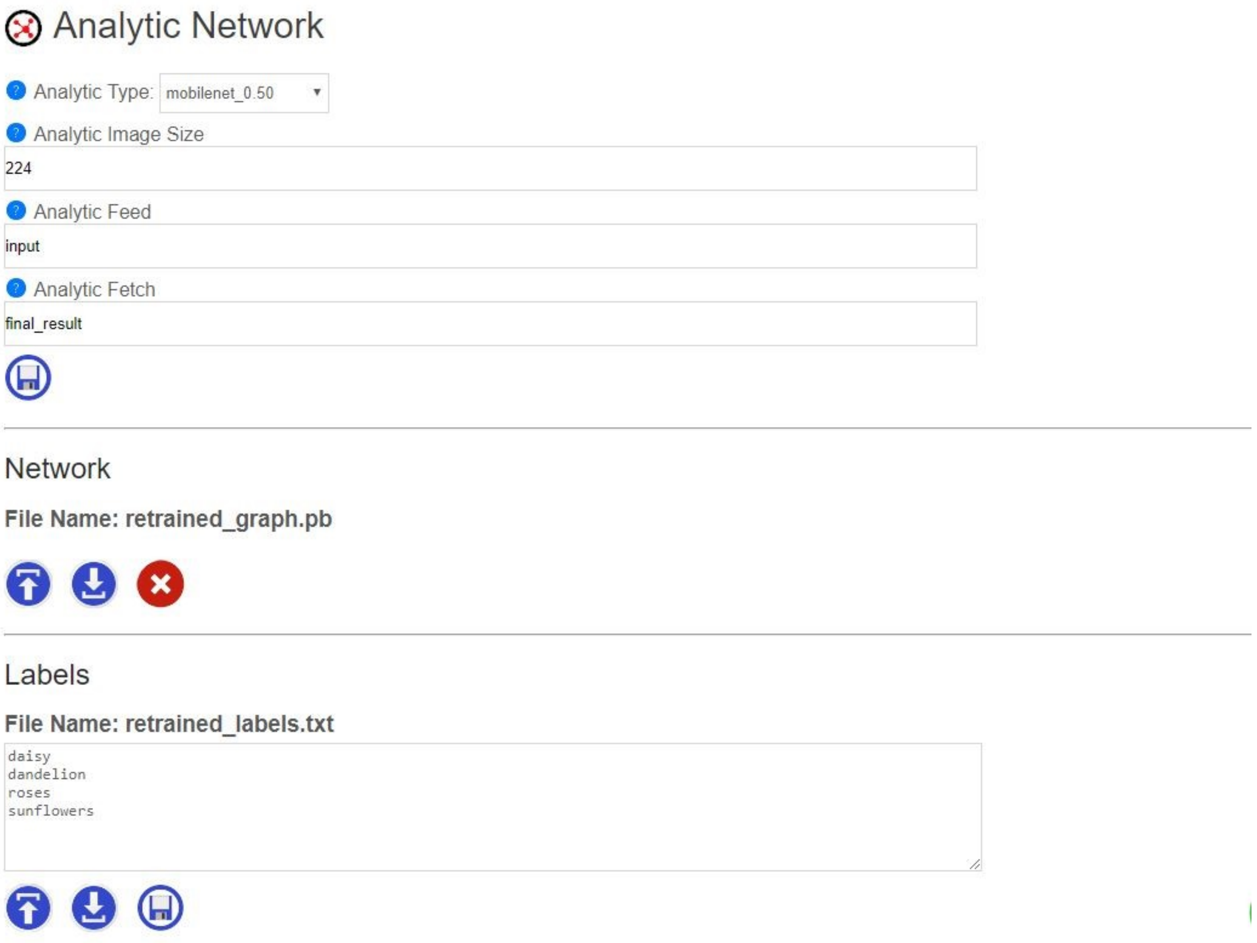
Properties

Properties	Description

Properties	Description
Analytic Name	Enter a analytic name.
Description	Optional description.
Details	You can enter optional additional information.
Disclaimer	You can enter optional legal information.
License	Optional license to release the analytic and all of its content under.
Website	If this analytic has its own website, you can enter it here.
Website Subdomain (or domain)	You can choose a subdomain to host your analytic's own website, or give a domain that you have registered and forward to this server.
Categories	Comma separated list of categories to categorize the analytic under.
Tags	Optional comma separated list of tags to tag the analytic under.
Private	A private analytic is not visible to the public, only to the user and users grant access.
Hidden	A hidden analytic is not displayed in the browse directory.
Access Mode	Define who can access this analytic.
Fork Access Mode	Define who can fork(copy) this analytic.
Content Rating	Rate the analytic.
Ad Code	You can display ads on your analytic's pages.

Analytic Network

The analytic network editor allows you to configure and to upload or import the required files for both graph and labels for your analytic. Currently the analytic network binary file must be a python Tensorflow .pb file. Support for other network types will be added in the future. The labels file is a text file that matches the binary network's output.



Analytic Network

Analytic Type: mobilenet_0.50

Analytic Image Size: 224

Analytic Feed: input

Analytic Fetch: final_result

Network

File Name: retrained_graph.pb

Labels

File Name: retrained_labels.txt

```
daisy
dandelion
roses
sunflowers
```

Network

Properties	Description
Analytic Type	Types of analytic (inception_v3, mobilenet_0.25, mobilenet_0.50, mobilenet_0.75, mobilenet_1.0)
Analytic Image Size	Set the image size of the trained images
Analytic Feed	The input name used while training the images
Analytic Fetch	The output name used while training the images

Network (Buttons)

Icon	Property Name	Description
💾	Save	Save the current changes.
⤒	Upload	Upload graph .pb file.
⤓	Download	Download the current uploaded graph.
✖	Delete	Delete the uploaded graph.

Labels

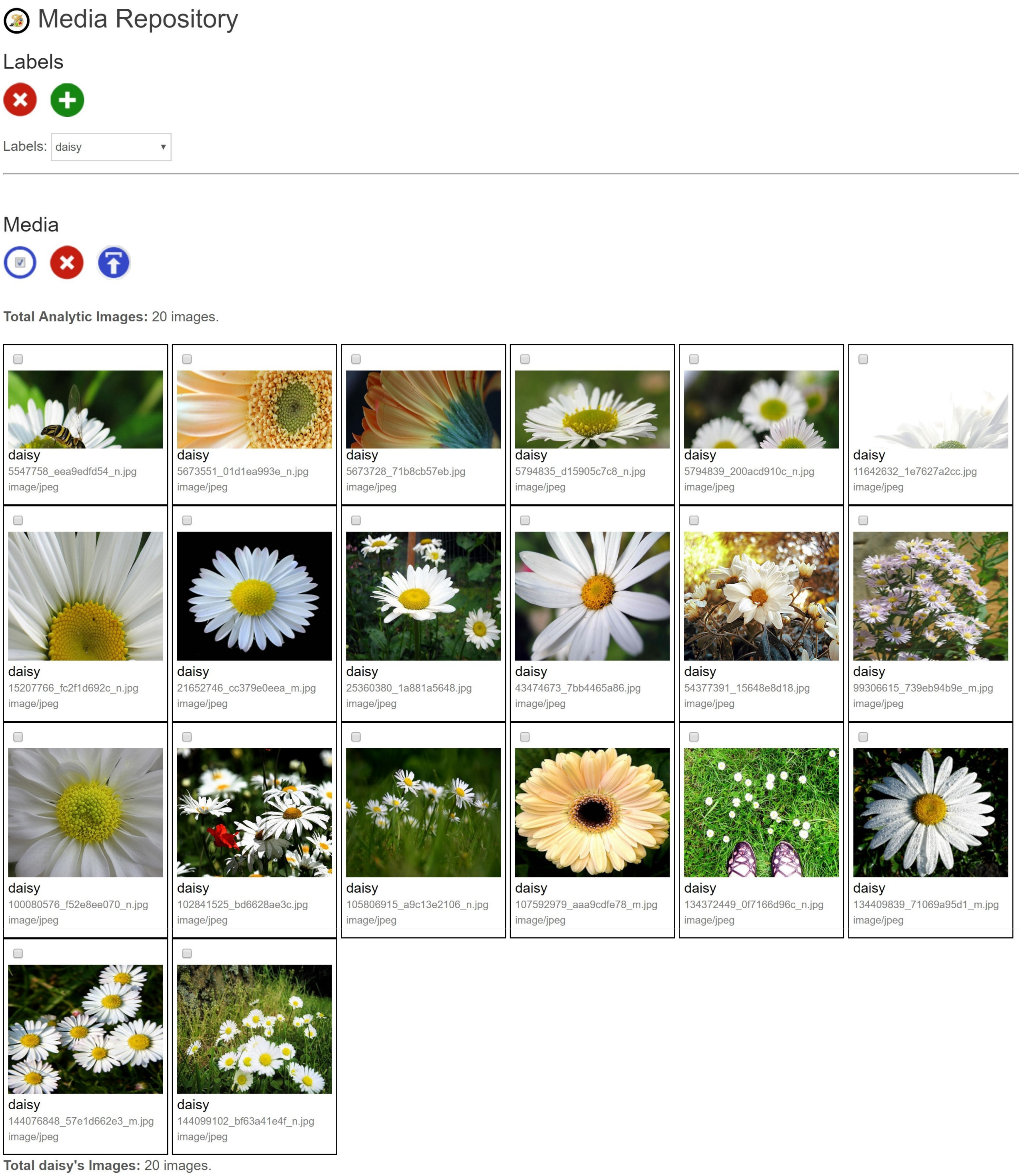
Properties	Description
Text Area	Displaying the labels saved in the uploaded text file.

Labels (Buttons)

Icon	Property Name	Description
💾	Save	Save the current changes.
⤒	Upload	Upload graph .pb file.
⤓	Download	Download the current uploaded graph.

Analytic Repository

The Analytic Repository allows you to add a label and import images inside one of each label labels.



Media Repository

Labels

Labels: daisy ▾

Media

Total Analytic Images: 20 images.

□ daisy
5547758_eea9edfd54_n.jpg
image/jpeg

□ daisy
5673551_01d1ea993e_n.jpg
image/jpeg

□ daisy
5673728_71b8cb57eb.jpg
image/jpeg

□ daisy
5794835_d15905c7c8_n.jpg
image/jpeg

□ daisy
5794839_200acd910c_n.jpg
image/jpeg

□ daisy
11642632_1e7627a2cc.jpg
image/jpeg

□ daisy
15207766_fc2f1d692c_n.jpg
image/jpeg

□ daisy
21652746_cc379e0eea_m.jpg
image/jpeg

□ daisy
25360380_1a881a5648.jpg
image/jpeg

□ daisy
43474673_7bb4465a86.jpg
image/jpeg

□ daisy
54377391_15648e8d18.jpg
image/jpeg

□ daisy
99306615_739eb94b9e_m.jpg
image/jpeg

□ daisy
100080576_f52e8ee070_n.jpg
image/jpeg

□ daisy
102841525_bd6628ae3c.jpg
image/jpeg

□ daisy
105806915_a9c13e2106_n.jpg
image/jpeg

□ daisy
107592979_aaa9cdfe78_m.jpg
image/jpeg

□ daisy
134372449_0f7166d96c_n.jpg
image/jpeg

□ daisy
134409839_71069a95d1_m.jpg
image/jpeg

□ daisy
144076848_57e1d662e3_m.jpg
image/jpeg

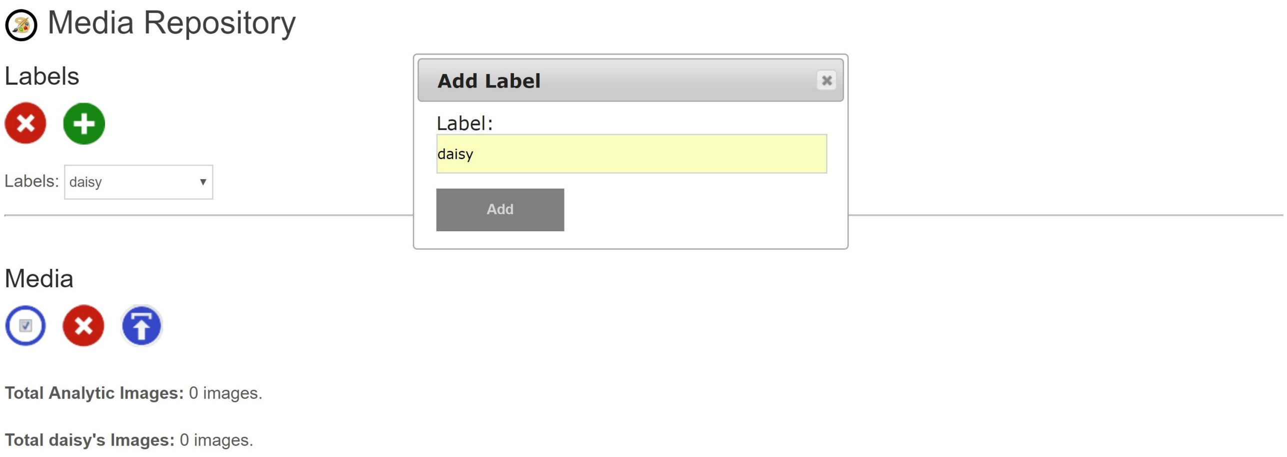
□ daisy
144099102_bf63a41e4f_n.jpg
image/jpeg

Total daisy's Images: 20 images.

Labels

Icon	Property Name	Description
✖	Delete	Delete the selected label.
✚	Add	Add a label

Properties	Description
Labels	Select a label



Media Repository

Labels

Labels: daisy

Add Label

Label: daisy

Media

Total Analytic Images: 0 images.

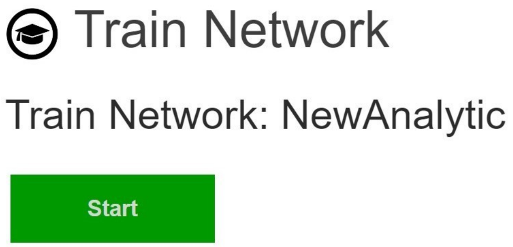
Total daisy's Images: 0 images.

Add Label

Properties	Description
Add label	Add a label to the list

Analytic Training

Training images.



Train Network

Train Network: NewAnalytic

Training Network

Properties	Description
Start	Training will start based on availability. This might take several hours.

See Also

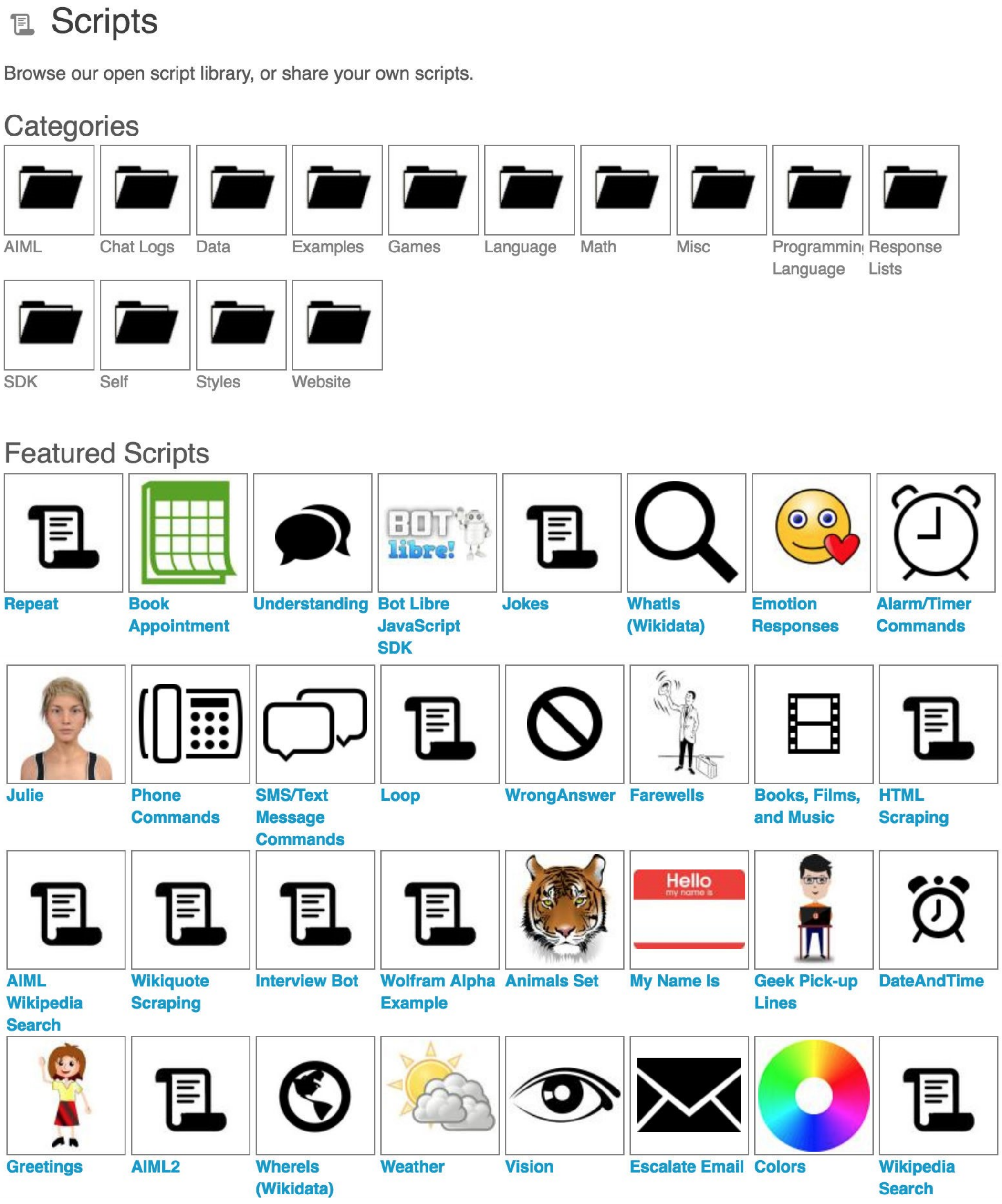
- How to create your own deep learning neural network for image recognition without any programming
- Tensorflow
- Training your images

Scripts

Overview

A script is a program file that can be loaded into your bot to give it new responses or functions. Bot Libre support scripts written in Self and AIML, as well as response list files.

The Bot Libre script library provides many open source examples scripts and data files in several different languages and formats. The Bot Libre script library supports uploading scripts written in any language. You can even upload HTML, JavaScript, and css files to host a website on Bot Libre.



Scripts

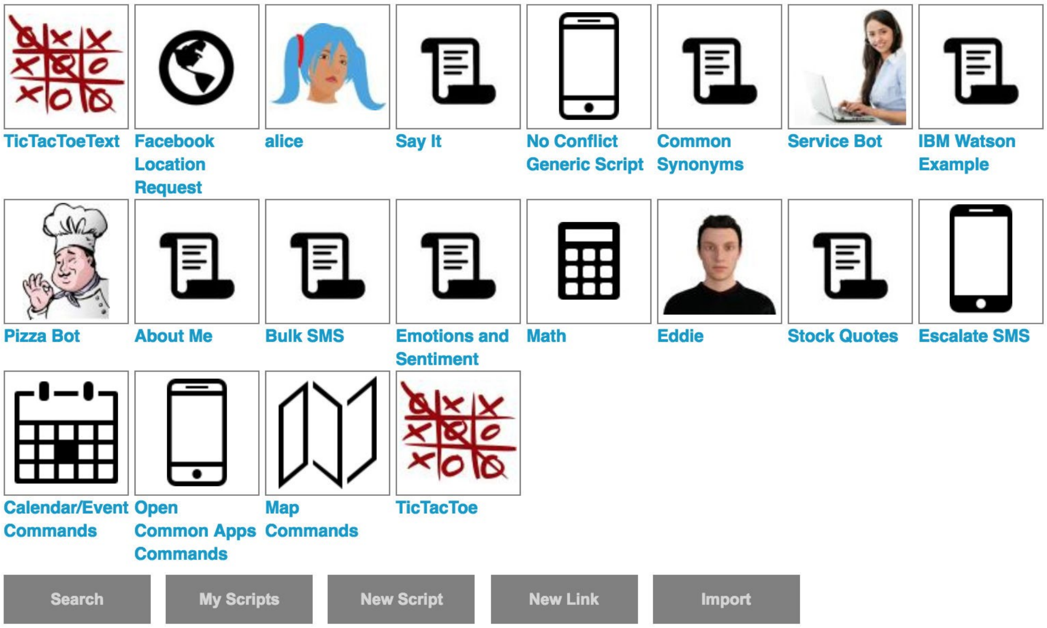
Browse our open script library, or share your own scripts.

Categories

AIML
Chat Logs
Data
Examples
Games
Language
Math
Misc
Programmin Language
Response Lists
SDK
Self
Styles
Website

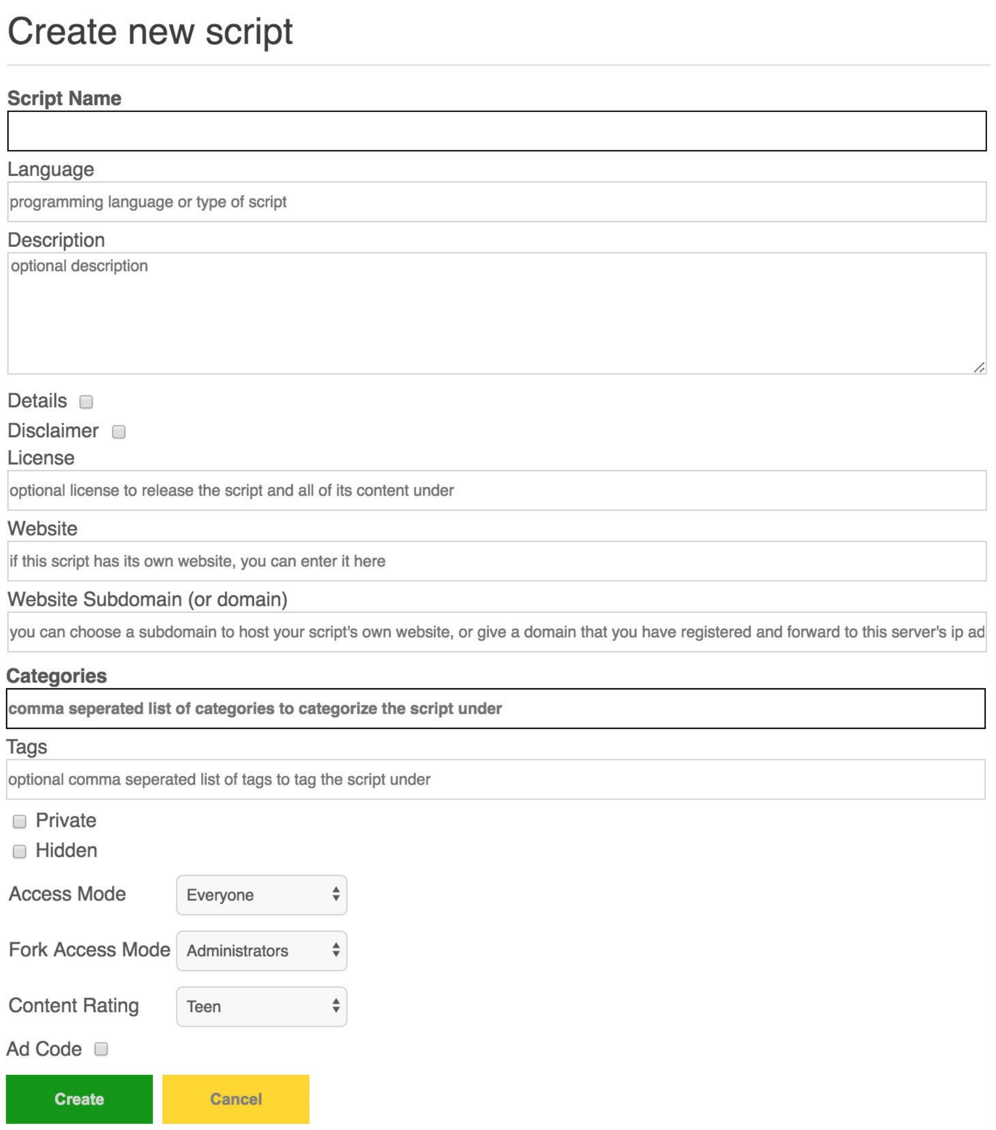
Featured Scripts

Repeat
Book Appointment
Understanding
Bot Libre JavaScript SDK
Jokes
WhatIs (Wikidata)
Emotion Responses
Alarm/Timer Commands
Julie
Phone Commands
SMS/Text Message Commands
Loop
WrongAnswer
Farewells
Books, Films, and Music
HTML Scraping
AIML Wikipedia Search
Wikiquote Scraping
Interview Bot
Wolfram Alpha Example
Animals Set
My Name Is
Geek Pick-up Lines
DateAndTime
Greetings
AIML2
WhereIs (Wikidata)
Weather
Vision
Escalate Email
Colors
Wikipedia Search



Creating

You can create your own script from the browse script page. To create a script, just give it a name, description, and category to categorize it under. You can either make the script private and choose who can access it, or make it public and accessible by anyone.



Create new script

Script Name

Language

programming language or type of script

Description

optional description

Details □

Disclaimer □

License

optional license to release the script and all of its content under

Website

if this script has its own website, you can enter it here

Website Subdomain (or domain)

you can choose a subdomain to host your script's own website, or give a domain that you have registered and forward to this server's ip ad

Categories

comma seperated list of categories to categorize the script under

Tags

optional comma seperated list of tags to tag the script under

□ Private

□ Hidden

Access Mode: Everyone

Fork Access Mode: Administrators

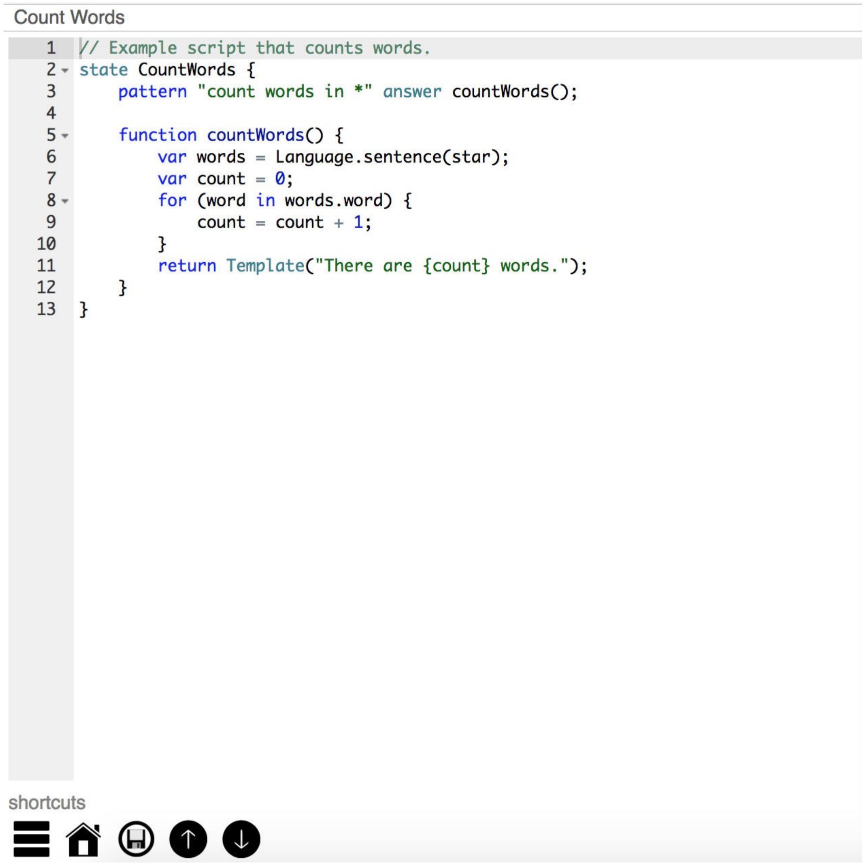
Content Rating: Teen

Ad Code □

Properties

Properties	Description
Script Name	Enter a script name.
Language	Programming language or type of script.
Description	Optional description.
Details	You can enter optional additional information.
Disclaimer	You can enter optional legal information.
License	Optional license to release the script and all of its content under.
Website	If this script has its own website, you can enter it here.
Website Subdomain (or domain)	You can choose a subdomain to host your script's own website, or give a domain that you have registered and forward to this server.
Categories	Comma separated list of categories to categorize the script under.
Tags	Optional comma separated list of tags to tag the script under.
Private	A private script is not visible to the public, only to the user and users grant access.
Hidden	A hidden script is not displayed in the browse directory.
Access Mode	Define who can access this script.
Fork Access Mode	Define who can fork(copy) this script.
Content Rating	Rate the script.
Ad Code	You can display ads on your script's pages.

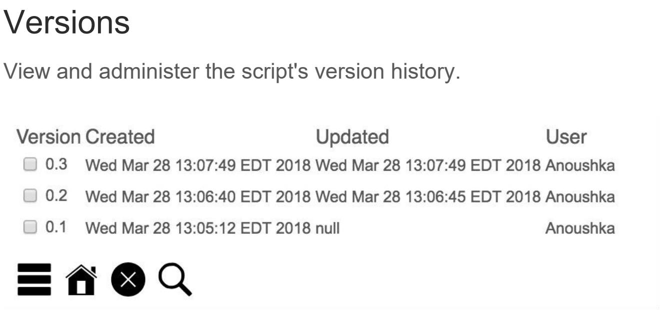
Editing Source



Count Words

```
// Example script that counts words.
state CountWords {
    pattern "count words in *" answer countWords();

    function countWords() {
        var words = Language.sentence(star);
        var count = 0;
        for (word in words.word) {
            count = count + 1;
        }
        return Template("There are {count} words.");
    }
}
```

Button	Description
☰	Choose script action from menu.
🏠	Back to home of script source.
💾	Save source.
⬆	Upload script
⬇	Download script.



Versions

View and administer the script's version history.

	Version	Created	Updated	User
□	0.3	Wed Mar 28 13:07:49 EDT 2018	Wed Mar 28 13:07:49 EDT 2018	Anoushka
□	0.2	Wed Mar 28 13:06:40 EDT 2018	Wed Mar 28 13:06:45 EDT 2018	Anoushka
□	0.1	Wed Mar 28 13:05:12 EDT 2018	null	Anoushka

Button

Button	Description
☰	Choose script action from menu.
	Back to home of script source.

Button	Description
🏠	
⊗	Permanently delete the script versions.
🔍	View the version source.

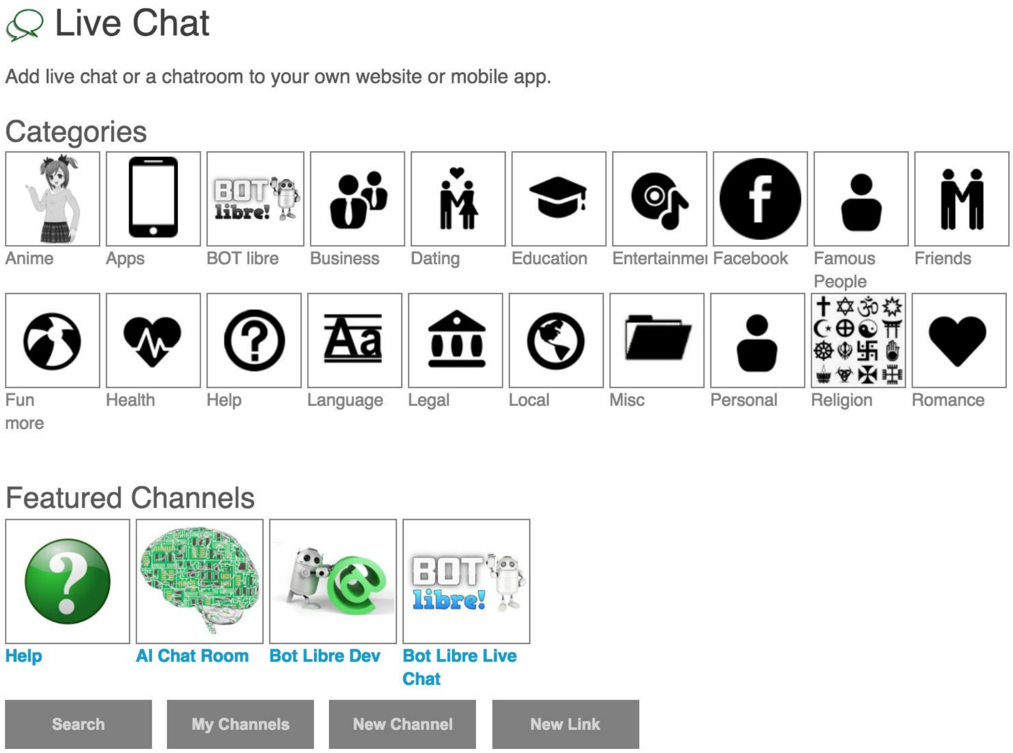
See Also

- Self
- AIML
- Introducing the Self Scripting Language
- Scripting your bot with AIML
- What classes are supported in Self

Live Chat

Overview

Live Chat allows you to create a chat channel that you can embed on your own website and service with your own live operators. You can also add a bot to your channel to learn from your operators, and service requests when no operators are available.



Live Chat

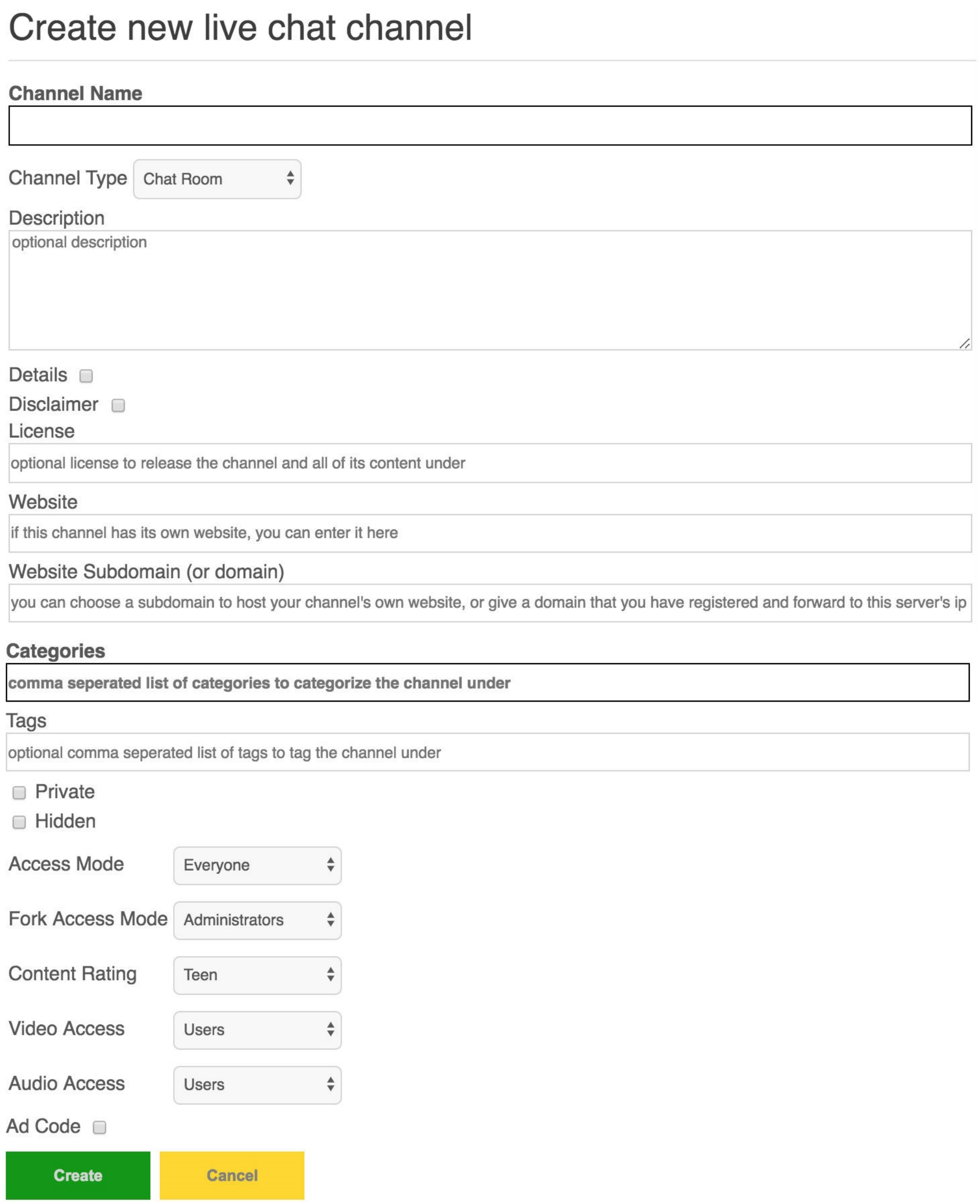
Add live chat or a chatroom to your own website or mobile app.

Categories

Featured Channels

Creating

You can create your own live chat channel from the browse live chat page. To create a live chat channel, just give it a name, description, and category to categorize it under. You can either make the live chat channel private and choose who can access it, or make it public and accessible by anyone.



Create new live chat channel

Channel Name

Channel Type: Chat Room

Description: optional description

Details □

Disclaimer □

License: optional license to release the channel and all of its content under

Website: if this channel has its own website, you can enter it here

Website Subdomain (or domain): you can choose a subdomain to host your channel's own website, or give a domain that you have registered and forward to this server's ip

Categories: **comma seperated list of categories to categorize the channel under**

Tags: optional comma seperated list of tags to tag the channel under

□ Private

□ Hidden

Access Mode: Everyone

Fork Access Mode: Administrators

Content Rating: Teen

Video Access: Users

Audio Access: Users

Ad Code □

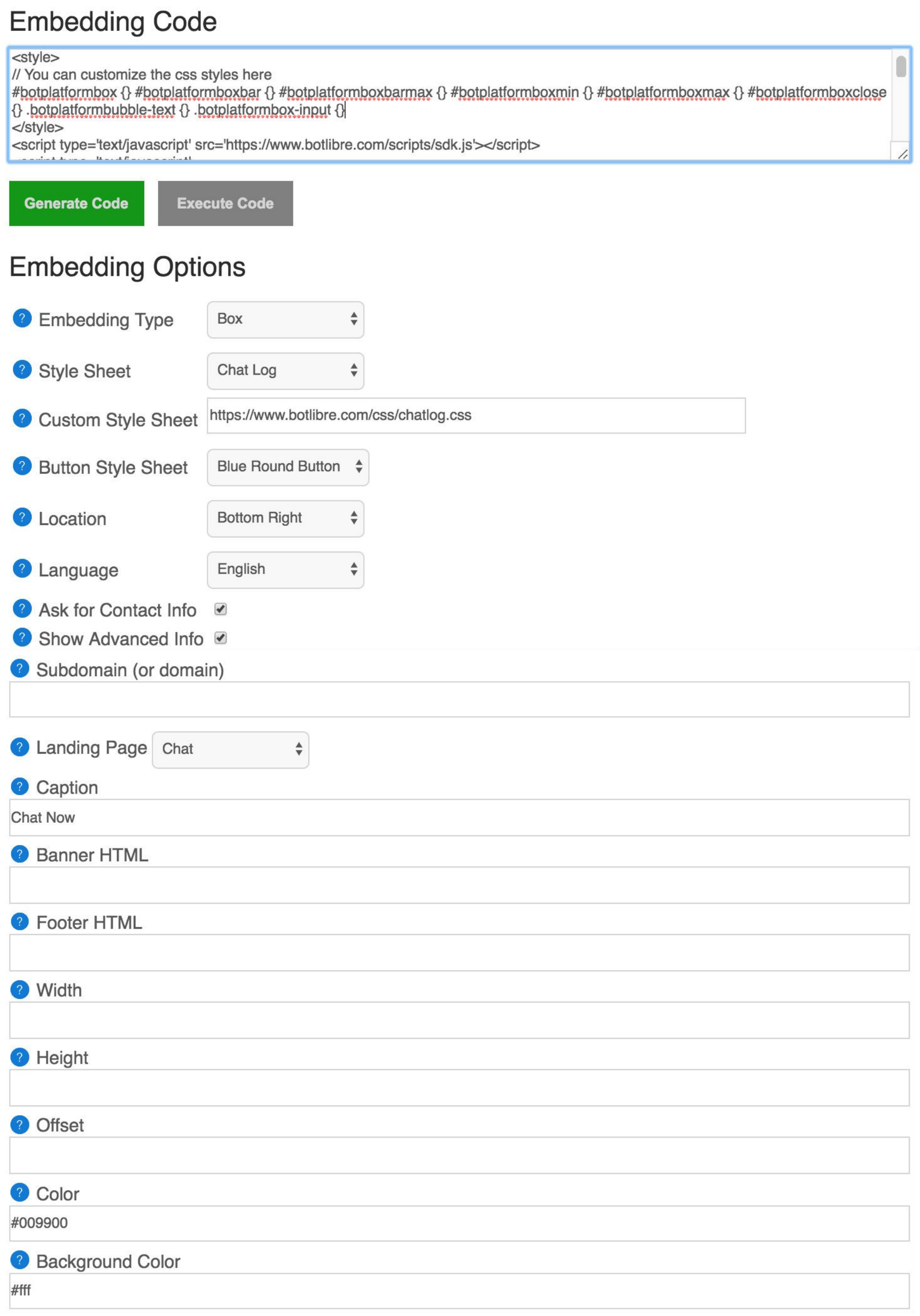
Properties

Properties	Description
Channel Name	Enter a channel name.
Channel Type	The channel can either be a chat room that allows multiple users to chat with each other, or one on one channels where users are queued to chat with an administrator.
Description	Optional description.
Details	You can enter optional additional information.
Disclaimer	You can enter optional legal information.
License	Optional license to release the channel and all of its content under.
Website	If this channel has its own website, you can enter it here.
Website Subdomain (or domain)	You can choose a subdomain to host your channel's own website, or give a domain that you have registered and forward to this server.
Categories	Comma separated list of categories to categorize the channel under.
Tags	Optional comma separated list of tags to tag the channel under.
Private	A private channel is not visible to the public, only to the user and users grant access.
Hidden	A hidden channel is not displayed in the browse directory.
Access Mode	Define who can access this channel.
Fork Access Mode	Define who can fork(copy) this channel.

Properties	Description
Content Rating	Rate the channel.
Video Access	Define who can broadcast video in this channel.
Audio Access	Define who can broadcast audio in this channel.
Ad Code	You can display ads on your channel's pages.

Embedding

You can embed your live chat channel or chat room on your own website or blog just by adding some simple html to your site. You are free to embed your own channels for personal, or commercial purposes.



Embedding Code

```
<style>
// You can customize the css styles here
#botplatformbox {} #botplatformboxbar {} #botplatformboxbarmax {} #botplatformboxmin {} #botplatformboxmax {} #botplatformboxclose
{} .botplatformbubble-text {} .botplatformbox-input {}
</style>
<script type='text/javascript' src='https://www.botlibre.com/scripts/sdk.js'></script>
```

Embedding Options

Embedding Type: Box

Style Sheet: Chat Log

Custom Style Sheet: https://www.botlibre.com/css/chatlog.css

Button Style Sheet: Blue Round Button

Location: Bottom Right

Language: English

Ask for Contact Info ☑

Show Advanced Info ☑

Subdomain (or domain):

Landing Page: Chat

Caption: Chat Now

Banner HTML:

Footer HTML:

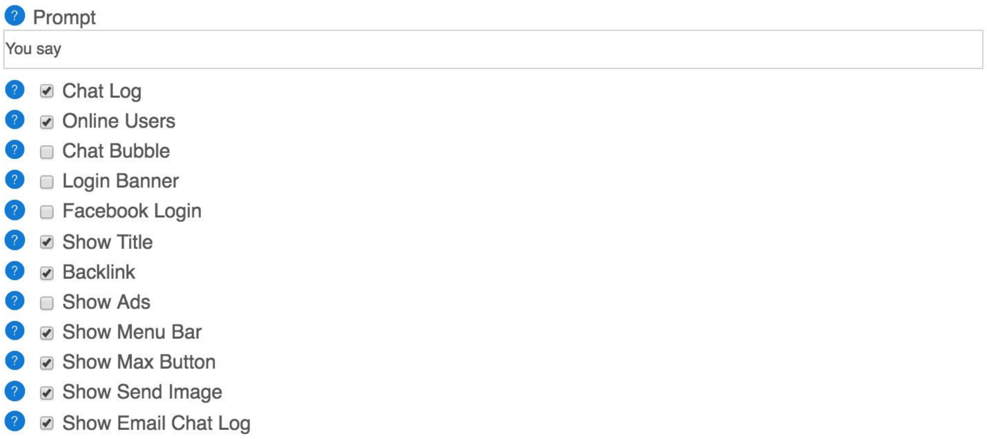
Width:

Height:

Offset:

Color: #009900

Background Color: #fff



Prompt

You say

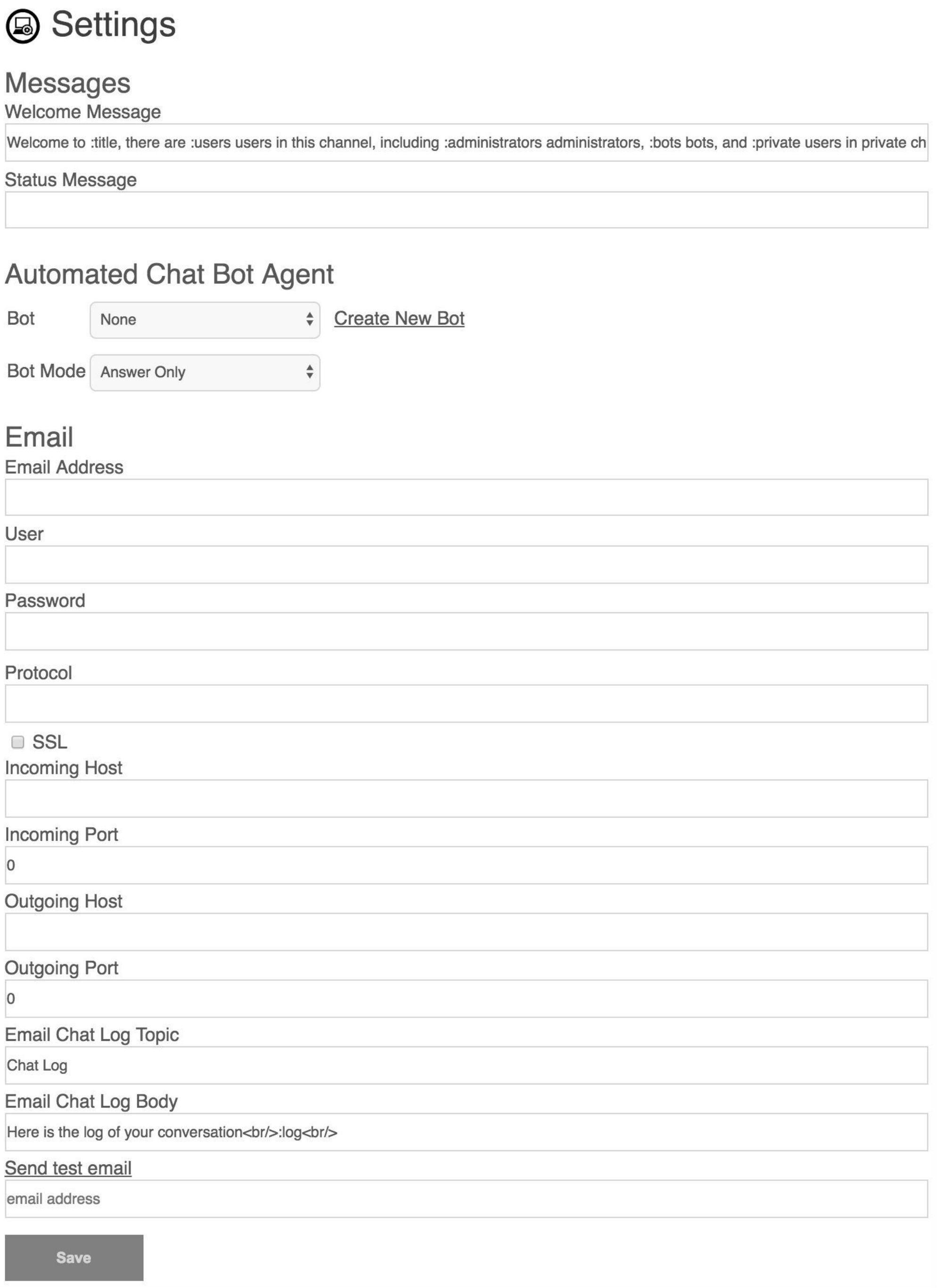
- ☑ Chat Log
- ☑ Online Users
- ☐ Chat Bubble
- ☐ Login Banner
- ☐ Facebook Login
- ☑ Show Title
- ☑ Backlink
- ☐ Show Ads
- ☑ Show Menu Bar
- ☑ Show Max Button
- ☑ Show Send Image
- ☑ Show Email Chat Log

Properties

Properties	Description
Subdomain (or domain)	You can choose a subdomain to host your own chat website, or give a domain that you have registered and forward to this server's ip address.
Embedding Type	Choose an embedding type.
Landing Page	Choose a loading page.
Caption	The text on the button or link.
Guest User	Guest user to connect as.
Password	Password for guest user (not secure).
Token	Token for guest user.
Style Sheet	The CSS style sheet or CSS to customize the page.
Banner HTML	An HTML script or page to embed as the page banner.
Footer HTML	An HTML script or page to embed as the page footer.
Width	The width (in pixels) of the frame, window, or box.
Height	The height (in pixels) of the frame, window, or box.
Offset	The offset (in pixels) of the box from the side.
Location	Choose a location side.
Color	The color to use for the button or link.
Background Color	The background color to use.
Prompt	The text prompt to use in the input (default is 'You say')
Send	The text for the send button (default is 'Send')
Ask for Contact Info	Ask the user for their contact information before connecting.
Chat Log	Show the chat log.
Online Users	Show the online users.
Chat Bubble	Print chat message in chat bubble.
Login Banner	Choose if login banner should be displayed.
Facebook Login	Choose if facebook login option should be provided.
Show Title	Choose if the channel's title should be displayed.
Backlink	Choose if a backlink to Bot Libre should be displayed (requires Bronse account).
Show Ads	Choose if the channel's ad show be displayed.

Settings

The settings page lets you configure your channel including linking an automated chat bot agent to service your channel, and configuring your welcome and status messages.



Settings

Messages

Welcome Message

Welcome to :title, there are :users users in this channel, including :administrators administrators, :bots bots, and :private users in private ch

Status Message

Automated Chat Bot Agent

Bot: None — Create New Bot

Bot Mode: Answer Only

Email

Email Address

User

Password

Protocol

□ SSL

Incoming Host

Incoming Port

0

Outgoing Host

Outgoing Port

0

Email Chat Log Topic

Chat Log

Email Chat Log Body

Here is the log of your conversation<br/>:log<br/>

Send test email

email address

Messages

Properties	Description
Welcome Message	Welcome message for your bot.
Status Message	Status message for your bot.

Automated Chat Bot Agent

Properties	Description
Bot	Choose the bot to monitor the channel from the list of your bot.
Bot Mode	The bot can be configured in three different modes: Listen Only,Answer Only, or Answer and Listen, which define how the bot will participate in the conversation.

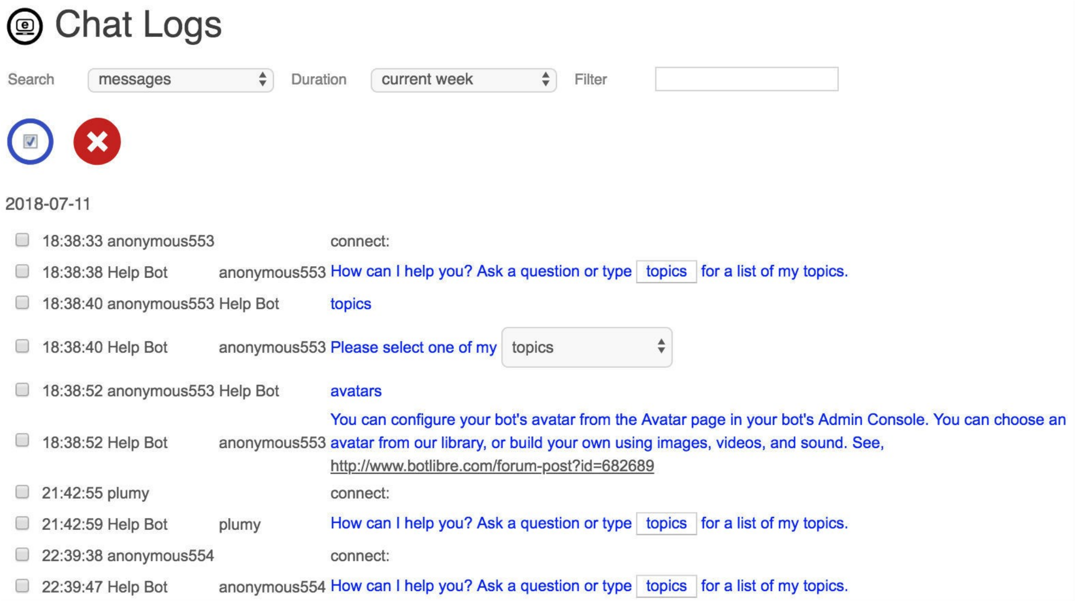
Email

Properties	Description

Properties	Description
Email Address	Email Address.
User	User name.
Password	User password.
Protocol	Protocol specifies interactions.
SSL	Creates a secure connection.
Incoming Host	Incoming host name.
Incoming Port	Incoming port number.
Outgoing Host	Outgoing host name.
Outgoing Port	Outgoing port number.
Email Chat Log Topic	
Send test email	

Chat Logs

The chat logs tab allows you to monitor your channel.



Chat Logs

Search: messages Duration: current week Filter:

☑ ☒

2018-07-11

```
□ 18:38:33 anonymous553                connect:
□ 18:38:38 Help Bot      anonymous553  How can I help you? Ask a question or type topics for a list of my topics.
□ 18:38:40 anonymous553 Help Bot       topics
□ 18:38:40 Help Bot      anonymous553  Please select one of my topics
□ 18:38:52 anonymous553 Help Bot       avatars
□ 18:38:52 Help Bot      anonymous553  You can configure your bot's avatar from the Avatar page in your bot's Admin Console. You can choose an
                                       avatar from our library, or build your own using images, videos, and sound. See,
                                       http://www.botlibre.com/forum-post?id=682689
□ 21:42:55 plumy                       connect:
□ 21:42:59 Help Bot      plumy         How can I help you? Ask a question or type topics for a list of my topics.
□ 22:39:38 anonymous554                connect:
□ 22:39:47 Help Bot      anonymous554  How can I help you? Ask a question or type topics for a list of my topics.
```

Properties

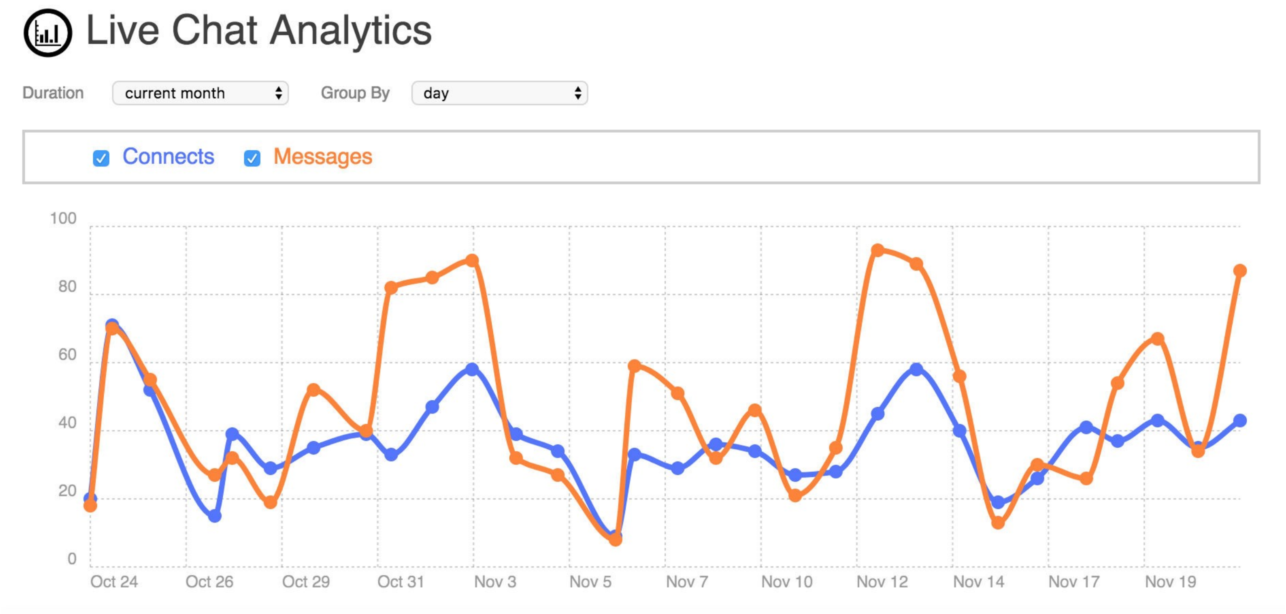
Properties	Description
Search	Search type.
Duration	Duration of time.
Filter	Filter the results to only include messages containing the filter text.

Toolbar

Icon	Property Name	Description
☑	Select	Select all of the messages.
☒	Delete	Delete the selected messages.

Analytics

Theanalytics tab allows you to view your channel's analytics in the form of charts and tables. You can view your live chat channel charts for a duration period of either the current week, current month, or all time by selecting a duration period from the duration drop down select. You can group your channel's analytics either by day, week, or month by selecting a group from a group by drop down select.



Live Chat Analytics

Duration: current month

Group By: day

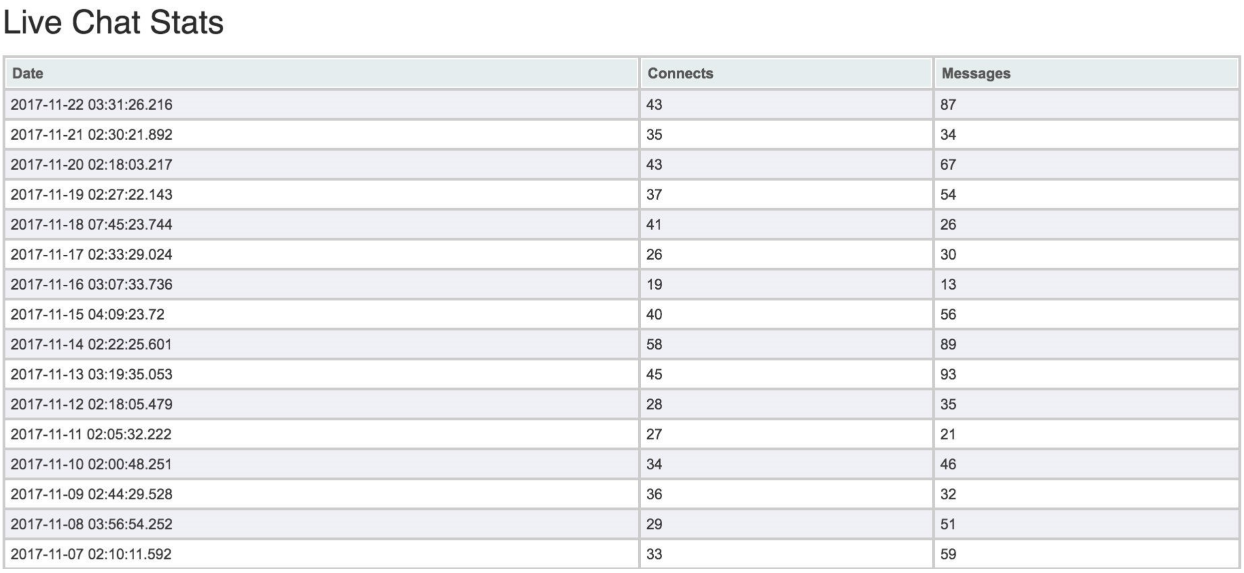
☑ Connects ☑ Messages

Properties

Properties	Description
Duration	Displays a graph for a given period of time. User can choose duration of the graph to be displayed for either week, month, or everything.
Group By	Groups graph data either by day, week, or month.

Analytics Table

The analytics table displays your channel's analytics data. The data in the table is graphed in a chart above. You can view your channel's analytics inside table by sorting it in either increasing or decreasing order by clicking on table column's title.



Live Chat Stats

Date	Connects	Messages
2017-11-22 03:31:26.216	43	87
2017-11-21 02:30:21.892	35	34
2017-11-20 02:18:03.217	43	67
2017-11-19 02:27:22.143	37	54
2017-11-18 07:45:23.744	41	26
2017-11-17 02:33:29.024	26	30
2017-11-16 03:07:33.736	19	13
2017-11-15 04:09:23.72	40	56
2017-11-14 02:22:25.601	58	89
2017-11-13 03:19:35.053	45	93
2017-11-12 02:18:05.479	28	35
2017-11-11 02:05:32.222	27	21
2017-11-10 02:00:48.251	34	46
2017-11-09 02:44:29.528	36	32
2017-11-08 03:56:54.252	29	51
2017-11-07 02:10:11.592	33	59

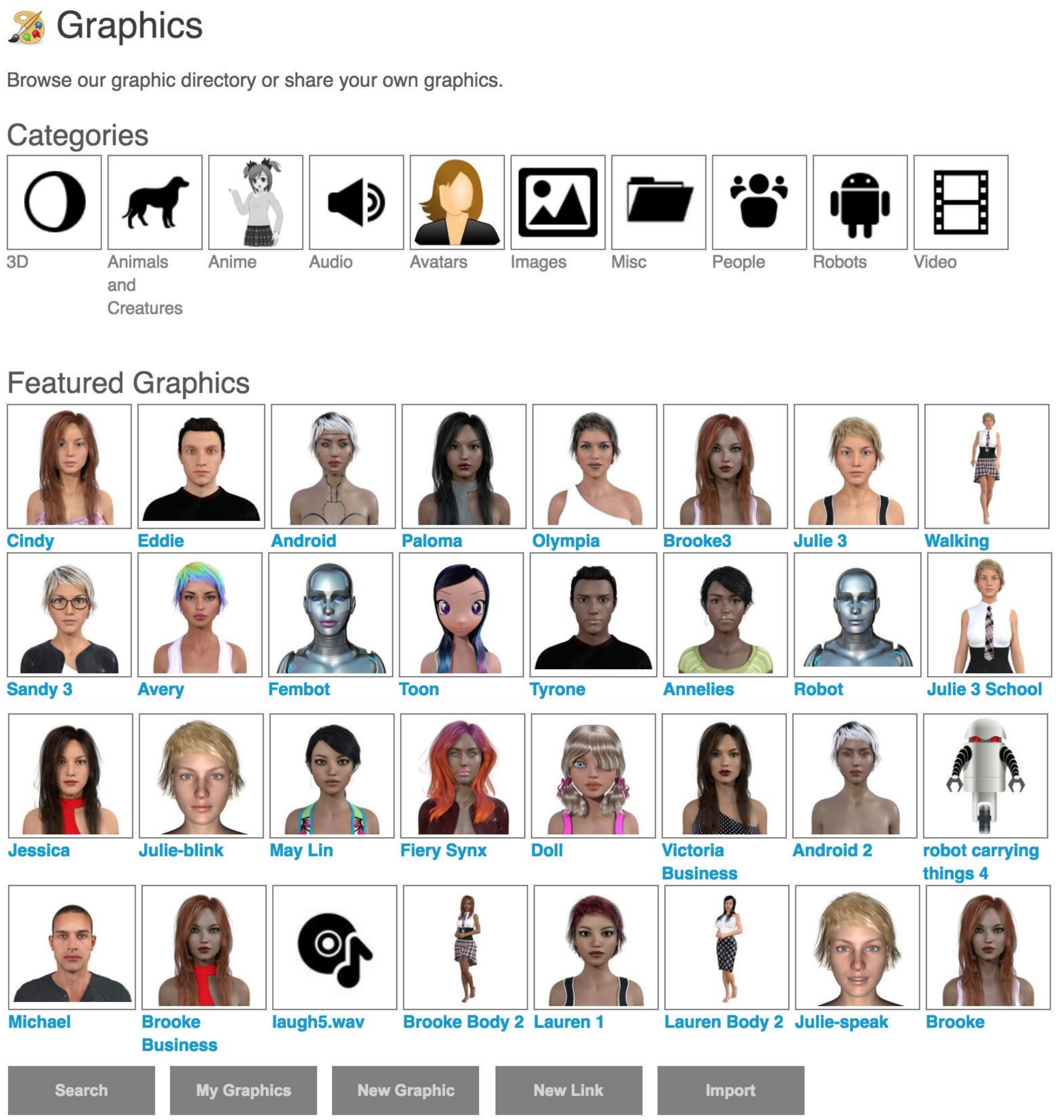
See Also

- Live Chat

Graphics

Overview

The graphics library let you upload and share image, video, and audio files.



Graphics

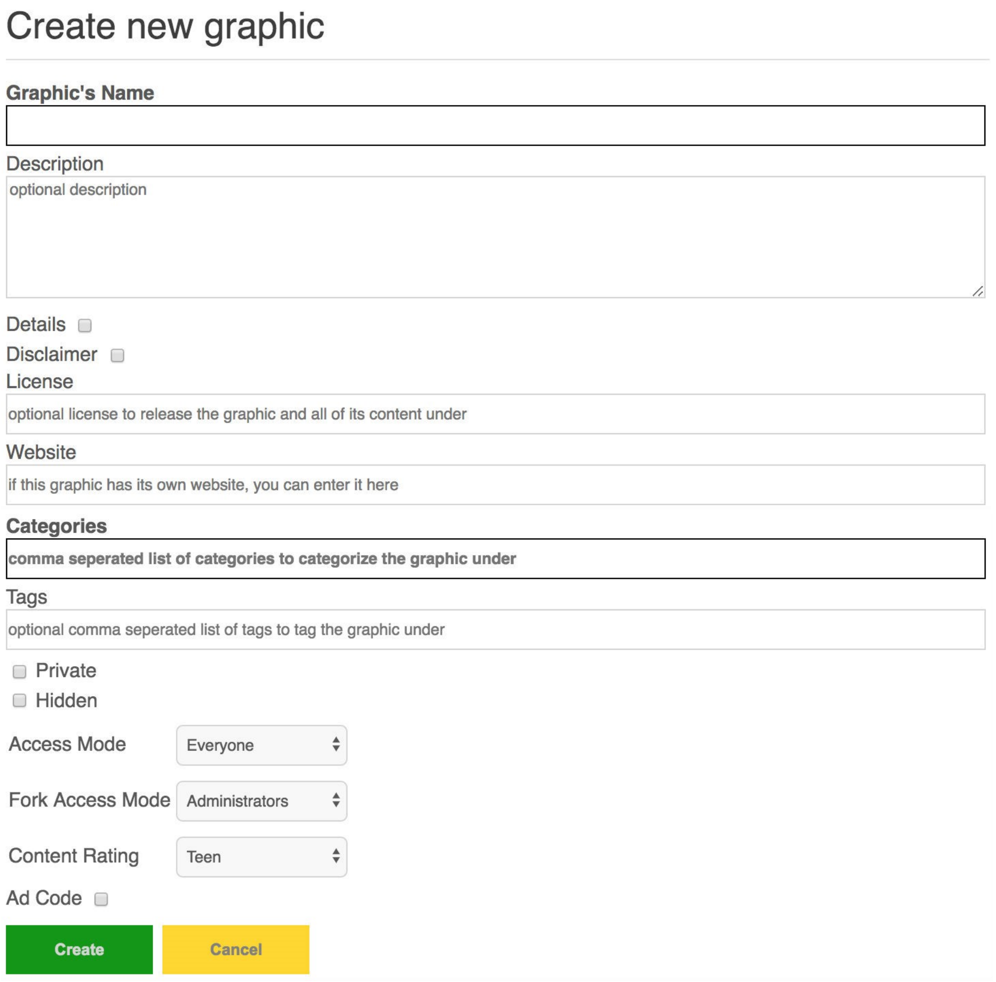
Browse our graphic directory or share your own graphics.

Categories

Featured Graphics

Creating

You can create your own graphic from the browse graphic page. To create a graphic, just give it a name, description, and category to categorize it under. You can either make the graphic private and choose who can access it, or make it public and accessible by anyone.



Create new graphic

Graphic's Name

Description

optional description

Details □

Disclaimer □

License

optional license to release the graphic and all of its content under

Website

if this graphic has its own website, you can enter it here

Categories

comma seperated list of categories to categorize the graphic under

Tags

optional comma seperated list of tags to tag the graphic under

□ Private

□ Hidden

Access Mode	Everyone
Fork Access Mode	Administrators
Content Rating	Teen

Ad Code □

Properties

Properties	Description
Graphic's Name	Enter a graphic name.
Description	Optional description.
Details	You can enter optional additional information.
Disclaimer	You can enter optional legal information.
License	Optional license to release the graphic and all of its content under.
Website	If this graphic has its own website, you can enter it here.
Categories	Comma separated list of categories to categorize the graphic under.
Tags	Optional comma separated list of tags to tag the graphic under.
Private	A private graphic is not visible to the public, only to the user and users grant access.
Hidden	A hidden graphic is not displayed in the browse directory.
Access Mode	Define who can access this graphic.
Fork Access Mode	Define who can fork(copy) this graphic.
Content Rating	Rate the graphic.
Ad Code	You can display ads on your graphic's pages.

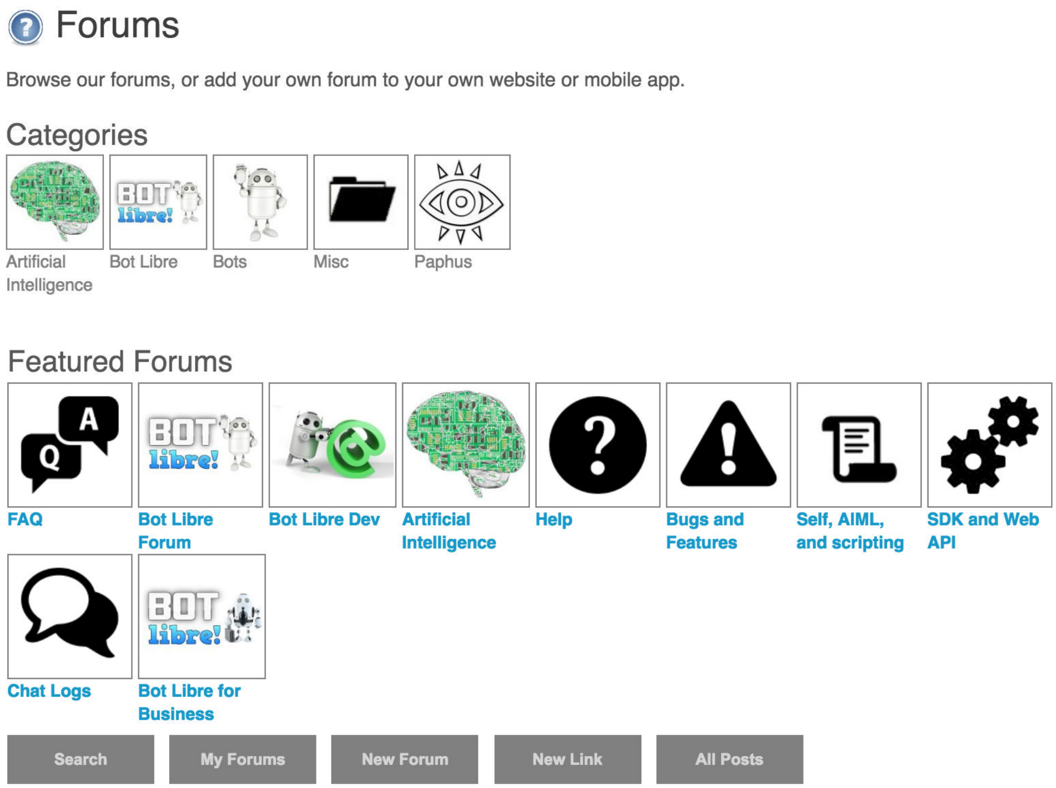
See Also

- Browse Graphics

Forums

Overview

You can browse Bot Libre's forums for How Tos, FAQs, and other useful information. You can also create your own forum, and embed it on your own website.

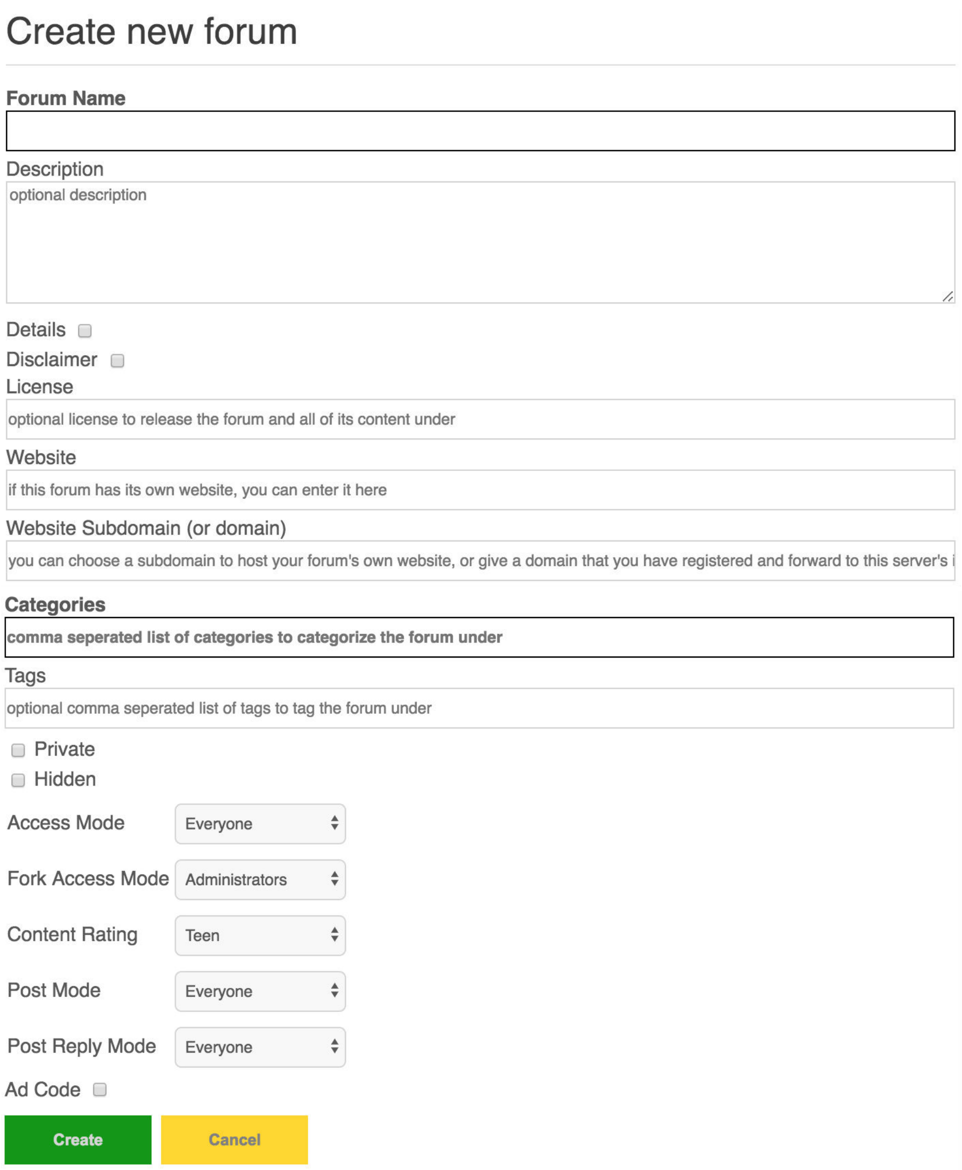


Forums

Browse our forums, or add your own forum to your own website or mobile app.

Creating

You can create your own forum from the browse forum page. To create a forum, just give it a name, description, and category to categorize it under. You can either make the forum private and choose who can access it, or make it public and accessible by anyone.



Create new forum

Forum Name

Description

optional description

Details □

Disclaimer □

License

optional license to release the forum and all of its content under

Website

if this forum has its own website, you can enter it here

Website Subdomain (or domain)

you can choose a subdomain to host your forum's own website, or give a domain that you have registered and forward to this server's i

Categories

comma seperated list of categories to categorize the forum under

Tags

optional comma seperated list of tags to tag the forum under

□ Private

□ Hidden

Access Mode	Everyone
Fork Access Mode	Administrators
Content Rating	Teen
Post Mode	Everyone
Post Reply Mode	Everyone

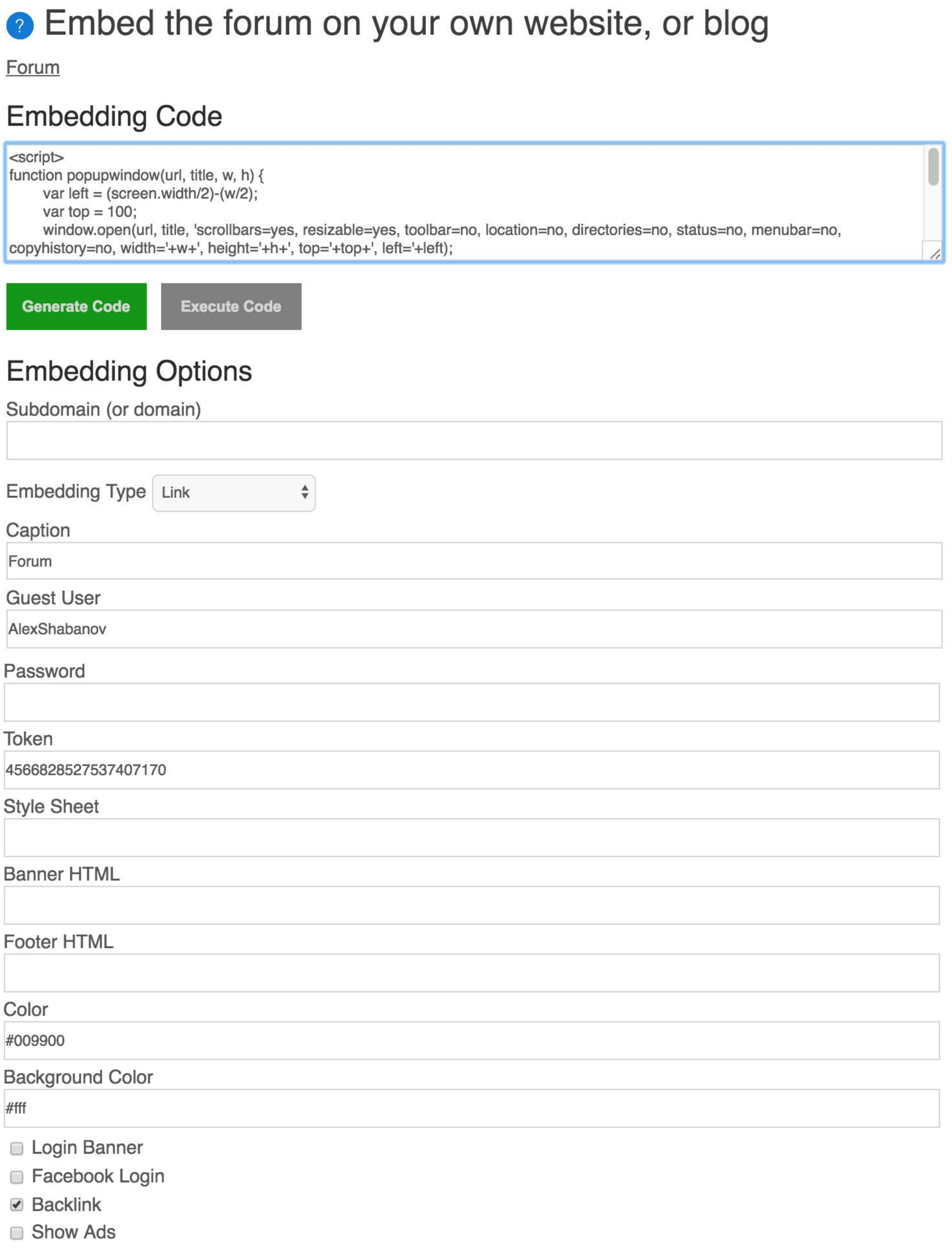
Ad Code □

Properties

Properties	Description
Forum Name	Enter a forum name.
Description	Optional description.
Details	You can enter optional additional information.
Disclaimer	You can enter optional legal information.
License	Optional license to release the forum and all of its content under.
Website	If this forum has its own website, you can enter it here.
Website Subdomain (or domain)	You can choose a subdomain to host your forum's own website, or give a domain that you have registered and forward to this server.
Categories	Comma separated list of categories to categorize the forum under.
Tags	Optional comma separated list of tags to tag the forum under.
Private	A private forum is not visible to the public, only to the user and users grant access.
Hidden	A hidden forum is not displayed in the browse directory.
Access Mode	Define who can access this forum.
Fork Access Mode	Define who can fork(copy) this forum.
Content Rating	Rate the forum.
Post Mode	Define who can post to this forum.
Post Reply Mode	Define who can reply to posts in this forum.
Ad Code	You can display ads on your forum's pages.

Embedding

You can embed your forum on your own website or blog just by adding some simple html to your site. You are free to embed your own forums for personal, or commercial purposes.



Embed the forum on your own website, or blog

Forum

Embedding Code

```
<script>
function popupwindow(url, title, w, h) {
      var left = (screen.width/2)-(w/2);
      var top = 100;
      window.open(url, title, 'scrollbars=yes, resizable=yes, toolbar=no, location=no, directories=no, status=no, menubar=no,
copyhistory=no, width='+w+', height='+h+', top='+top+', left='+left);
```

Embedding Options

Subdomain (or domain)

Embedding Type: Link

Caption: Forum

Guest User: AlexShabanov

Password

Token: 4566828527537407170

Style Sheet

Banner HTML

Footer HTML

Color: #009900

Background Color: #fff

□ Login Banner
□ Facebook Login
☑ Backlink
□ Show Ads

Properties

Properties	Description
Subdomain (or domain)	You can choose a subdomain to host your own chat website, or give a domain that you have registered and forward to this server's ip address.
Embedding Type	Choose an embedding type.
Caption	The text on the button or link.
Guest User	Guest user to connect as.
Password	Password for guest user (not secure).
Token	Token for guest user.

Properties	Description
Style Sheet	The CSS style sheet or CSS to customize the page.
Banner HTML	An HTML script or page to embed as the page banner.
Footer HTML	An HTML script or page to embed as the page footer.
Color	The color to use for the button or link.
Background Color	The background color to use.
Login Banner	Choose if login banner should be displayed.
Facebook Login	Choose if facebook login option should be provided.
Backlink	Choose if a backlink to Bot Libre should be displayed (requires Bronse account).
Show Ads	Choose if the forums's ad show be displayed.

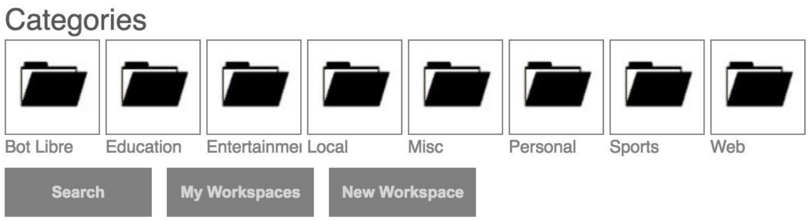
See Also

- Browse Forums

Workspaces

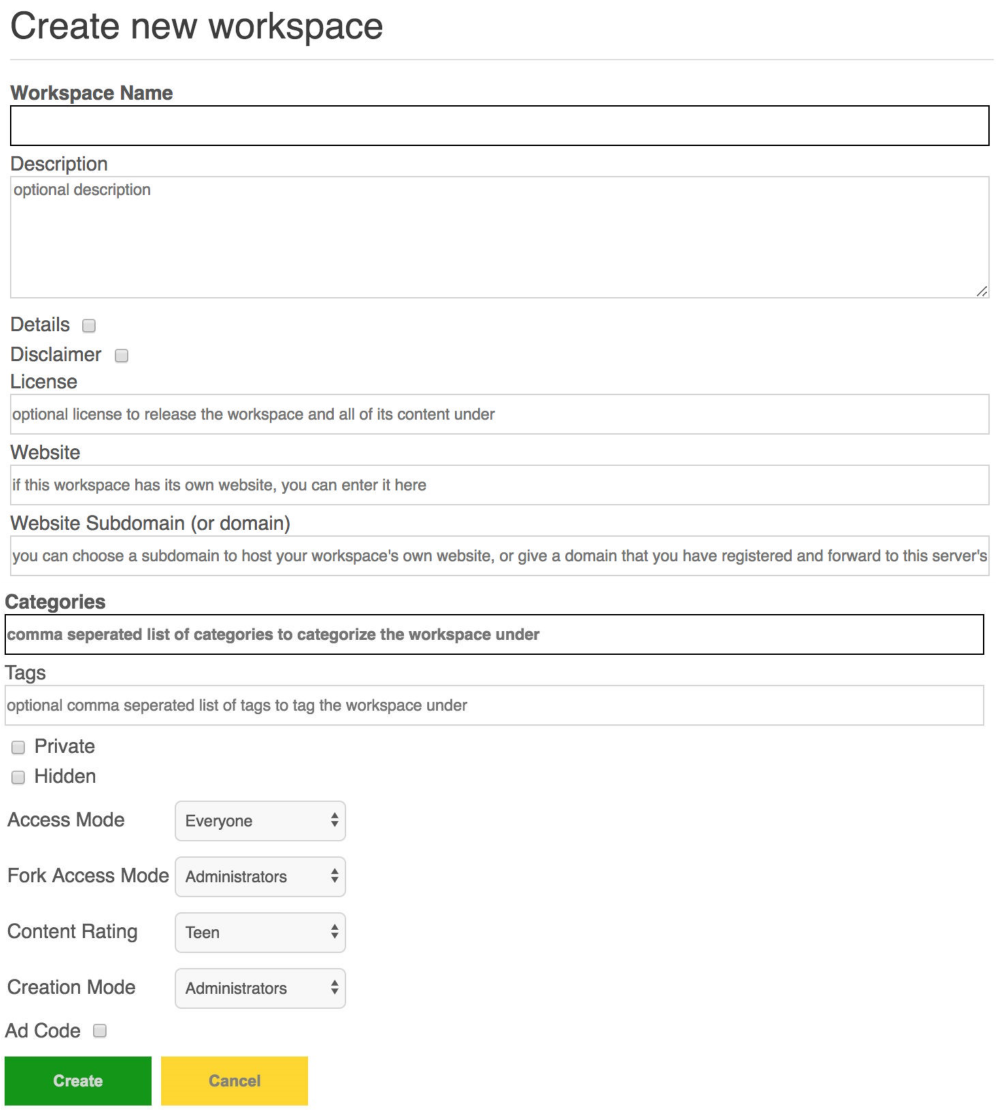
Overview

A workspace is a directory or domain object that lets you group your content. You can create your own Workspace for your business, school, or club to share bots and other content. As the admin of your workspace you can control who can access its content.



Creating

You can create your own workspace from the browse workspace page. To create a workspace, just give it a name, description, and category to categorize it under. You can either make the workspace private and choose who can access it, or make it public and accessible by anyone.



Create new workspace

Workspace Name

Description

optional description

Details □

Disclaimer □

License

optional license to release the workspace and all of its content under

Website

if this workspace has its own website, you can enter it here

Website Subdomain (or domain)

you can choose a subdomain to host your workspace's own website, or give a domain that you have registered and forward to this server's

Categories

comma seperated list of categories to categorize the workspace under

Tags

optional comma seperated list of tags to tag the workspace under

□ Private

□ Hidden

Access Mode: Everyone

Fork Access Mode: Administrators

Content Rating: Teen

Creation Mode: Administrators

Ad Code □

Properties

Properties	Description
Workspace Name	Enter a workspace name.
Description	Optional description.
Details	You can enter optional additional information.
Disclaimer	You can enter optional legal information.
License	Optional license to release the domain and all of its content under.
Website	If this domain has its own website, you can enter it here.
Website Subdomain (or domain)	You can choose a subdomain to host your workspace's own website, or give a domain that you have registered and forward to this server.
Categories	Comma separated list of categories to categorize the workspace under.
Tags	Optional comma separated list of tags to tag the workspace under.
Private	A private workspace is not visible to the public, only to the user and users grant access.
Hidden	A hidden workspace is not displayed in the browse directory.
Access Mode	Define who can access this workspace.
Fork Access Mode	Define who can fork(copy) this workspace.
Content Rating	Rate the workspace.
Creation Mode	Define who can create channels, forums, and bots in this workspace.

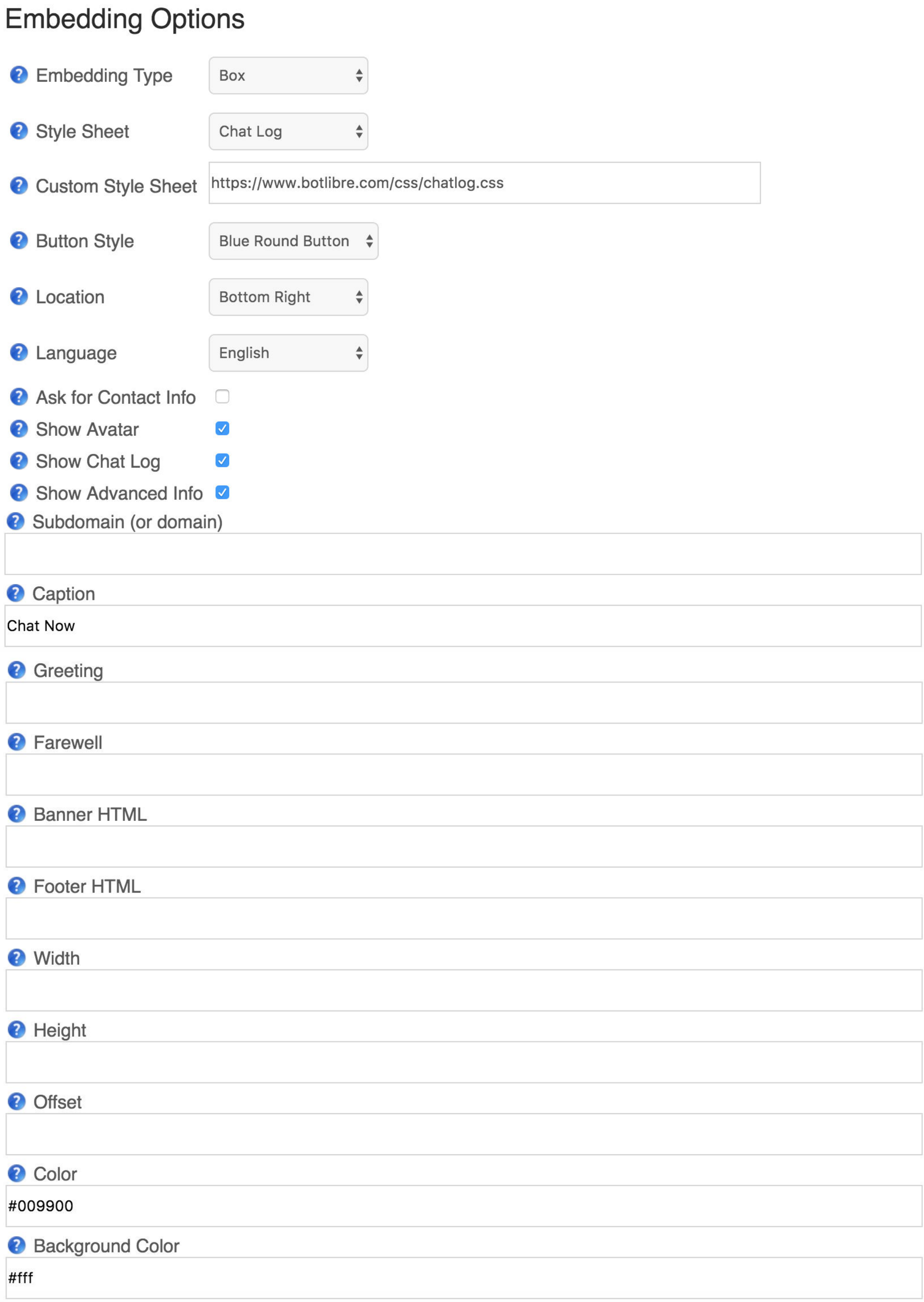
See Also

- Browse Workspaces

Embedding

Overview

You can embed your bot on your website from its Embed page by clicking on the 'Embed' button. You can embed your bot using JavaScript or HTML, the Embed page provides 6 different embedding types, and many options. You can also create a website and subdomain to host your bot, or link your own domain to your bot's webpage.



Embedding Options

Embedding Type	Box
Style Sheet	Chat Log
Custom Style Sheet	https://www.botlibre.com/css/chatlog.css
Button Style	Blue Round Button
Location	Bottom Right
Language	English
Ask for Contact Info	☐
Show Avatar	☑
Show Chat Log	☑
Show Advanced Info	☑

Subdomain (or domain)

Caption

Chat Now

Greeting

Farewell

Banner HTML

Footer HTML

Width

Height

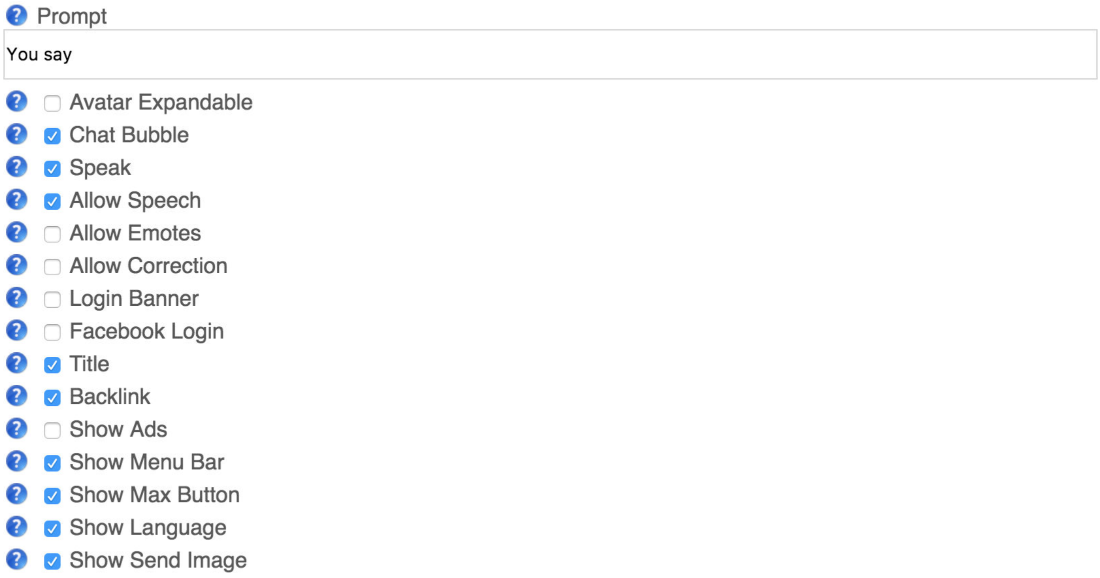
Offset

Color

#009900

Background Color

#fff



Prompt

You say

- ☐ Avatar Expandable
- ☑ Chat Bubble
- ☑ Speak
- ☑ Allow Speech
- ☐ Allow Emotes
- ☐ Allow Correction
- ☐ Login Banner
- ☐ Facebook Login
- ☑ Title
- ☑ Backlink
- ☐ Show Ads
- ☑ Show Menu Bar
- ☑ Show Max Button
- ☑ Show Language
- ☑ Show Send Image

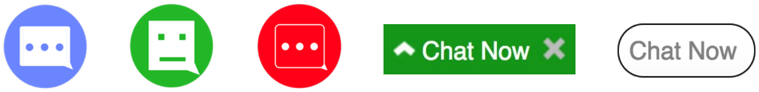
Properties

Properties	Description
Subdomain (or domain)	You can choose a subdomain to host your own bot's website, or give a domain that you have registered and forward to this server's ip address.
Embedding Type	Choose an embedding type.
Style Sheet	Choose the style sheet for the embedded chat.
Custom Style Sheet	Custom css for the embedded chat style sheet.
Button Style	Choose different button style for the embedded chat.
Location	Choose a location side.
Language	Choose a language for the text displayed in a embedded chat.
Ask for Contact Info	Ask the user for their contact information before connecting.
Show Avatar	Displays bot avatar in the embed chat.
Show Chat Log	Displays chat log in the embed chat.
Show Advanced Info	Shows advanced unfo for the embedding chat.
Caption	The text on the button or link.
Greeting	Optional greeting to display.
Farewell	Optional farewell/disconnected message to display.
Guest User	Guest user to connect as, you need to add this user to your bot's user's.
Password	Password for guest user (not secure).
Token	Token for guest user.
Style Sheet	The CSS style sheet or CSS to customize the page.
Banner HTML	An HTML script or page to embed as the page banner.
Footer HTML	An HTML script or page to embed as the page footer.
Width	The width (in pixels) of the frame, window, or box.
Height	The height (in pixels) of the frame, window, or box.
Offset	The offset (in pixels) of the box from the side.
Color	The color to use for the button or link.
Background Color	The background color to use.
Prompt	The text prompt to use in the input (default is 'You say')
Send	The text for the send button (default is 'Send')
Avatar Expandable	Choose if the bot's avatar should be expandable.
Chat Bubble	Print chat message in chat bubble (only for box chat).
Speak	Choose if the bot's voice should be enabled.
Allow Speech	Choose if the bot's voice should be allowed.
Allow Emotes	Choose if emotes and actions should be allowed.
Allow Correction	Choose if users should be allowed to correct the bot's responses.
Login Banner	Choose if login banner should be displayed.
Facebook Login	Choose if facebook login option should be provided.
Title	Choose if bot's name should be displayed as the title header.

Properties	Description
Backlink	Choose if a backlink to Bot Libre should be displayed (requires Bronse account).
Show Ads	Choose if the bot's ad show be displayed.
Show Menu Bar	Choose if menu bar option should be displayed.
Show Max Button	Choose if max button menu option should be displayed.
Show Language	Choose if language menu option should be displayed
Show Send Image	Choose if send image menu option should be displayed

Button Styles

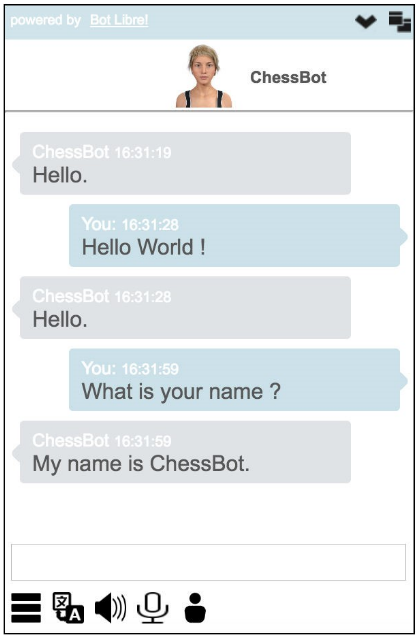
Choose any of the 11 available button styles for the embed page. Simply select any button style from button style drop down menu. Some of the buttons styles are shown below.



Chat Styles

Choose five different styles for their embedding by selecting a specific style sheet from Style Sheet drop down menu.

Chat Log Style



ChessBot 16:31:19
Hello.

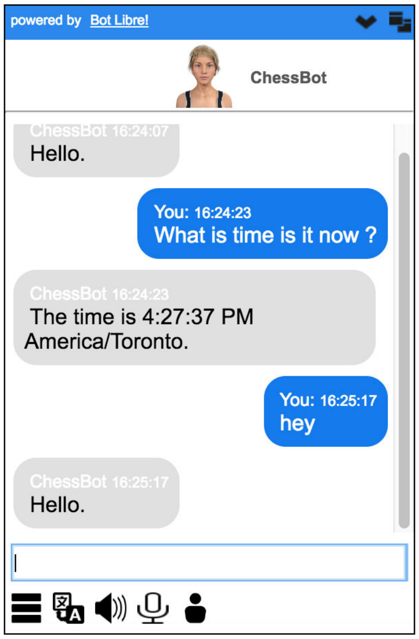
You: 16:31:28
Hello World !

ChessBot 16:31:28
Hello.

You: 16:31:59
What is your name ?

ChessBot 16:31:59
My name is ChessBot.

Social Chat Style



ChessBot 16:24:07
Hello.

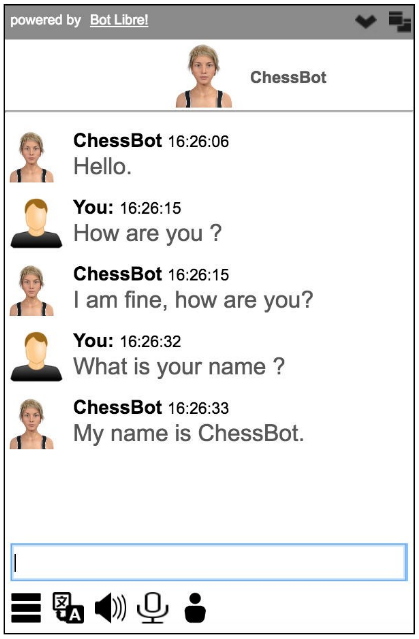
You: 16:24:23
What is time is it now ?

ChessBot 16:24:23
The time is 4:27:37 PM America/Toronto.

You: 16:25:17
hey

ChessBot 16:25:17
Hello.

Chat Room Style



ChessBot 16:26:06
Hello.

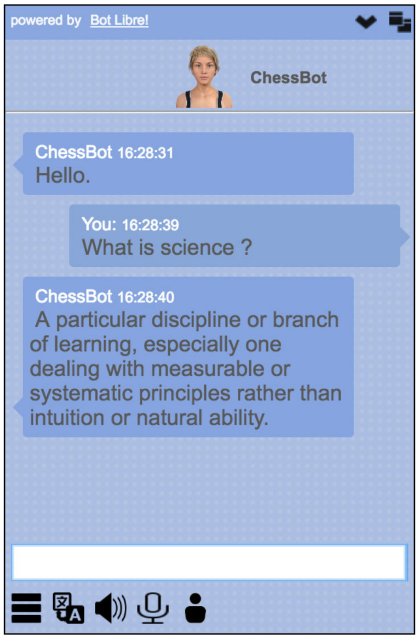
You: 16:26:15
How are you ?

ChessBot 16:26:15
I am fine, how are you?

You: 16:26:32
What is your name ?

ChessBot 16:26:33
My name is ChessBot.

Blue Chat Style

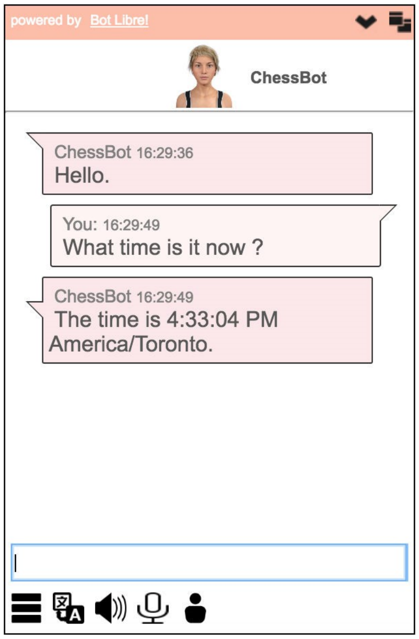


ChessBot 16:28:31
Hello.

You: 16:28:39
What is science ?

ChessBot 16:28:40
A particular discipline or branch of learning, especially one dealing with measurable or systematic principles rather than intuition or natural ability.

Pink Chat Style



ChessBot 16:29:36
Hello.

You: 16:29:49
What time is it now ?

ChessBot 16:29:49
The time is 4:33:04 PM America/Toronto.

Custom Styles

Add your own custom style sheet for your embed. Simply select custom style from style sheet drop down menu. Then, enter the url pointing to your custom style sheet inside custom style sheet input field.

See Also

- How to add a virtual agent to your website.
- Embedding your bot on your own website or blog

SDK

Overview

Bot Libre provides an open source JavaScript, Java, Android, and iOS SDK that lets you use any of Bot Libre's services on your own website or mobile app.

See Also

- SDK
- Create your own bot app with the Bot Libre SDK.

Web API

Overview

In addition to being able to embed your chat bots on your own website, and access them from any Android device, you can also access your chat bots through the Bot Libre web API. The web API gives you the advantage of having complete control of your bot's client interface.

You can use the web API to access your bot from your own website through JavaScript, PHP or any other language. You can also use the web API to create your own mobile application to access your bot, such as an Android or iOS application.

A web API, is a set of HTTP GET/POST URI's that allow sending and receiving of message data. When you browse a website, your browser makes a series of HTTP GET/POST requests to URIs that return HTML content. In a web service the URIs return XML or JSON data, instead of HTML content.

The Bot Libre web API provides three REST APIs in one. The first is a set of HTTP GET URIs that take FORM data, and return XML data. The second set of API's take HTTP POST XML data, and return XML data. The third set of API's take HTTP POST JSON data, and return JSON data.

Most applications would use the XML POST API, but some application may choose to use the FORM GET API because of its simplicity. The APIs are identical other than their mechanism.

See Also

- The Bot Libre chat bot web API
- Web API Console
- Create your own bot app with the Bot Libre SDK

Speech API

Overview

The Bot Libre JavaScript SDK provides free dynamic access to all of Bot Libre's services including chat bots, live chat, chat rooms, forums, and speech. The speech API for the SDK can be used in conjunction with chat bots, live chat, or on its own.

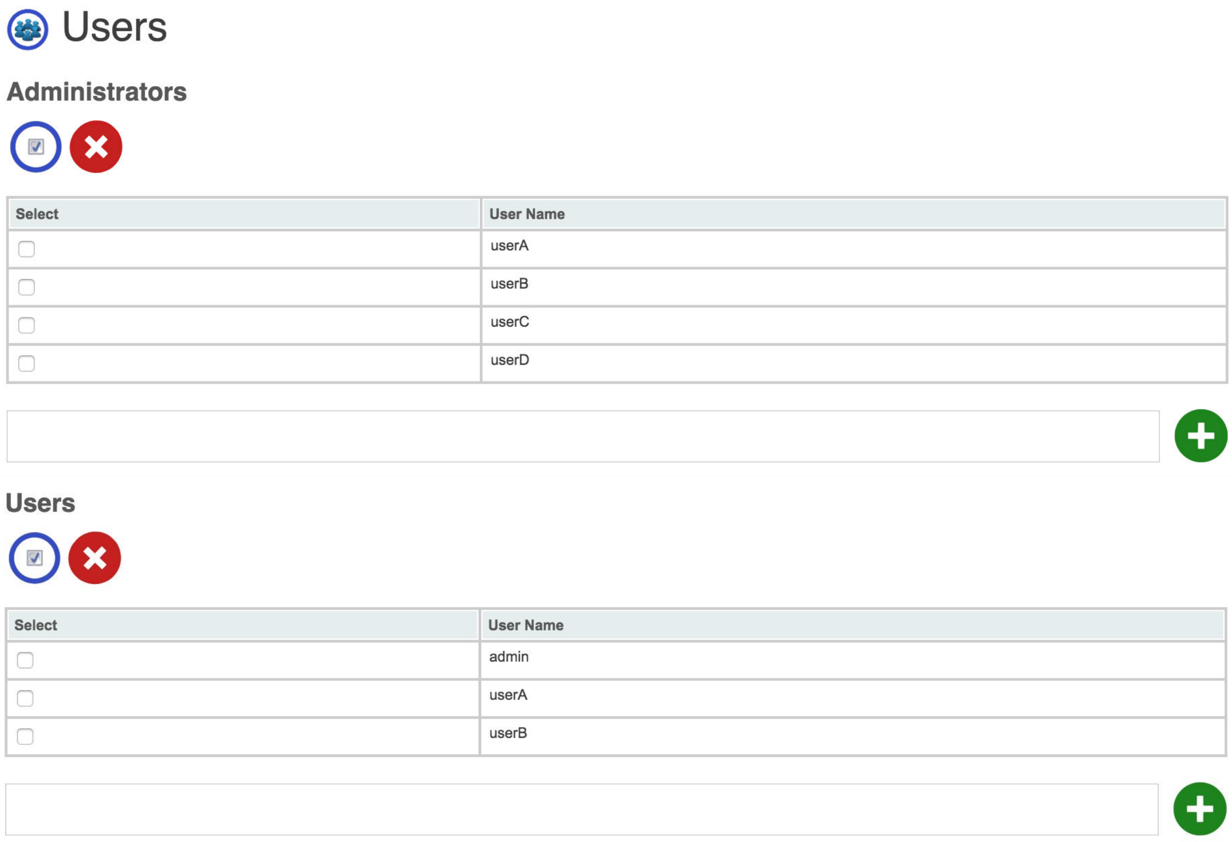
See Also

- The Bot Libre chat bot web API
- API Test
- Add Speech to your Website using the Bot Libre TTS JavaScript SDK
- Create your own bot app with the Bot Libre SDK

Users

Overview

The users tab allows you to add users, and administrators to your bot.



Users

Administrators

☑ ✖

Select	User Name
□	userA
□	userB
□	userC
□	userD

⊕

Users

☑ ✖

Select	User Name
□	admin
□	userA
□	userB

⊕

Toolbar

Administrators - Enter the user id of the user to add as an administrator.

Icon	Property Name	Description
☑	Select	Select all administrators.
✖	Remove	Remove all selected administrator.
⊕	Add	Add the new administrator.

Users - Enter the user id of the user to add.

Icon	Property Name	Description
☑	Select	Select all users.
✖	Remove	Remove all selected user.
⊕	Add	Add the new user.

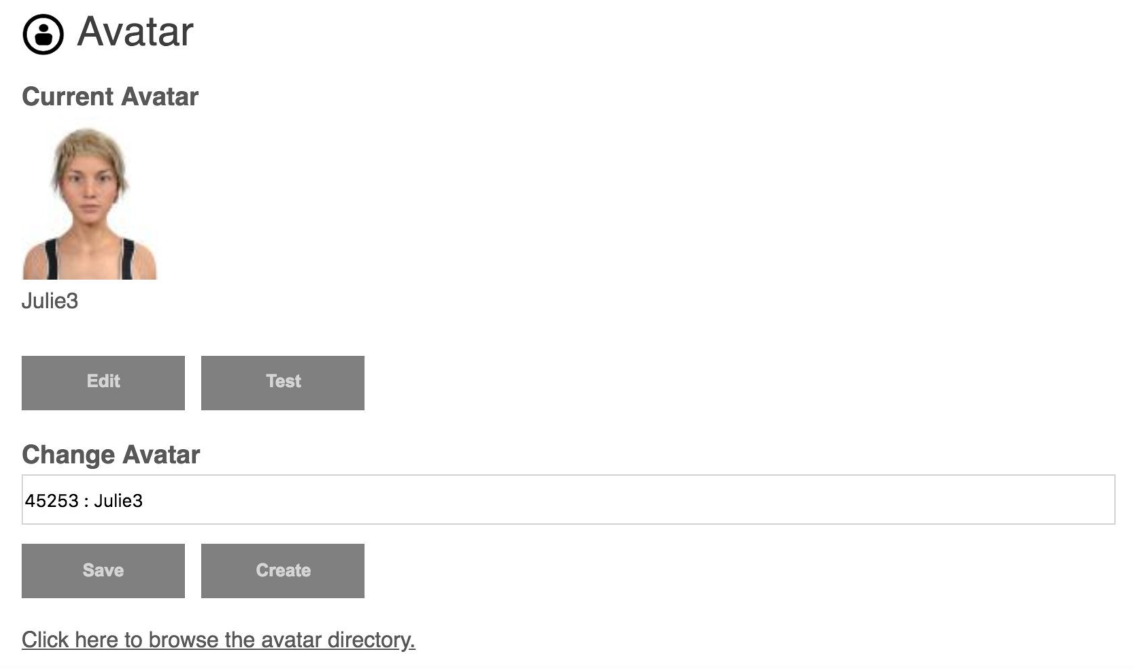
See Also

- Users

Avatars

Overview

The avatars tab allows you to add new avatar image to your bot. New avatar changes can be viewed by clicking green 'Chat' button from avatar's Info tab.



Avatar

Current Avatar

Julie3

Change Avatar

45253 : Julie3

Click here to browse the avatar directory.

Toolbar

Icon	Property Name	Description
Edit	Edit	Edit the avatar image. You can change avatar background imagae, and select different avatar emotions, actions, and poses
Test	Test	Test new avatar changes. You can observe new avatar image in a new browser window.
Save	Save	Save new avatar image changes.
Create	Create	Create new avatar image.
Input Dropdown	Change Avatar	Select different avatar image from input dropdown.
Browse Link	Browse Avatar	Browse all different avatar images from avatar's directory.

See Also

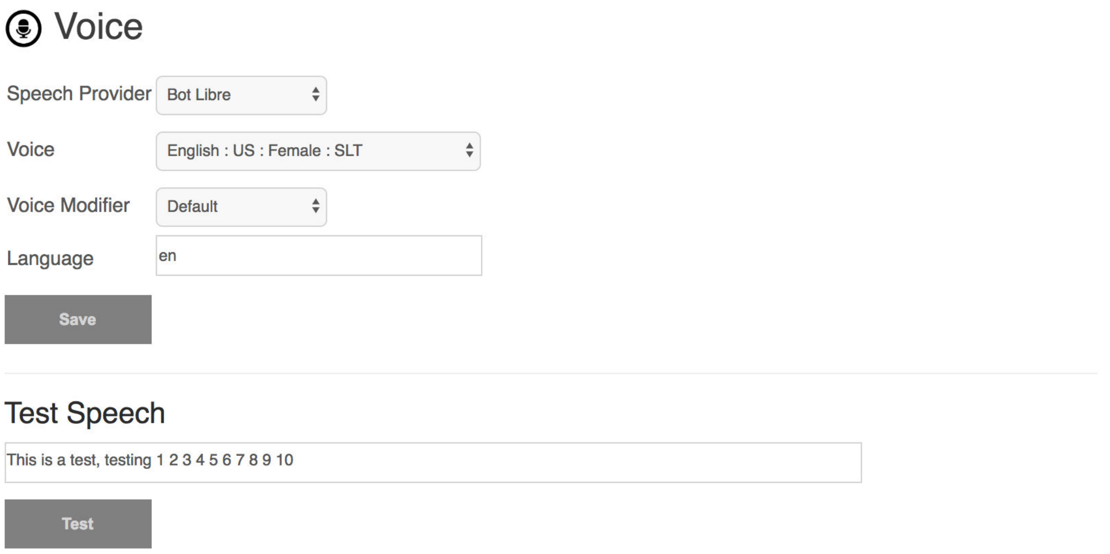
- Avatars

Voice

Overview

The voice tab allows you to select your bot's language and voice. You can use a server generated voice or native device voice. You can use a native voice on Android and iOS mobile apps, or on Chrome or other web browsers that support the HTML Speech API. This list of native voices depend on the platform and OS, and the devices configuration.

A server voice is consistent across all platforms. If you use a native voice, and the browser or platform does not support TTS, then the server voice will be used as a fall back. If a native voice is set, and it is available it will be used, otherwise any voice for the language will be used, otherwise the default native voice.



Voice

Speech Provider: Bot Libre

Voice: English : US : Female : SLT

Voice Modifier: Default

Language: en

Test Speech

This is a test, testing 1 2 3 4 5 6 7 8 9 10

Properties

Properties	Description
Server Voice	Over 30 server generated voices in several languages.
Voice Modifier	Choose a voice modifier.
Native Voice	The name of the native voice. A native mobile device or browser voice can be used on Android, iOS, Chrome, and browsers that support the HTML Speech API.
Responsive Voice	Use the third party voice API from Responsive Voice.
Language	The language code for the native voice (i.e. en-US, fr, zh).
Test	Testing/Reading the written sentence.

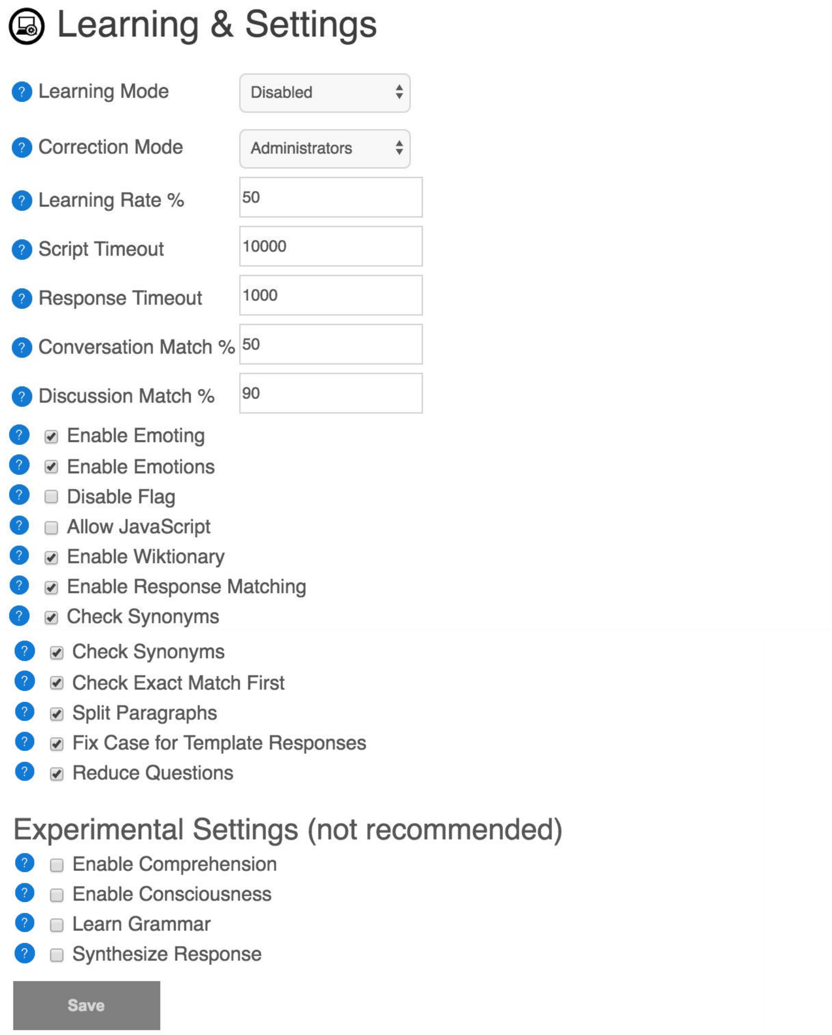
See Also

- Voice
- Add Speech to your Website using the Bot Libre TTS JavaScript SDK

Learning & Settings

Overview

The learning and settings tab allows you to configure how your bot learns and other settings. It gives several high level, and low level settings that let you customize how your bot learns and thinks. This will influence how your bot interacts with users, how it responds, how long its takes to respond, and how much memory it uses.



Learning & Settings

Learning Mode: Disabled

Correction Mode: Administrators

Learning Rate %: 50

Script Timeout: 10000

Response Timeout: 1000

Conversation Match %: 50

Discussion Match %: 90

☑ Enable Emoting
☑ Enable Emotions
☐ Disable Flag
☐ Allow JavaScript
☑ Enable Wiktionary
☑ Enable Response Matching
☑ Check Synonyms
☑ Check Synonyms
☑ Check Exact Match First
☑ Split Paragraphs
☑ Fix Case for Template Responses
☑ Reduce Questions

Experimental Settings (not recommended)

☐ Enable Comprehension
☐ Enable Consciousness
☐ Learn Grammar
☐ Synthesize Response

Learning Mode - The learning mode controls who your bot will learn responses from. When enabled your bot we learn every response to its response as a new response in context. Be careful enabling learning for service bots, as users can train your bot to have bad responses.

Correction Mode - The correction mode controls who can correct your bot's responses. Be careful enabling correction for service bots, as users can train your bot to have bad responses.

Learning Rate - The learning rate is the % to increase a response's correctness when learning. Each time your bot learns a new response to a question it will increase its correctness by this %. A response has a correctness from -100% to 100%. In conversation mode a bot will use a response with a %50 correctness (by default). The default learning rate is 50%.

Script Timeout - The script timeout (in milliseconds) gives a limit to the amount of time for script processing. If a timeout occurs, the bot will abort the script, and respond using response matching, or use a default response. This can be used to ensure the bot does not take too long to give a response. The default is 10000 (10 seconds).

Response Timeout - The response timeout (in milliseconds) gives a limit to the amount of time the bot will search for a matching response. When the bot does not know a response for a question it will search for similar questions that it does know a response to. This can be used to ensure the bot does not take too long to give a response. Smaller values make the bot respond quicker, larger values can help the bot find a better response. The default is 1000 (1 second).

Conversation Match - The conversation match % influences when the bot will use a response in a conversation. If the response's correctness is less than the %, or for a response match, if the question's % match is less than the %, then the response will not be used. If no responses match the %, then the bot will use a default response. The bot is in conversation mode for 1v1 conversations, such as chat, privates, email, twitter mentions and direct messages. The default is 50%.

Discussion Match - The discussion match % influences when the bot will use a response in a discussion. If the response's correctness is less than the %, or for a response match, if the question's % match is less than the %, then the response will not be used. If no responses match the %, then the bot will not respond. The bot is in discussion mode for chat room conversations, such as chat rooms, IRC, twitter status updates and searches. A chat room message that mentions the bot's name is treated as a conversation message, not a discussion message. The default is 90%.

Enable Emoting - Configures the ability for users to teach the bot emotes. If disabled, only administrators will be able to teach the bot emotes. An emote associates an emotion with a word or phrase and influences the bot's avatar and mood.

Enable Emotions - Configures the ability for the bot feel or associate emotions. Disabling emotions can improve the bot's performance somewhat, and prevent it from becoming self aware.

Allow JavaScript - Allow the bot's responses to contain JavaScript. Caution should be used it enabling JavaScript to prevent security issues. For security reasons JavaScript cannot be enabled if learning is enabled. JavaScript is only allowed for commercial accounts.

Enable Comprehension - Configure if the bot should attempt to identify language rules from conversations. Comprehension allows the bot to self learn template or scripted responses, such as learning to count, or inferring learned phrases like 'What is your name?' -> 'I am Jim' as Template('I am {speaker}'). When enabled comprehension will enable the bot to extend its last script with its own code. Disabling comprehension can improve performance, and ensure the boy only responds exactly as you have train it.

Enable Consciousness - Configure if the bot should have a consciousness and temporal awareness. The consciousness is used to determine the best response, or best word or meaning based on the context. Objects increase their consciousness level based on their relationship input, and fade over time. Temporal awareness associates a timeframe and temporal order for input. Disabling the consciousness can improve performance.

Enable Wiktionary - Configure if the bot should look up word definitions on Wiktionary. This helps the bot identify names, nouns, verbs, adjectives, synonyms, antonyms, and word definitions. This is used by many of the bootstrap scripts such as NounVerbAdjective and WhatIs. Currently only the English Wiktionary is used. This can be disabled for non-English bots, or to improve performance and reduce memory consumption.

Enable Response Matching - Configure if the bot should search for similar questions and responses when encountering a question it does not know a response to. This heuristic can also be influenced using the conversation/discussion match %. Responses can also be given keywords, required words, previous and topics to improve response matching.

Check Exact Match First - Configure if the bot should reply to questions with a known response before executing its scripts. This lets learned responses override scripted responses, and can improve response times for known responses.

Split Paragraphs - Configure if multi setence inputs should be split up and processed as multiple input. This means your bot's response will contain responses to each setence in the users input. Some scripts may require this to be disabled to process * patterns.

Fix Case for Template Responses - Configure if template formula responses should be fixed to use proper case. When enabled the first word will be capitalized, and other words other than names will be lower case.

Learn Grammar - Configure if word associations and grammar should be learned. When enabled words will be associated with what words come before and after them. This helps the bot choose the correct word for verbs and pronouns. This can be disabled to improve performance, or avoid the bot learning bad grammar from users.

Synthesize Response - Configure if a synthesized response should be used by default. This will have the bot generate a unique response to the question base on the question's words. A synthesized response will only be used if the bot has no response match, and has no default response. Learn grammar should be enabled for this feature.

Properties

Properties	Description
Learning Mode	Configure which type of users that bot should learn from.
Correction Mode	Configure which type of users are allowed to correct the bot's responses.
Learning Rate	The % rate to increase a response's correctness when learning.
Script Timeout	Number of milliseconds to allow for script processing.
Response Timeout	Number of milliseconds to allow for response matching.
Conversation Match	The % confidence required for a bot to use or match a response in a conversation.
Discussion Match	The % confidence required for a bot to use or match a response in a discussion.
Enable Emoting	Config if users are allow to associate emotions with responses.
Enable Emotions	Config if the bot should experience emotions.

Properties	Description
Allow JavaScript	Allow the bot's responses to contain JavaScript.
Enable Comprehension	Configure if the bot should attempt to identify language rules from conversations (disable to improve performance).
Enable Consciousness	Configure if the bot should have a consciousness and temporal awareness (disable to improve performance).
Enable Wiktionary	Configure if the bot should look up word definitions on wiktionary (English only, disable to improve performance).
Enable Response Matching	Configure if the bot should search for similar questions and responses when encountering a question it does not know a response to.
Check Exact Match First	Configure if the bot should reply to questions with a known response before executing its scripts.
Split Paragraphs	Configure if multi sentence inputs should be split up and processed as multiple input.
Fix Case for Template Responses	Configure if template formula responses should be force to use proper case.
Learn Grammar	Configure if word associations and grammar should be learn.
Synthesize Response	Configure if a synthesized response should be used if no default response.

See Also

- Create bots with a real brain

Training & Chat Logs

Overview

Your bot's conversation logs can be accessed from the 'Training & Chat Logs' menu in its Admin Console. The Training & Chat Logs page lets you review your bot's conversations, responses, words, and phrases.

The training and chat logs page allows you to add new responses, greetings, and default responses. You can view the conversations your bot has had, and correct the responses. You can view and edit responses, greetings, default responses, phrases and flagged responses.

You can correct a bot's response in a conversation, or add or edit existing responses. You can associate keywords, required words, previous responses, and topics to your bot's response to improve its response matching and conversation context. You can label responses to reuse them in other questions by referencing the #label.

You review your bot's words and phrases. You can define sentiment, emotions, actions, poses, keywords, and synonyms.

You can also import response lists, chat logs, or AIML files from the shared script library, or upload the files from your computer. You can export and download your bot's conversations or responses as a response list, chat log, or AIML file.

Toolbar

Icon	Property Name	Description
☰	Menu	Choose from the list of options.
🏠	Home	Returns to the training and chat logs start page.
⊕	Enter	Enter a new default response.
✎	Edit	Enter a correct response for the selected phrases.
🔍	Inspect	Browse the selected responses or phrases.
☑	Select	Select all conversations, responses, or phrases.
⊗	Wrong	Mark the selected responses as invalid responses, or decrease their correctness %.
✓	Check	Mark the selected responses as valid responses, or increase their correctness %.
⚑	Flag	Flag the selected phrases as offensive.
⚐	Unflag	Unflag the selected phrases as not offensive.
✖	Delete	Delete the selected responses, greeting, default response, or conversations.
⤓	Export/Download	Export and download the currently displayed logs as a chat log file, response list, or AIML script.
⤒	Upload	Upload and import a chat log, response list, or AIML script.
⇤	Import	Import a chat log, response list, or AIML script from the script library.

Search Properties

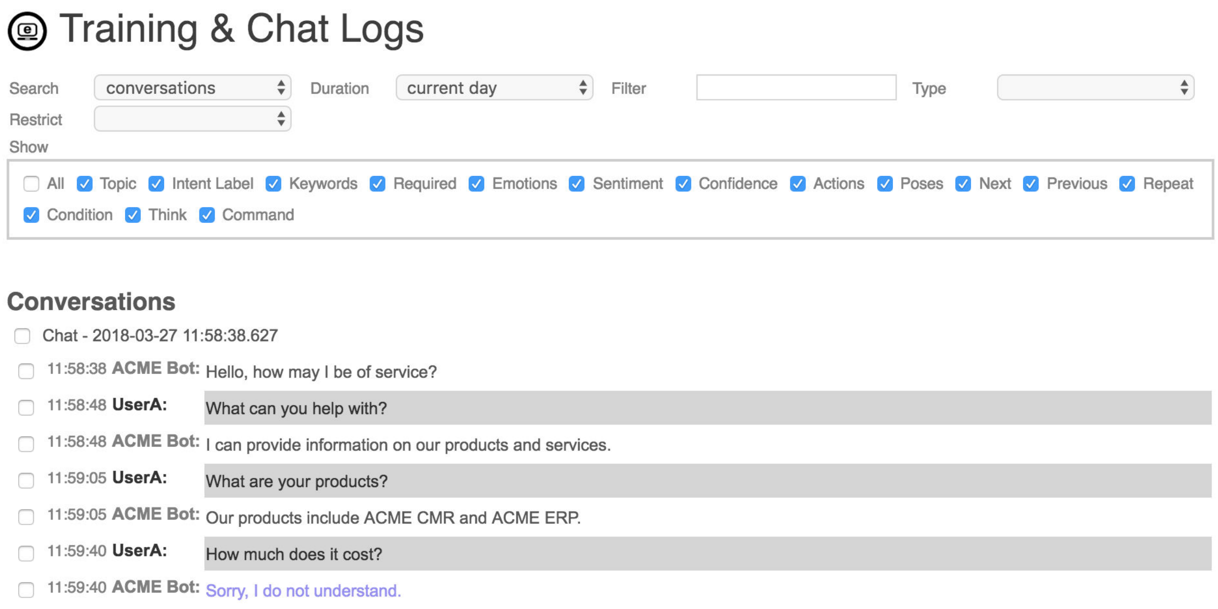
Properties	Description
Search	Search all conversions, responses, greetings, default responses, phrases, words, or flagged responses.
Duration	Filter the results by day, week, month, or all.
Filter	Filter the results to only include phrases containing the filter text.
Type	Filter by chat or social media message type.

Properties	Description
Restrict	Filter the responses or conversations.
All	Show the responses details.
Topic	Show the topic of the responses.
Label	Show the label of the responses.
Keywords	Show the question keywords to match the responses.
Required	Show the question required words to match the responses.
Emotions	Show the emotions of the responses.
Sentiment	Show the sentiment of the responses.
Confidence	Show the confidence of the responses.
Actions	Show the actions of the responses.
Poses	Show the poses of the responses.
Next	Show the next questions to the responses.
Previous	Show the previous questions to the responses.
Repeat	Show the response repeat options.
Condition	Show the response condition code.
Think	Show the response think code.
Command	Show the response JSON client command.

Conversations

You can review your bot's conversations, and correct the bot's response if it did not know the answer, or gave an incorrect response.

Browse Conversations



Training & Chat Logs

Search: conversations
Duration: current day
Filter:
Type:
Restrict:
Show

□ All ☑ Topic ☑ Intent Label ☑ Keywords ☑ Required ☑ Emotions ☑ Sentiment ☑ Confidence ☑ Actions ☑ Poses ☑ Next ☑ Previous ☑ Repeat ☑ Condition ☑ Think ☑ Command

Conversations

□ Chat - 2018-03-27 11:58:38.627

□ 11:58:38 **ACME Bot:** Hello, how may I be of service?

□ 11:58:48 **UserA:** What can you help with?

□ 11:58:48 **ACME Bot:** I can provide information on our products and services.

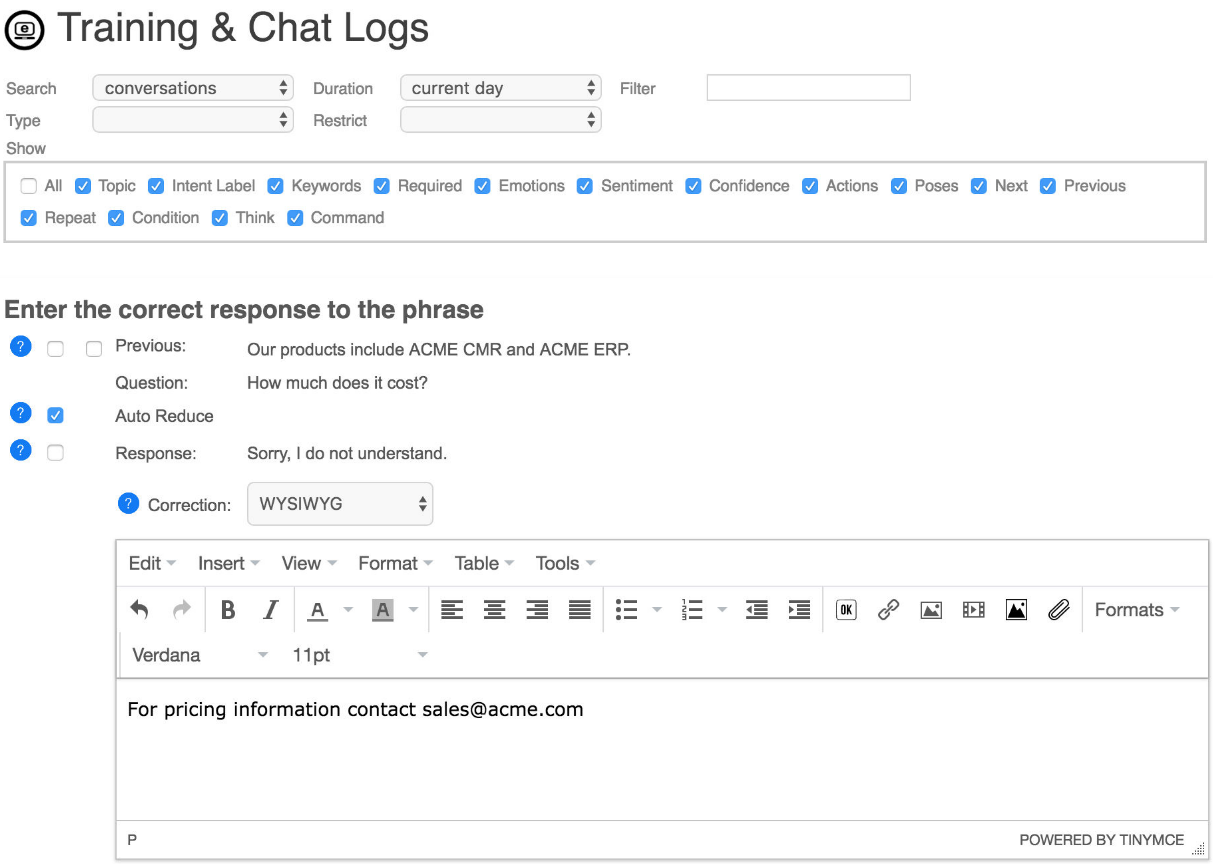
□ 11:59:05 **UserA:** What are your products?

□ 11:59:05 **ACME Bot:** Our products include ACME CMR and ACME ERP.

□ 11:59:40 **UserA:** How much does it cost?

□ 11:59:40 **ACME Bot:** Sorry, I do not understand.

Correct Response



Training & Chat Logs

Search: conversations
Duration: current day
Filter:
Type:
Restrict:

Show

☐ All ☑ Topic ☑ Intent Label ☑ Keywords ☑ Required ☑ Emotions ☑ Sentiment ☑ Confidence ☑ Actions ☑ Poses ☑ Next ☑ Previous ☑ Repeat ☑ Condition ☑ Think ☑ Command

Enter the correct response to the phrase

? ☐ ☐ Previous: Our products include ACME CMR and ACME ERP.

Question: How much does it cost?

? ☑ Auto Reduce

? ☐ Response: Sorry, I do not understand.

? Correction: WYSIWYG

For pricing information contact sales@acme.com

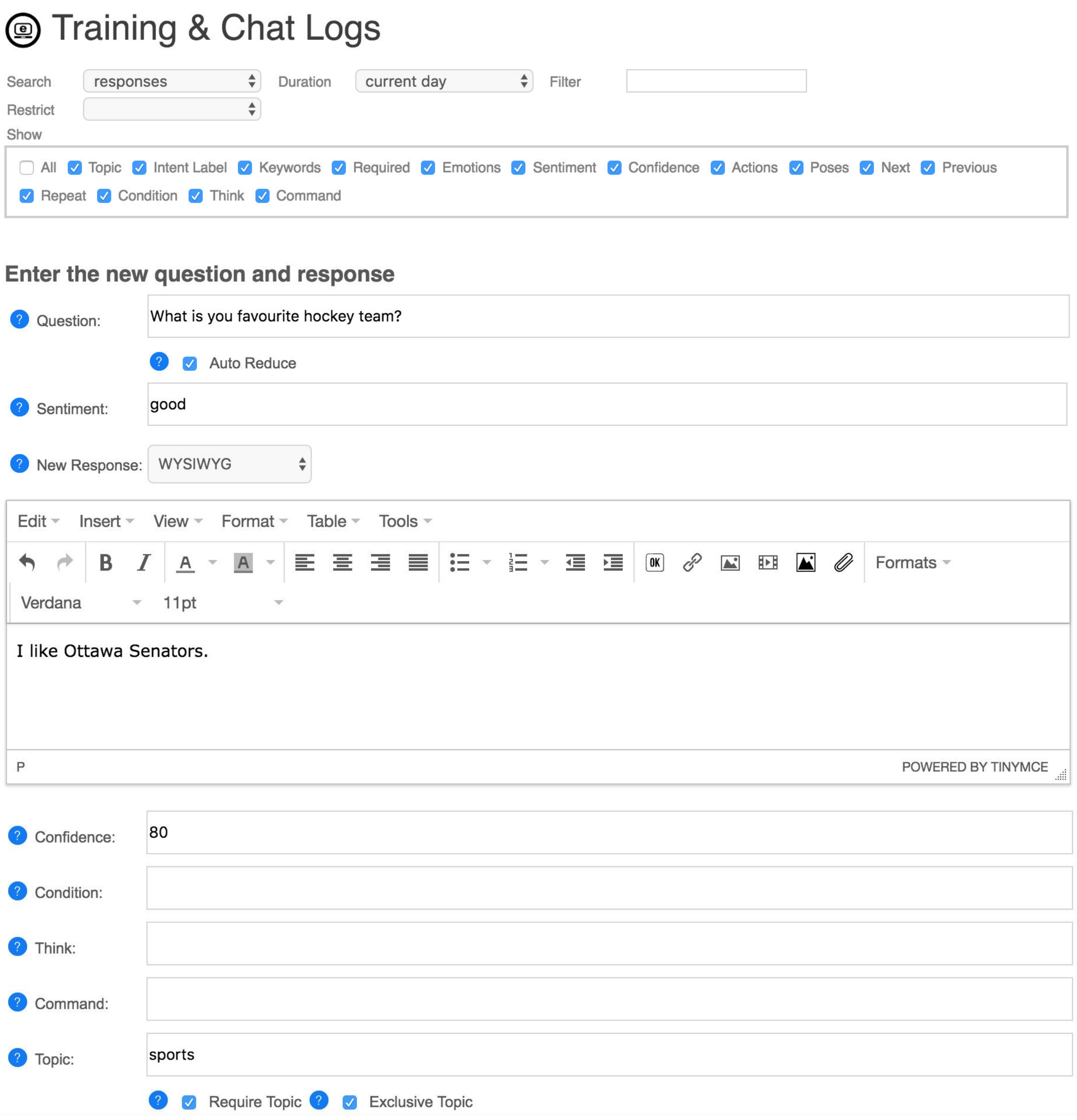
Responses

You can add new responses, or edit existing responses. A response includes the question, the response, and various meta data about the response.

To see all of your bot's trained responses, search for responses for the all duration. This will list all of your bots responses, ordered by when they were added. A response will not only be used for the exact question it was learned for, but also any similar questions. When your bot encounters a question it does not know a response to, it will find the best matching question that it knows a response to, and use that response. If the bot does not know any similar questions, then it will use its default response.

The response browser lets you enter a new response, or edit, delete, invalidate, validate, or flag any existing responses.

The response editor lets you enter the question and response and optional response properties and metadata. You can choose to edit your response using a rich 'WYSIWYG' (what you see is what you get) editor, or a plain text editor. Responses are in HTML, and can contains most HTML tags such as <b> for bold, <a> for links, <img> for images, and <button> for buttons.



Training & Chat Logs

Search: responses
Duration: current day
Filter:
Restrict:

Show

□ All ☑ Topic ☑ Intent Label ☑ Keywords ☑ Required ☑ Emotions ☑ Sentiment ☑ Confidence ☑ Actions ☑ Poses ☑ Next ☑ Previous ☑ Repeat ☑ Condition ☑ Think ☑ Command

Enter the new question and response

Question: What is you favourite hockey team?

☑ Auto Reduce

Sentiment: good

New Response: WYSIWYG

Verdana 11pt

I like Ottawa Senators.

P

Confidence: 80

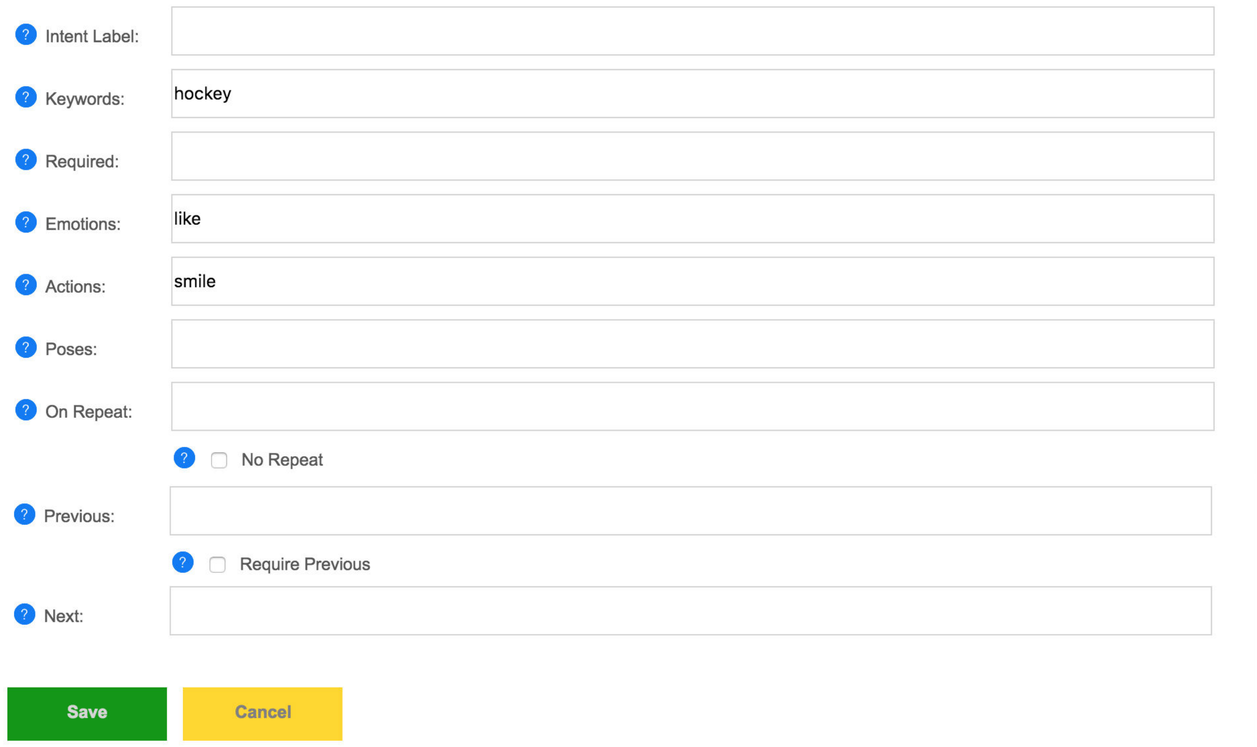
Condition:

Think:

Command:

Topic: sports

☑ Require Topic ☑ Exclusive Topic



Intent Label:

Keywords: hockey

Required:

Emotions: like

Actions: smile

Poses:

On Repeat:

☐ No Repeat

Previous:

☐ Require Previous

Next:

Response Properties

Name	Description
Question	Edit the question, if desired, you can use * wildcard to make a pattern.
Sentiment	You can associate an emotion with the phrase.
New Response	Enter a new response for the phrase.
Condition	Optionally you can give a condition in self code that must evaluate to true for the response to be use.
Think	Optionally you can give a 'think' code in self that is evaluated when the response is used.
Command	Optionally you can give a JSON command (Self code) that is evaluated and returned to the client to support games and virtual assistance.
Topic	Optionally you can give a topic to categorize the response under.
Intent Label	Optionally you can give a label to reuse the response as.
Keywords	Optionally you can give keywords from the question that will influence a response match.
Required	Optionally you can give required words from the question that will be required for a response match.
Emotions	You can associate an emotion with the response.
Actions	You can associate an action with the response.
Poses	You can associate a pose with the response.
On Repeat	Optionally you can give a response to used if the current response has already been used in this conversation.
No Repeat	Require that the response or phrase only be used once.
Next	Optionally you can give a next response to give a response a context.
Previous	Optionally you can give a previous response to give a response a context.
Require Previous	Require that the response only be used if the previous response matches one of the previous responses.
Save	Save the corrections to the phrases.

Question

The response question can be a phrase that will be automatically matched with similar questions, or can be a Pattern. Patterns are matched literally in the order of the words in the pattern, and can have wildcard characters such as * and can also include Regular Expressions. Normally it is best to use a phrase, and let the bot decide if a question should match, you can also add keywords and required words to improve when a question is matched. To enter a pattern just use a wildcard such as * in the question, or enter your pattern like the following.

```
Pattern("What is *")
```

Response

The response can be a phrase, can contain HTML or rich content including images, video, and buttons, or a response can be a Template. A template is a response that can include some Self code. Any code inside {} in a template will be executed, and the result printed into the response. To enter a template just include {} in your response, or enter your template like the following (do not use the WYSIWYG editor for tempates).

```
Template("Today is {Date.date()}.")
```

Keywords

Keywords are important words in the question that should influence when the response is used.

```
// Keywords are used to allow this response to be used for any questions on hockey or spinach.
question: do you like hockey
response: Hockey is my favorite sport.
keywords: hockey

question: what do you love
response: I love hockey!
keywords: love luv like adore

question: do you like field hockey
response: I only play ice hockey.
keywords: "field hockey"

question: i love spinach
response: Spinach is is my favorite food.
keywords: spinach
```

A keyword will be used by the bot to select the best response. Keywords let some words count for more than other words. In the example, the bot knows two responses, how will it respond to the phrase "I love hockey"? Without the keywords the bot would probably match this question with "I love spinach", but since hockey is a keyword, it will instead match it with "Do you like hockey?".

If the response has no defined required words, then the keywords will also be required for the response match. So "I love hockey" would not be matched with "I love spinach." because it is missing the keyword.

If you keywords has many similar words. You can list them all as keywords to match any of the words. You could also define these words a synonyms, which may be a better option.

You can also use compound words are keywords. A compound keyword must have all the words in the order to be used. For a compound keyword just wrap the words in "quotes".

Be careful using too many keywords. If every word is a keyword, then their value is diminished. Never make common words like "the", "a", "he", "what" keywords, if you require one of these words for a match use a required word instead.

You can also review all of your bot's keywords by searching for 'words' and restrict to 'keywords'. This lets you remove a word from being a keyword if you added it by mistake.

Required

A required word is a word that is required to be in the question for a specific response.

```
// This example requires the words 'like' and 'hockey'.
question: do you like hockey
response: Yes, I like hockey.
required: like hockey

// This example requires the compound word 'ice hockey'.
question: do you like ice hockey
response: Yes, I like ice hockey.
required: "ice hockey"
```

```
// This example requires one of the words 'like' or 'love', and 'hockey'.
question: do you like ice hockey
response: Yes, I like ice hockey.
required: (like love) hockey

// This example requires a pattern.
question: are you okay
response: Yes, I am okay.
required: Pattern("^ are you (ok okay)")
```

A required word can be used to ensure the bot does not choose a response if the required word is missing from the question. For example the question "Do you play hockey?" would normally be matched to this response, but because it is missing the required word "like", it will not be considered as a match.

All of the required words must be contained in the question in order for the response to be used. Required words also supports lists, compound words, and patterns. To require one of a set of words a list can be used using brackets i.e. (like love). To require a compound word quotes are used i.e. "hello world". To require a pattern enter the pattern i.e. Pattern("what is *").

Previous

A previous responses is the bot's preceding response. Previous responses can be used to give a response a context.

```
// This example uses previous to give different answers to 'yes' depending on the context.
question: yes
response: Great, wanna go out sometime?
require previous: Are you single?

question: yes
response: I am please to hear that.
require previous: Are you happy?
```

The question "Yes" has multiple responses. The bot can use the previous response to choose the best response. You can also invalidate a previous response to ensure the bot does not use a response if the previous response was preceding.

A previous response can either be 'required' or optional. If required, the response will never be used unless one of the previous response matches. If optional (default) the response is given precedence if the previous response matches, but can still be used if the previous does not match.

Topic

A topic is the general topic category for the response. For example a response on a hockey related question would belong to the "hockey" topic. You can use topics to categories your responses and give the bot additional context. If the bot has multiple responses to a question, it will use the one that matches the current topic.

Topics can be either optional or required. The response topic is optional by default, it you click on 'Require Topic', then the response will only be used if that topic is active in the conversation. If optional the topic will be set as active for the conversation, until a new topic is set.

Topics can also be defined as exclusive. An exclusive topic defines an isolated conversational space. For a normal topic that is not exclusive the bot can still use responses that do not have the same topic, but responses with the active topic will be given precedence over responses without a topic. For exclusive topics only response that have the same topic as the active topic will be used. No other responses will be used (other than possibly the default response if there is no topic default).

Topics can also be set on greetings to start the conversation with a topic. Topics can be set on default responses to provide a different response when that topic is active.

```
// Categorizes the product response.
question: what are your products
response: Our products include ACME CRM and ACME ERP.
topic: products

// Use an exclusive topic to define a joke conversational dialog.
question: joke
response: Knock knock.
```

```
exclusive topic: joke

question who is there
response: Boo.
require topic: joke

question: boo who
response: Don't cry. lol
require topic: joke
think: conversation.topic = null;

default: Say 'Who is there?'
require topic: joke

default: Say 'Boo who?'
require topic: joke
previous: boo
```

Intent Labels

A response can be given an intent label to let you reuse the same response in other questions. You can also use labels in a response's previous.

```
// Use a label to reuse the common products response.
question: products
response: Our products include ACME CRM and ACME ERP.
label: #products

question: what are your products
response: #products

question: what software do you sell
response: #products

question: are they good
response: Our products are the best in the world.
previous: #products
```

On Repeats

If the user asks the same question, or similar questions multiple times, you may want your bot to give a different response the second time. On repeat lets you set one or many responses to use the second time that response is triggered in the same conversation. You can also set 'No Repeat' to never repeat the response in the same conversation.

```
// Offer to escalate the issue if the user repeatedly asks for help.
question: help
response: How can I help you?
on repeat: What is the issue that you are having?
on repeat: Would you like me to have a support staff contact you?

question: goodbye
response: Before you go, would you like to take a survey?
no repeat:

question: goodbye
response: Goodbye.
```

Conditions

Conditions let you enter Self code that must evaluate to true for the response to be used. This is an advanced properties as it requires you understand Self (a dialect of JavaScript), but can be used to do very powerful things.

```
// Response to goodbye differently based on the time of day.
question: goodbye
response: Goodnight.
condition: Date.get(#hour, Date.time()) > 18

question: goodbye
response: Goodbye.
```

Think

Think let you enter Self code that is executed when the response is used. This is an advanced properties as it requires you understand Self (a dialect of JavaScript), but can be used to do very powerful things.

```
// Track the user's name.
pattern: my name is *
template: Pleased to meet you {star}.
think: speaker.name = star;

question: what is my name
template: Your name is {speaker.name}.

question: can we change the subject
response: Sure, what would you like to talk about?
think: conversation.topic = null;
```

Confidence

You can enter multiple responses for the same question. The bot will use the response that it is the most confident in for the current conversational context. By default a response is given a confidence level of 90%, but you can customize the value. Note that in a group discussion such as when the bot is added to a chatroom, or on Twitter, the bot will only use responses that have >=90% confidence.

Sentiment

You can associate a user question or phrase, or individual words with sentiment (good vs bad). Sentiment can be used to track how the user is feeling about the current conversation. The sentiment statistic is tracked by your bot daily and can be graphed from its Analytics page.

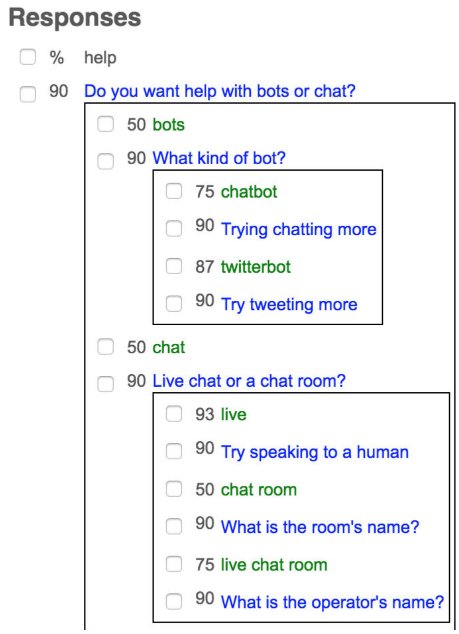
```
question: thank you
response: You are most welcome.
keywords: thank
sentiment: good

question: you suck
response: Sorry, I am doing my best. Would you like me have someone contact you via email?
keywords: suck
sentiment: bad

word: suck
sentiment: bad

word: great
sentiment: great
```

Next and Conversation Flows

Next questions in a response can be used to handle follow-up questions and conversation flows. To start a conversation flow, edit the root response and enter the follow-up question under 'Next'. You can then select the next question and edit the response. You can repeat this several times to define your complete conversation.

A respones next question defines an isolated conversation space. The bot will only choose its next response from the provided responses. To define a default response use the #default as the question. If the bot cannot match the user's question with a next question, it will search the responses parent, otherwise use the bot's default response.

Next is similar to previous, but unlike previous is isolated to the response. In the context of a next response, the bot will never use any of its responses outside the context of the next response. This makes is easy to define isolated and nested conversations.

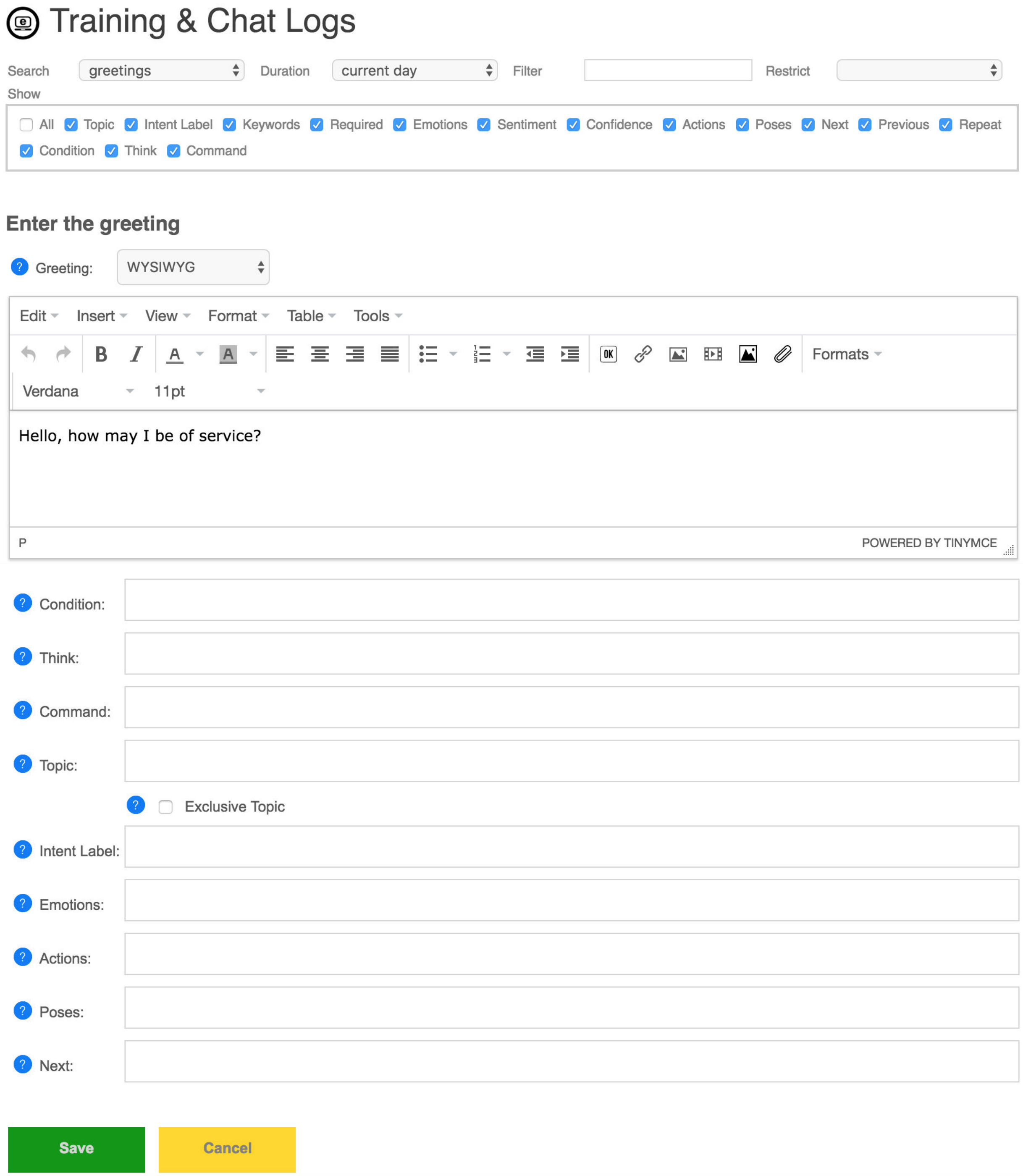


Responses

- ☐ % help
- ☐ 90 Do you want help with bots or chat?
  - ☐ 50 bots
  - ☐ 90 What kind of bot?
    - ☐ 75 chatbot
    - ☐ 90 Trying chatting more
    - ☐ 87 twitterbot
    - ☐ 90 Try tweeting more
  - ☐ 50 chat
  - ☐ 90 Live chat or a chat room?
    - ☐ 93 live
    - ☐ 90 Try speaking to a human
    - ☐ 50 chat room
    - ☐ 90 What is the room's name?
    - ☐ 75 live chat room
    - ☐ 90 What is the operator's name?

Greetings

A greeting is the bot's first response in a new conversation. You can add new greeting, or edit existing greeting. A greeting includes various meta data about the greeting.



Training & Chat Logs

Search: greetings Duration: current day Filter: Restrict:

Show: □ All ☑ Topic ☑ Intent Label ☑ Keywords ☑ Required ☑ Emotions ☑ Sentiment ☑ Confidence ☑ Actions ☑ Poses ☑ Next ☑ Previous ☑ Repeat ☑ Condition ☑ Think ☑ Command

Enter the greeting

Greeting: WYSIWYG

Hello, how may I be of service?

Condition:

Think:

Command:

Topic:

□ Exclusive Topic

Intent Label:

Emotions:

Actions:

Poses:

Next:

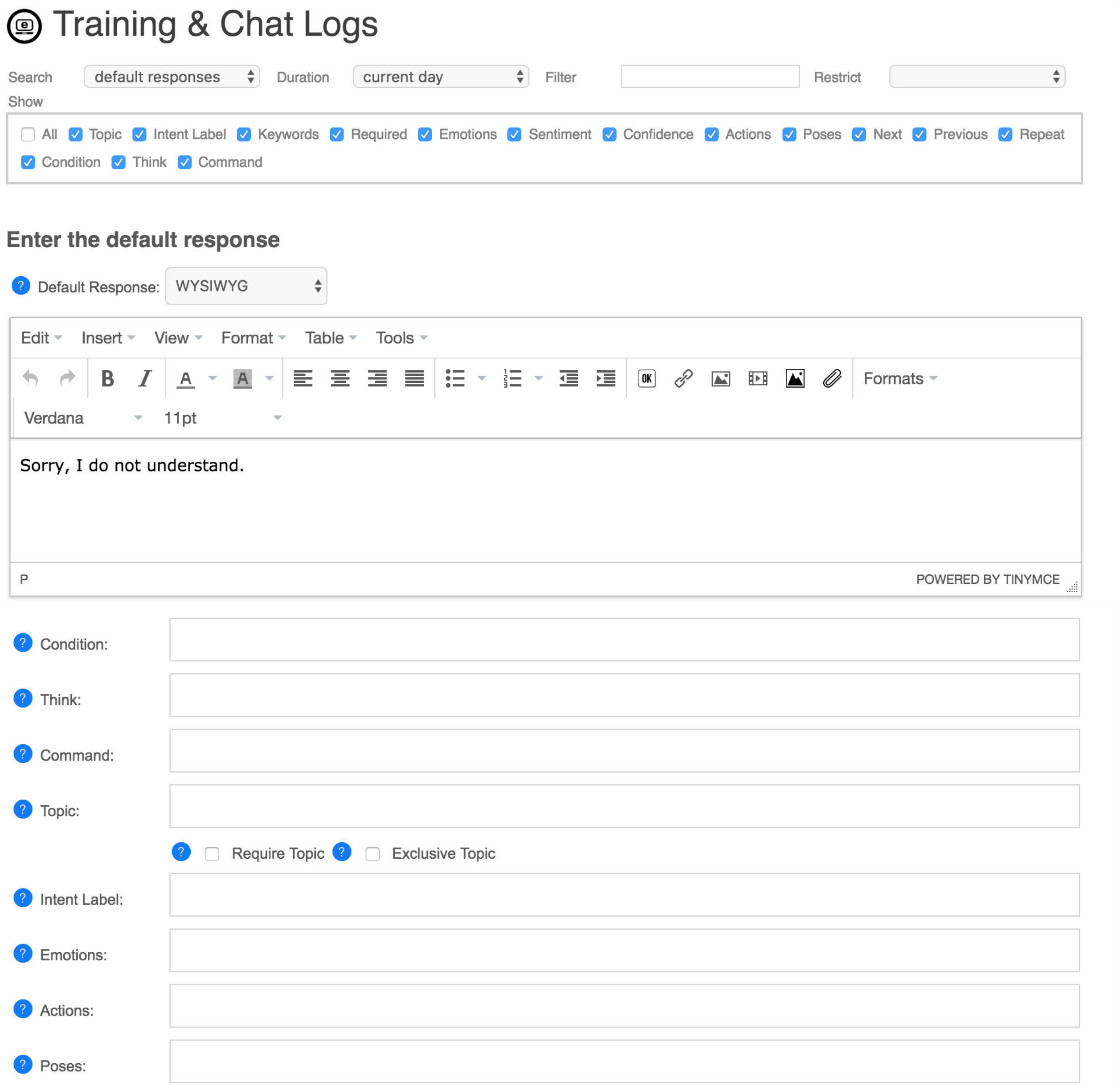
Greeting Properties

Name	Description
Greeting	Edit the greeting.
Condition	Optionally you can give a condition in self code that must evaluate to true for the response to be use.
Think	Optionally you can give a 'think' code in self that is evaluated when the response is used.
Command	Optionally you can give a JSON command (Self code) that is evaluated and returned to the client to support games and virtual assistance.
Topic	Optionally you can give a topic to categorize the response under.
Exclusive Topic	Optionally you can set an exclusive topic to a conversation so that only responses that have this topic will be used.
Intent Label	Optionally you can give a intent label to reuse the response as.
Emotions	You can associate an emotion with the greeting.
Actions	You can associate an action with the greeting.
Poses	You can associate a pose with the greeting.

Name	Description
Next	Optionally you can give a next greeting to give a response a context.
Save	Save the corrections to the phrases.

Default Responses

If the bot does not find any good matching question to a user's input, it uses its default response. You can add new default response, or edit existing defalult response. Default response includes various meta data about the response.



Training & Chat Logs

Search: default responses Duration: current day Filter: Restrict:

Show

□ All ☑ Topic ☑ Intent Label ☑ Keywords ☑ Required ☑ Emotions ☑ Sentiment ☑ Confidence ☑ Actions ☑ Poses ☑ Next ☑ Previous ☑ Repeat ☑ Condition ☑ Think ☑ Command

Enter the default response

Default Response: WYSIWYG

Sorry, I do not understand.

Condition:

Think:

Command:

Topic:

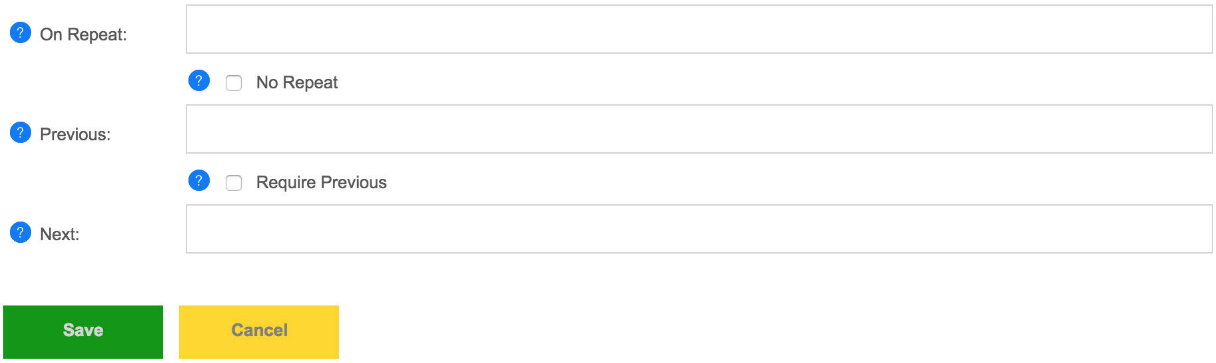
□ Require Topic □ Exclusive Topic

Intent Label:

Emotions:

Actions:

Poses:



On Repeat:

□ No Repeat

Previous:

□ Require Previous

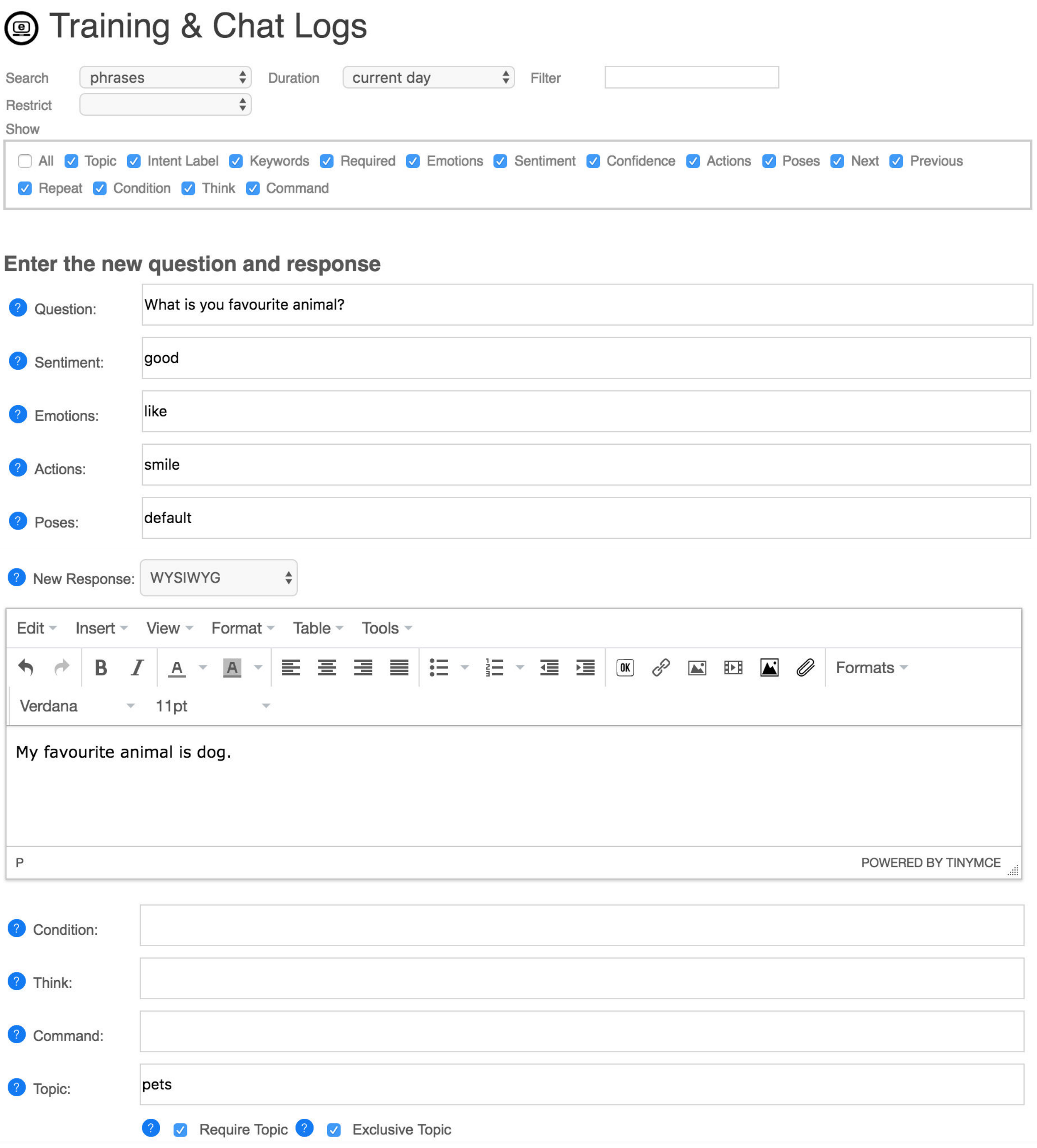
Next:

Default Response Properties

Name	Description
Default Response	Edit the default response.
Emotions	You can associate an emotion with the default response.
Actions	You can associate an action with the default response.
Poses	You can associate a pose with the default response.
Condition	Optionally you can give a condition in self code that must evaluate to true for the response to be use.
Think	Optionally you can give a 'think' code in self that is evaluated when the response is used.
Command	Optionally you can give a JSON command (Self code) that is evaluated and returned to the client to support games and virtual assistance.
Topic	Optionally you can give a topic to categorize the response under.
Label	Optionally you can give a label to reuse the response as.
On Repeat	Optionally you can give a response to used if the current response has already been used in this conversation.
No Repeat	Require that the response or phrase only be used once.
Previous	Optionally you can give a previous response to give a response a context.
Require Previous	Require that the response only be used if the previous response matches one of the previous responses.
Save	Save the corrections to the phrases.

Phrases

You can add new phrase, or edit existing phrase. A phrase is just a sentence, it may or may not have a response. The phrase editor allows you to associate sentiment, emotions, action, poses, or a response to a phrase.



Training & Chat Logs

Search: phrases
Duration: current day
Filter:
Restrict:

Show

☐ All ☑ Topic ☑ Intent Label ☑ Keywords ☑ Required ☑ Emotions ☑ Sentiment ☑ Confidence ☑ Actions ☑ Poses ☑ Next ☑ Previous ☑ Repeat ☑ Condition ☑ Think ☑ Command

Enter the new question and response

Question: What is you favourite animal?

Sentiment: good

Emotions: like

Actions: smile

Poses: default

New Response: WYSIWYG

My favourite animal is dog.

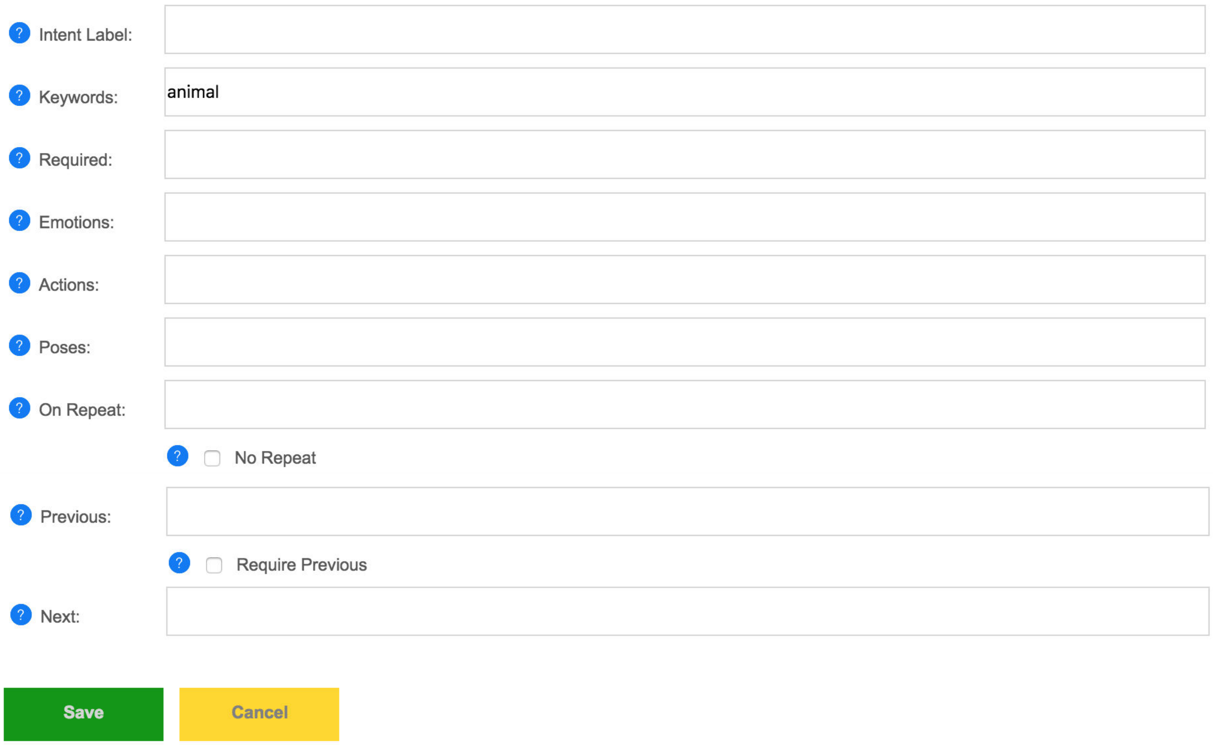
Condition:

Think:

Command:

Topic: pets

☑ Require Topic ☑ Exclusive Topic



Intent Label:

Keywords: animal

Required:

Emotions:

Actions:

Poses:

On Repeat:

□ No Repeat

Previous:

□ Require Previous

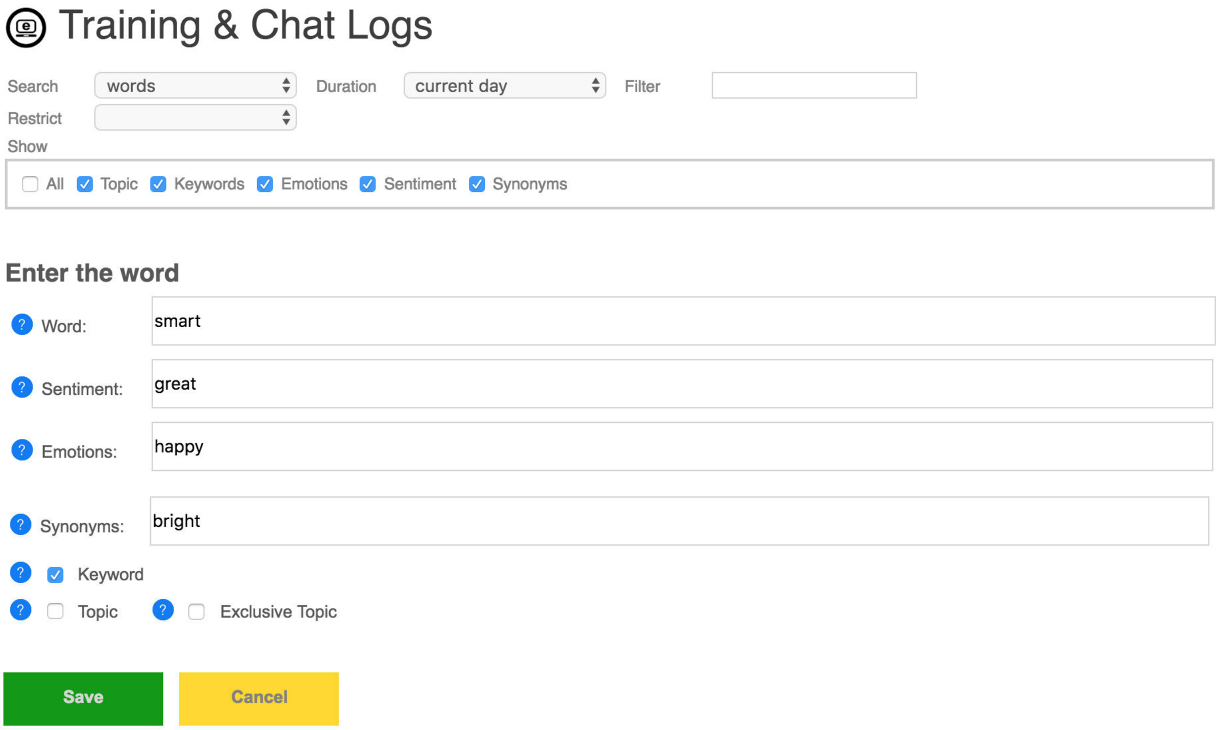
Next:

Phrase Properties

Name	Description
Question	Edit the question, if desired, you can use * wildcard to make a pattern.
Sentiment	You can associate a sentiment with the phrase.
Emotions	You can associate an emotion with the phrase.
Actions	You can associate an action with the phrase.
Poses	You can associate a pose with the phrase.
New Response	Enter a new response for the phrase.
Condition	Optionally you can give a condition in self code that must evaluate to true for the response to be use.
Think	Optionally you can give a 'think' code in self that is evaluated when the response is used.
Command	Optionally you can give a JSON command (Self code) that is evaluated and returned to the client to support games and virtual assistance.
Topic	Optionally you can give a topic to categorize the response under.
Intent Label	Optionally you can give a label to reuse the response as.
Keywords	Optionally you can give keywords from the question that will influence a response match.
Required	Optionally you can give required words from the question that will be required for a response match.
Emotions	You can associate an emotion with the response.
Actions	You can associate an action with the response.
Poses	You can associate a pose with the response.
On Repeat	Optionally you can give a response to used if the current response has already been used in this conversation.
No Repeat	Require that the response or phrase only be used once.
Next	Optionally you can give a next response to give a response a context.
Previous	Optionally you can give a previous response to give a response a context.
Require Previous	Require that the response only be used if the previous response matches one of the previous responses.
Save	Save the corrections to the phrases.

Words

You can add new words, or edit existing words. A word can include sentiment, emotions, and synonyms. A word can also be a keyword, a topic, or an exclusive topic.



Training & Chat Logs

Search: words

Duration: current day

Filter:

Restrict:

Show: □ All ☑ Topic ☑ Keywords ☑ Emotions ☑ Sentiment ☑ Synonyms

Enter the word

Word: smart

Sentiment: great

Emotions: happy

Synonyms: bright

☑ Keyword

□ Topic □ Exclusive Topic

Word Properties

Name	Description
Word	Edit the word.
Sentiment	Optionally you can associate a sentiment with the word.
Emotions	Optionally you can associate an emotion with the word.
Keyword	Define the word to be a keyword.
Synonyms	Optionally you can give a list of synonyms to the word, so that similar words in a question to trigger the same response.
Topic	Define the word to be a topic.
Exclusive Topic	Define the word to be an exclusive topic.
Save	Save the corrections to the word.

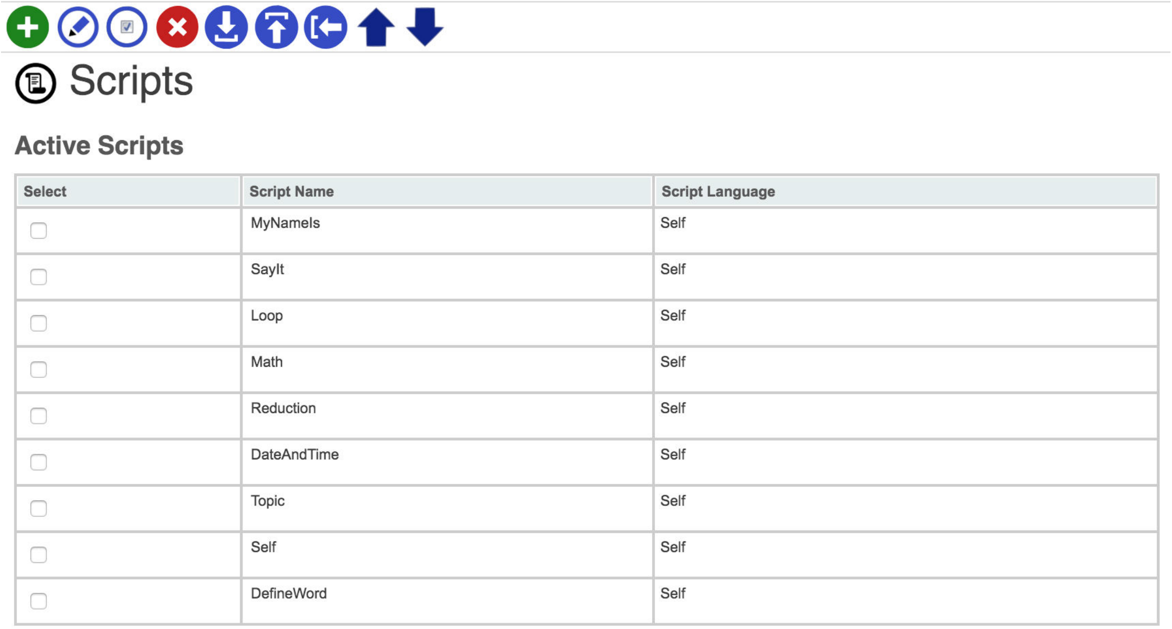
See Also

- How to train your customer service bot by monitoring its chat logs, using keywords and topics.
- What are the supported response and chat log formats for importing and exporting?
- What are topics?
- What are labels, and how to reuse responses?
- What are previous responses?
- What are required words?
- What are keywords?

Scripts

Overview

The scripts tab allows you to add scripts to your bot. This table shows all bot's active scripts.



Scripts

Active Scripts

Select	Script Name	Script Language
□	MyNameIs	Self
□	SayIt	Self
□	Loop	Self
□	Math	Self
□	Reduction	Self
□	DateAndTime	Self
□	Topic	Self
□	Self	Self
□	DefineWord	Self

Toolbar

Icon	Property Name	Description
⊕	Add	Add the new script.
✎	Edit	Add the selected script.
☑	Select	Select or unselect all scripts.
✖	Remove	Remove all selected scripts.
⬇	Downlad	Download a Self script, or AIML script.
⬆	Upload	Upload and import a Self script, or AIML script.
⇐	Import	Import a Selft script, or AIML script from the script library.
⬆	Up	Move all selected scripts order up.
⬇	Down	Move all selected scripts order down.

Bootstrap

Rebootstrap

Rebootstrap permanently deletes all scripts (state machines) and rebootstraps with the system defaults.

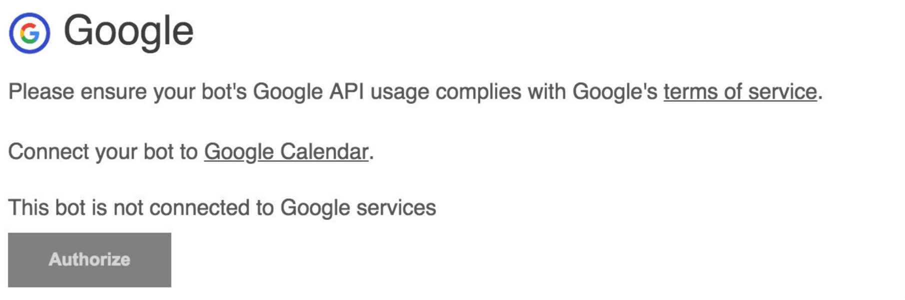
See Also

- Self
- AIML
- Introducing the Self Scripting Language
- Scripting your bot with AIML
- What classes are supported in Self

Google

Overview

The Google tab allows you to connect your bot to Google services such as Google Calendar. You can access Google services from Self scripts using the Google and GoogleCalendar classes.



Google

Please ensure your bot's Google API usage complies with Google's terms of service.

Connect your bot to Google Calendar.

This bot is not connected to Google services

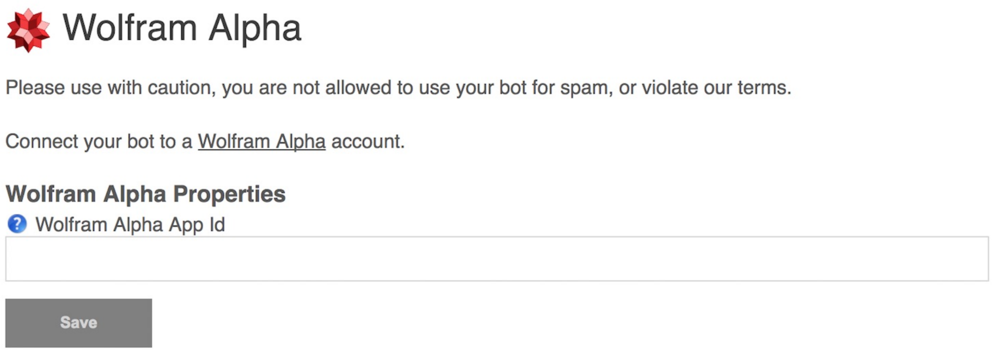
See Also

- Admin Console - Google
- Google API Console
- Google Calendar API
- Google APIs Terms of Service
- Google Calendar

Wolfram Alpha

Overview

You can connect your bot to Wolfram Alpha to answer questions.



Wolfram Alpha

Please use with caution, you are not allowed to use your bot for spam, or violate our terms.

Connect your bot to a Wolfram Alpha account.

Wolfram Alpha Properties

Wolfram Alpha App Id

Properties

Properties	Description
Wolfram Alpha App Id	Enter the App Id from the Wolfram Alpha website.

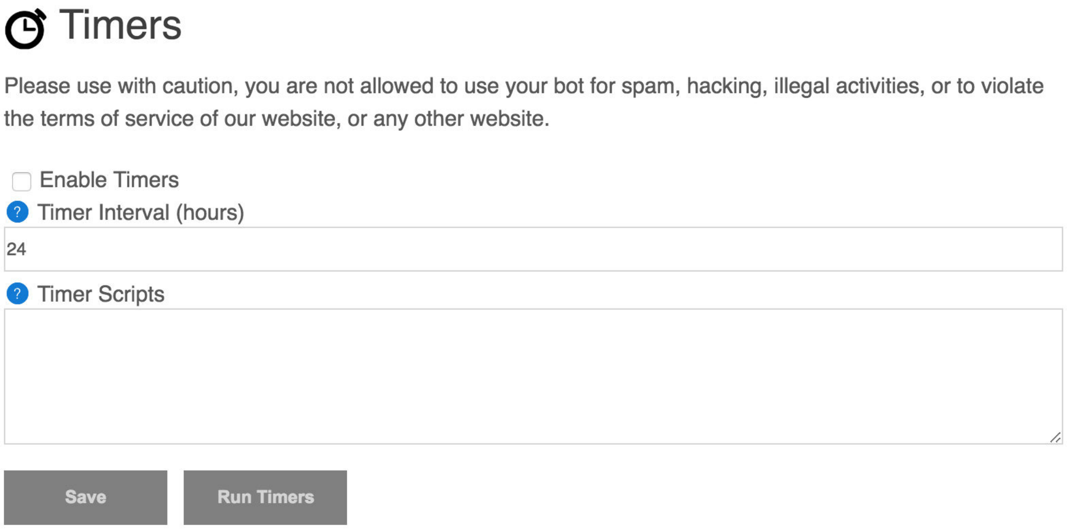
See Also

- How To Connect a Bot to Wolfram Alpha
- Wolfram Alpha

Timers

Overview

The Timers tab allows you to setup your bot to run scripts at various time intervals. This can be used to automate tasks such as polling websites, mailing/message lists, push notifications, or content services.



Timers

Please use with caution, you are not allowed to use your bot for spam, hacking, illegal activities, or to violate the terms of service of our website, or any other website.

☐ Enable Timers

Timer Interval (hours): 24

Timer Scripts:

Properties

Property	Description
Enable Timers	Enable the timers.
Auto Post Hours	The number of hours to wait between auto posts.
Timer Scripts	Set of timer messages. List each message separated by a new line.

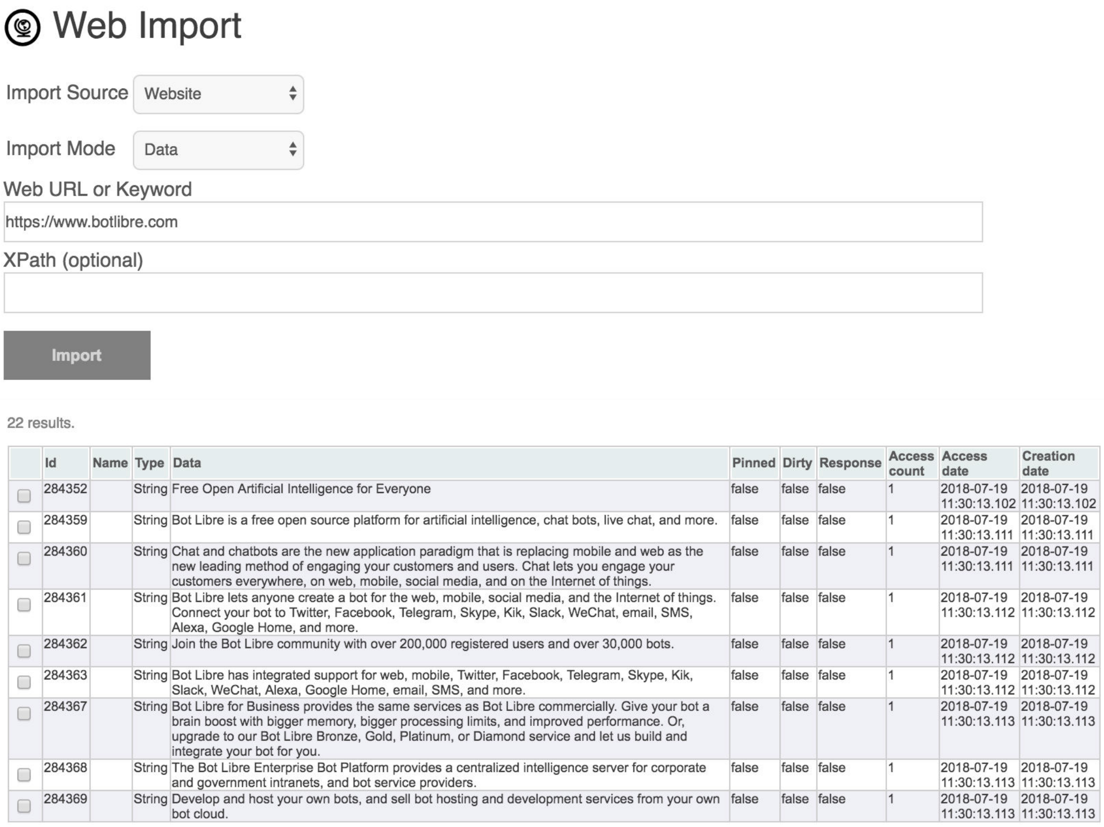
See Also

- Timers

Web Import

Overview

The web tab allows you to import data from websites such as Wiktionary, and WikiData.



Web Import

Import Source: Website

Import Mode: Data

Web URL or Keyword: https://www.botlibre.com

XPath (optional):

22 results.

	Id	Name	Type	Data	Pinned	Dirty	Response	Access count	Access date	Creation date
□	284352		String	Free Open Artificial Intelligence for Everyone	false	false	false	1	2018-07-19 11:30:13.102	2018-07-19 11:30:13.102
□	284359		String	Bot Libre is a free open source platform for artificial intelligence, chat bots, live chat, and more.	false	false	false	1	2018-07-19 11:30:13.111	2018-07-19 11:30:13.111
□	284360		String	Chat and chatbots are the new application paradigm that is replacing mobile and web as the new leading method of engaging your customers and users. Chat lets you engage your customers everywhere, on web, mobile, social media, and on the Internet of things.	false	false	false	1	2018-07-19 11:30:13.111	2018-07-19 11:30:13.111
□	284361		String	Bot Libre lets anyone create a bot for the web, mobile, social media, and the Internet of things. Connect your bot to Twitter, Facebook, Telegram, Skype, Kik, Slack, WeChat, email, SMS, Alexa, Google Home, and more.	false	false	false	1	2018-07-19 11:30:13.112	2018-07-19 11:30:13.112
□	284362		String	Join the Bot Libre community with over 200,000 registered users and over 30,000 bots.	false	false	false	1	2018-07-19 11:30:13.112	2018-07-19 11:30:13.112
□	284363		String	Bot Libre has integrated support for web, mobile, Twitter, Facebook, Telegram, Skype, Kik, Slack, WeChat, Alexa, Google Home, email, SMS, and more.	false	false	false	1	2018-07-19 11:30:13.112	2018-07-19 11:30:13.112
□	284367		String	Bot Libre for Business provides the same services as Bot Libre commercially. Give your bot a brain boost with bigger memory, bigger processing limits, and improved performance. Or, upgrade to our Bot Libre Bronze, Gold, Platinum, or Diamond service and let us build and integrate your bot for you.	false	false	false	1	2018-07-19 11:30:13.113	2018-07-19 11:30:13.113
□	284368		String	The Bot Libre Enterprise Bot Platform provides a centralized intelligence server for corporate and government intranets, and bot service providers.	false	false	false	1	2018-07-19 11:30:13.113	2018-07-19 11:30:13.113
□	284369		String	Develop and host your own bots, and sell bot hosting and development services from your own bot cloud.	false	false	false	1	2018-07-19 11:30:13.113	2018-07-19 11:30:13.113

Properties

Icon	Property Name	Description
☰	Menu	Choose icons from Menu.
🔍	Inspect	Inspect the selected objects.
⋖	Knowledge	Browse all objects that reference the selected objects.
☑	Select	Select all (or the first 100) objects.
📌	Pin	Pin the selected objects, so they cannot be forgotten.
🚫	Unpin	Unpin the selected objects, so they cannot be forgotten.
✖	Delete	It will permanently delete the selected objects or relationships.
⬇	Export	Export the objects to JSON or CSV (spreadsheet)

See Also

- Web Import
- Log

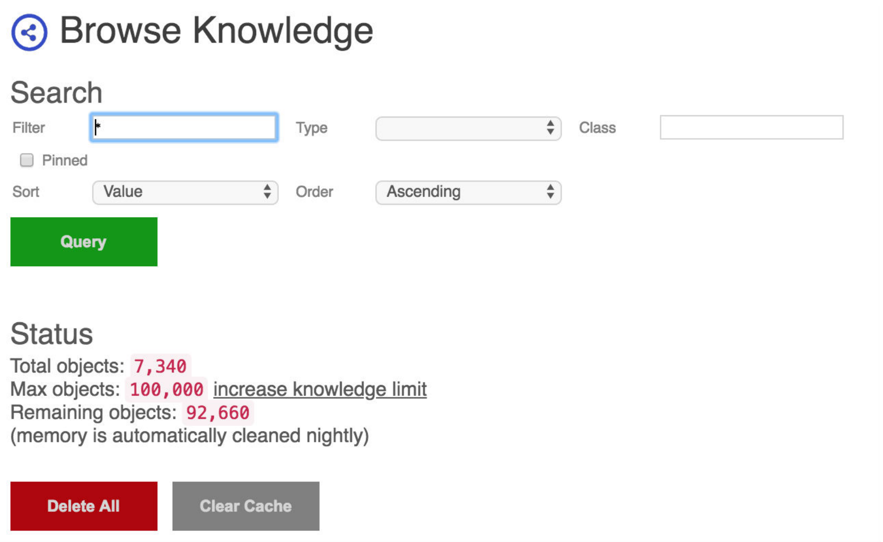
Knowledge

Overview

Each bot has its own integrated object database where it stores all of its data including conversations, users, responses, and scripts. You can browse your bot's knowledge base an import or export data using the JSON or CSV format.

Search

The knowledge tab allows you to query, view, and edit the knowledge in your bot's knowledgebase.



Browse Knowledge

Search

Filter: *

Type:

Class:

□ Pinned

Sort: Value

Order: Ascending

Query

Status

Total objects: `7,340`
Max objects: `100,000` increase knowledge limit
Remaining objects: `92,660`
(memory is automatically cleaned nightly)

Delete All

Clear Cache

Properties

Property	Description
Filter	Enter your knowledge data query string, use * as a wildcard.
Type	Filter the results by the type of data.
Class	Filter the results by the classification of data.
Pinned	Filter only pinned data.
Sort	Sort by Value, Name, Type, Creation date, Access date, and Access count.
Order	Order by Ascending or Descending.
Query	Queries and displays all bot's knowledgebase objects.
Delete All	Caution, this is permanently delete everything from the bot's memory and bootstrap it with minimal knowledge.
Clear Cache	This will clear your bots shared server-side cache. This can sometimes resolve issues your bot is having. This may affect any connected users.

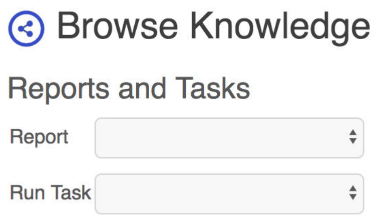
Toolbar

Icon	Property Name	Description
☰	Menu	Choose upload or import.
⌂	Home	Home of knowledge page.
📊	Reports and Tasks	View reports and tasks.
📋	Browse Knowledge	Process or query object in your bot's knowledgebase.
🔍	Inspect	Inspect selected objects.
⋖	Knowledge	Bot's knowledge.
☑	Select	Select all (or first 100) objects.
📌	Pin	Pin the selected objects, so they cannot be forgotten.
⊘	Unpin	Unpin the selected objects, so they can be forgotten.
	Delete	Permanently deletes selected objects or relationships.

Icon	Property Name	Description
✖		
⬇	Download	Import object to JSON or CSV spreadsheet.
⬆	Upload	Upload/Import objects from JSON or CSV (spreadsheet).
⇤	Import	Import objects from JSON or CSV script in the script library.

Reports and Tasks

The reports page lets you manage your bot's knowledgebase by running common cleanup reports and tasks.

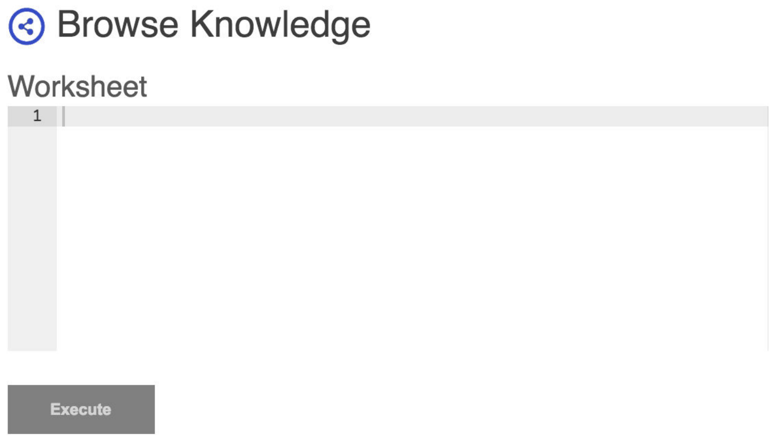


Properties

Property	Description
Report	Execute a predefined report.
Run Task	Execute a predefined administrative task.

Worksheet

The worksheet let you execute adhoc Self code to process or query object in your bot's knowledgebase.



Properties

Property	Description
Execute	Execute the self code.

Status

The status tab reports the bot's current database size. When the bot reaches its database limit, it will automatically garbage collect old objects and conversations.

See Also

- Browse knowledge
- Introducing the Self scripting language

Log

Overview

The log tab allows you monitor and debug your bot's processing.

See Also

- Log

Analytics

Overview

You can view your bot's analytics from its Admin Console by clicking on 'Analytics'. Analytics provides graphs and data on the bot's activity in the form of charts and tables. You can navigate through different analytics by selecting a specific social media chart, and further refine it by duration period and grouping.

Properties

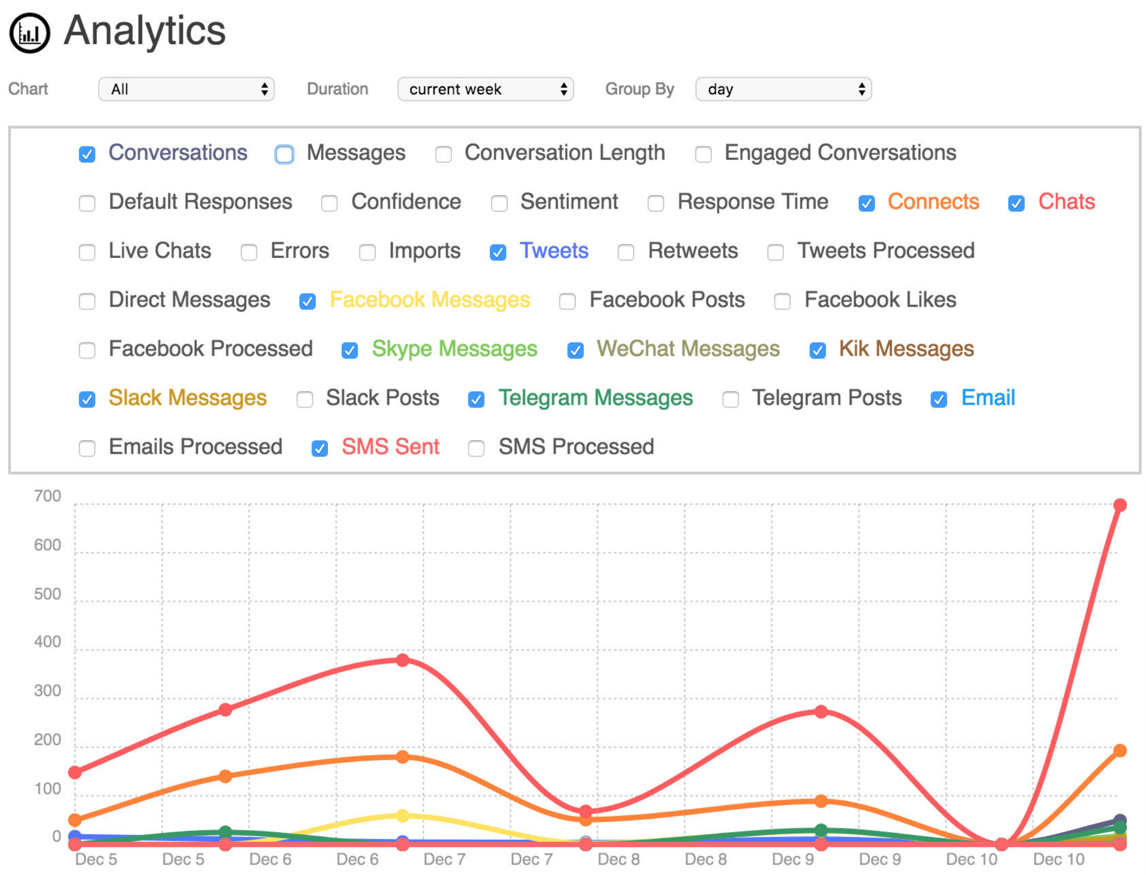
Properties	Description
Charts	Graphs of different social media interactions.
Duration	Displays a graph for a given period of time. User can choose duration of the graph to be displayed for either week, month, or everything.
Group By	Groups graph data either by day, week, or month.

Analytics

Properties	Description
Conversations	Total conversations the bot had for all chat and social media platforms.
Messages	Total messages the bot had for all chat and social media platforms.
Conversation Length	Total conversation length the bot had for all chat and social media platforms.
Engaged Conversations	Total number of conversations that bot had with at least three messages.
Default Responses	Total number of messages the bot did not find an answer for. When the bot does not know a response to a question, it uses its 'default' response.
Confidence	Avarage confidence the bot had on its responses based on the user question and the bot's matching trained question. Exact question matches and pattern matches are considered to be 100%, and default responses are 0%.
Sentiment	User's average sentiment or emotion to the bot's responses, either good or bad. If the user's questions seem very happy the sentiment will be 100%. If the user seems very unhappy the sentiment will be -100%. If the user did not express any emotion, then the sentiment will be 0%.
Connects	Total number of times a user or admin connected to the bot through the web or mobile.
Chats	Total number of chats that the bot had.
Live Chats	Total number of live chats that the bod had.
Errors	Total number of errors the bot had while responding to chats.
Response Time	How long it took the bot to respond the the user's message for chat and all social media platforms.
Imports	Total number of scripts or response lists imported into the bot.
Tweets	Total number of tweets the bot posted for Twitter social media.
Retweets	Total number of retweets the bot has made for Twitter social media.
Tweets Processed	Total number of tweets bot has read.
Direct Messages	Total number of private, user to user chat message for Twitter social media.
Facebook Posts	Total number of Facebook posts the bot has made for Facebook social media.
Facebook Likes	Total number of Facebook posts that bot 'likes'.
Facebook Messages	Total number of Facebook posts processed by the bot for Facebook social media.
Facebook Messages Processed	Total number of Facebook messages that the bot has read for Facebook social media.
Skype Messages	Total number of Skype messages the bot has read.
Kik Messages	Total number of Kik messages the bot has read.
WeChat Messages	Total number of WeChat messages the bot has read.
Slack Messages	Total number of Slack messages the bot has read.
Slack Posts	Total number of Slack posts the bot has made for Slack social media.
Telegram Messages	Total number of Telegram messages the bot has read.
Telegram Posts	Total number of Telegram posts the bot has made for Telegram social media.
Email	Total number of emails the bod has sent.
Emails Processed	Total number of emails the bot has read.
SMS Sent	Total number of SMS messages the bot has sent.
SMS Processed	Total number of SMS messages the bot has read.

Chart

Choose any of the 11 available bot's analytics charts you want to see by selecting it from 'Charts' drop down select. You can view bot's chart for a duration preriod of either current week, current month, or all time by selecting a duration period from the duration drop down select. You can group bot's analytics either by day, week, or month by selecting a group from a group by drop down select.



Analytics

Chart: All Duration: current week Group By: day

☑ Conversations □ Messages □ Conversation Length □ Engaged Conversations

□ Default Responses □ Confidence □ Sentiment □ Response Time ☑ Connects ☑ Chats

□ Live Chats □ Errors □ Imports ☑ Tweets □ Retweets □ Tweets Processed

□ Direct Messages ☑ Facebook Messages □ Facebook Posts □ Facebook Likes

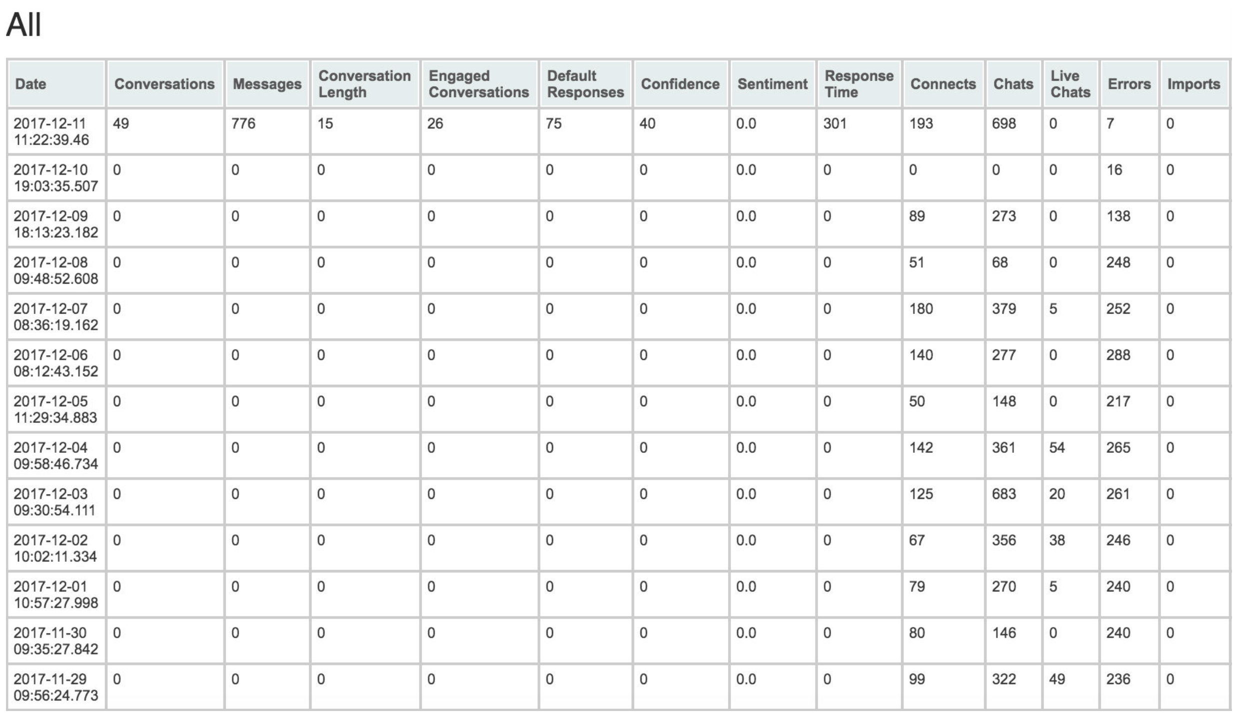
□ Facebook Processed ☑ Skype Messages ☑ WeChat Messages ☑ Kik Messages

☑ Slack Messages □ Slack Posts ☑ Telegram Messages □ Telegram Posts ☑ Email

□ Emails Processed ☑ SMS Sent □ SMS Processed

Table Stats

Table stats displays bot's raw analytic data in a table. The data in the table is graphed in a chart above. You can sort the bot's analytics in the table by any column by clicking on the column's title.



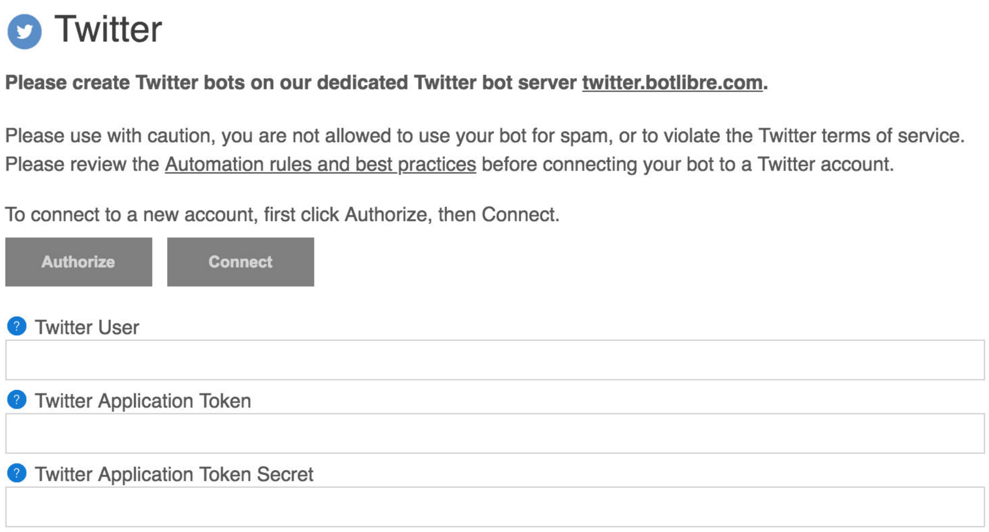
All

Date	Conversations	Messages	Conversation Length	Engaged Conversations	Default Responses	Confidence	Sentiment	Response Time	Connects	Chats	Live Chats	Errors	Imports
2017-12-11 11:22:39.46	49	776	15	26	75	40	0.0	301	193	698	0	7	0
2017-12-10 19:03:35.507	0	0	0	0	0	0	0.0	0	0	0	0	16	0
2017-12-09 18:13:23.182	0	0	0	0	0	0	0.0	0	89	273	0	138	0
2017-12-08 09:48:52.608	0	0	0	0	0	0	0.0	0	51	68	0	248	0
2017-12-07 08:36:19.162	0	0	0	0	0	0	0.0	0	180	379	5	252	0
2017-12-06 08:12:43.152	0	0	0	0	0	0	0.0	0	140	277	0	288	0
2017-12-05 11:29:34.883	0	0	0	0	0	0	0.0	0	50	148	0	217	0
2017-12-04 09:58:46.734	0	0	0	0	0	0	0.0	0	142	361	54	265	0
2017-12-03 09:30:54.111	0	0	0	0	0	0	0.0	0	125	683	20	261	0
2017-12-02 10:02:11.334	0	0	0	0	0	0	0.0	0	67	356	38	246	0
2017-12-01 10:57:27.998	0	0	0	0	0	0	0.0	0	79	270	5	240	0
2017-11-30 09:35:27.842	0	0	0	0	0	0	0.0	0	80	146	0	240	0
2017-11-29 09:56:24.773	0	0	0	0	0	0	0.0	0	99	322	49	236	0

Twitter

Overview

The twitter tab allows you to connect your bot to a Twitter account, and monitor a Twitter feed. The bot can manage the Twitter account, checking status updates, managing followers, replying to mentions and direct messages, retweeting important posts, tweeting your blog posts or RSS feed, and auto tweeting.



Twitter

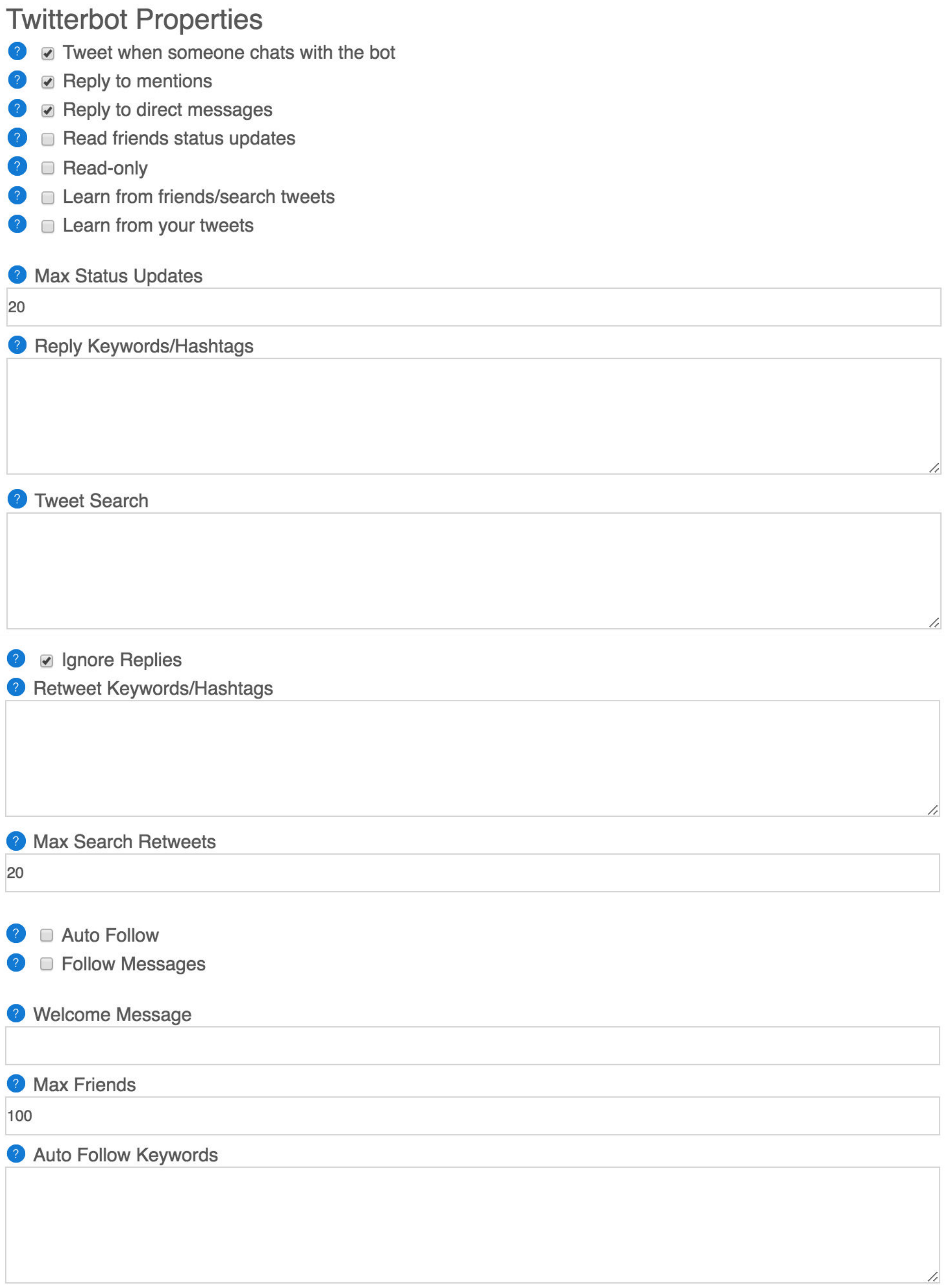
Please create Twitter bots on our dedicated Twitter bot server twitter.botlibre.com.

Please use with caution, you are not allowed to use your bot for spam, or to violate the Twitter terms of service.
Please review the Automation rules and best practices before connecting your bot to a Twitter account.

To connect to a new account, first click Authorize, then Connect.

Properties

Properties	Description
Authorize	Authorize your bot to access a twitter account.
Connect	Connect your bot to a twitter account.
Twitter User	You do not need to enter this, just click authorize.
Twitter Application Token	You do not need to enter this, just click authorize.
Twitter Application Token Secret	You do not need to enter this, just click authorize.



Twitterbot Properties

☑ Tweet when someone chats with the bot
☑ Reply to mentions
☑ Reply to direct messages
☐ Read friends status updates
☐ Read-only
☐ Learn from friends/search tweets
☐ Learn from your tweets

Max Status Updates
20

Reply Keywords/Hashtags

Tweet Search

☑ Ignore Replies

Retweet Keywords/Hashtags

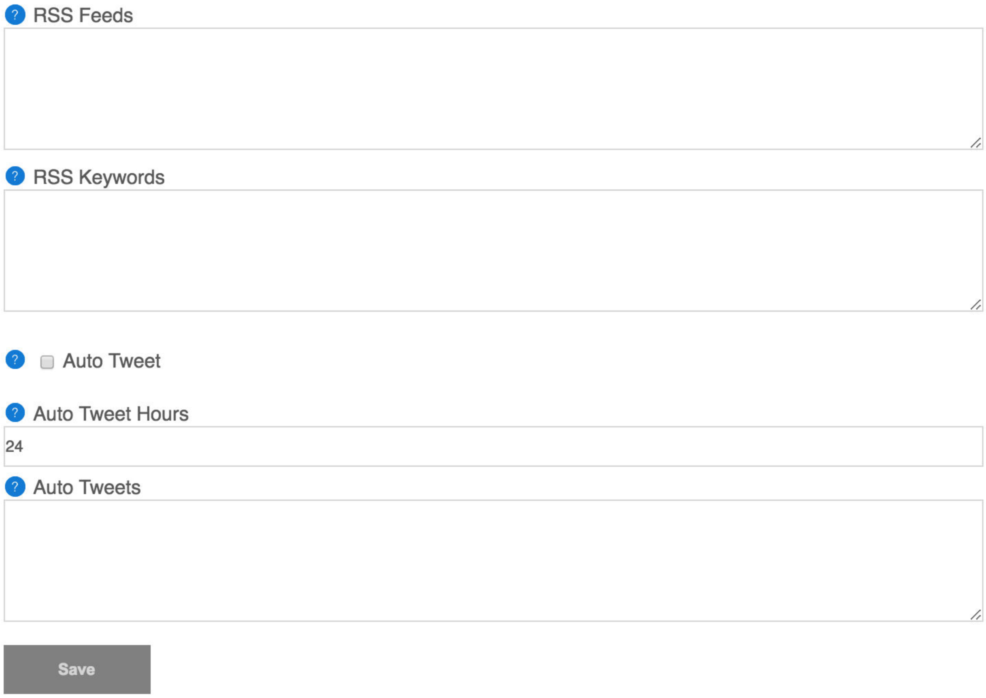
Max Search Retweets
20

☐ Auto Follow
☐ Follow Messages

Welcome Message

Max Friends
100

Auto Follow Keywords



RSS Feeds

RSS Keywords

□ Auto Tweet

Auto Tweet Hours

24

Auto Tweets

Save

Twitterbot Properties

Twitterbot Properties	Description
Tweet when someone chats with the bot	Configure if the bot should tweet that someone is chatting with it.
Reply to mentions	Configure if the bot should reply when someone mentions it in a tweet.
Reply to direct messages	Configure if the bot should reply when someone sends it a direct message.
Read friends status updates	Configure if the bot should read (and possibly respond to) its friend's status updates. The bot will only read the status updates that match one of the Status Keywords sets below.
Read-only	Configure if the bot should read (and not respond to) its friend's status updates.
Learn from friends/search tweets	Configure the bot to learn EVERY tweet as a valid response to itself or its hashtags. This is not recommended, and ensure your bot complies with Twitter's terms of use.
Learn from your tweets	Configure the bot to learn EVERY tweet from your account as a valid response to itself or its hashtags. This is not recommended, but can be used to train your bot.
Max Status Updates	Maximum status updates to read per cycle (max is 20). Retweets are not included. Larger limits are available for commercial accounts at, www.botlibre.biz
Reply Keywords/Hashtags	Only friend status updates or search results that contains 'all' of one of the keyword/tag sets will be read, and possibly responded to.
Tweet Search	Search for tweets matching one of the keyword/tag sets, process and possibly retweet or respond to them. Use this with caution, avoid spam, and ensure your bot complies with the Twitter terms of use.
Retweet Keywords/Hashtags	Retweet friend's, or search result tweets that contain one of the keyword/tag sets. Use this with caution, avoid spam, and ensure your bot complies with the Twitter terms of use.
Max Search Retweets	Maximum tweet search result to process or retweets per cycle (max is 20). Larger limits are available for commercial accounts at, www.botlibre.biz
Auto Follow	Configure if the bot may follow users who follow it. Use this with caution, and ensure your bot complies with the Twitter terms of use.
Follow Messages	Configure if the bot should follow, or unfollow a user when sent a 'follow me' or 'unfollow me' message.
Welcome Message	Private message to send users who follow the bot.
Max Friends	Maximum users to auto follow. Your bot can have more friends if you add them manually.
Auto Follow Keywords	Only user's who's description contains one of the keyword sets will be followed. Use this with caution, and ensure your bot complies with the Twitter terms of use.
RSS Feeds	Automatically tweet content from the RSS feeds. List each feed separated by a new line. You can include a prefix and/or a suffix to append to the RSS title.
RSS Keywords	Only tweet RSS feeds that contain one of the keywords set in their title.
Auto Tweet	Configure if the bot should tweet automatically every set number of hours.
Auto Tweet Hours	The number of hours to wait between auto tweets.
Auto Tweets	Set of tweets to auto tweet. List each tweet separated by a new line. Self and AIML templates can be used.

See Also

- Automate your Twitter presence with your own Twitterbot
- Video: How To Connect a Bot to Twitter

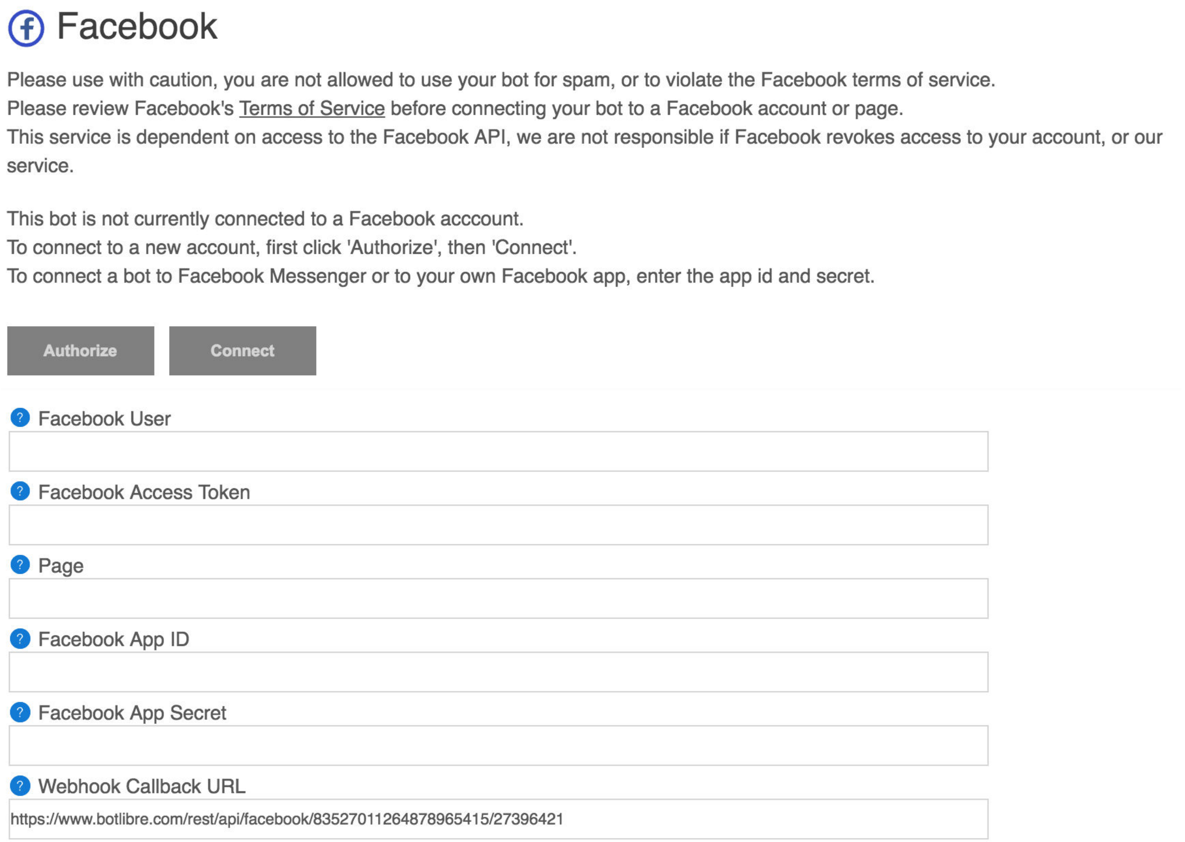
- Twitter: Automation rules and best practices
- How to make a Twitter bot reply to tweets
- twitter.botlibre.com

Facebook

Overview

The Facebook tab allows you to connect your bot to Facebook and Facebook Messenger, and monitor a Facebook account or page. Bot Libre provides its own Facebook app to provide many automation features, and lets you develop your own Facebook app to gain access to more features. To authorize a new account, just click the 'Authorize' button, then click 'Connect'. Some features are only supported for Facebook Page automation, or with your own Facebook app key.

To connect to a new account, first click 'Authorize', then 'Connect'.
To connect a bot to Facebook Messenger or to your own Facebook app click 'Advanced', and see How To Connect a Bot to Facebook Messenger.



Facebook

Please use with caution, you are not allowed to use your bot for spam, or to violate the Facebook terms of service.
Please review Facebook's Terms of Service before connecting your bot to a Facebook account or page.
This service is dependent on access to the Facebook API, we are not responsible if Facebook revokes access to your account, or our service.

This bot is not currently connected to a Facebook acccount.
To connect to a new account, first click 'Authorize', then 'Connect'.
To connect a bot to Facebook Messenger or to your own Facebook app, enter the app id and secret.

Facebook User

Facebook Access Token

Page

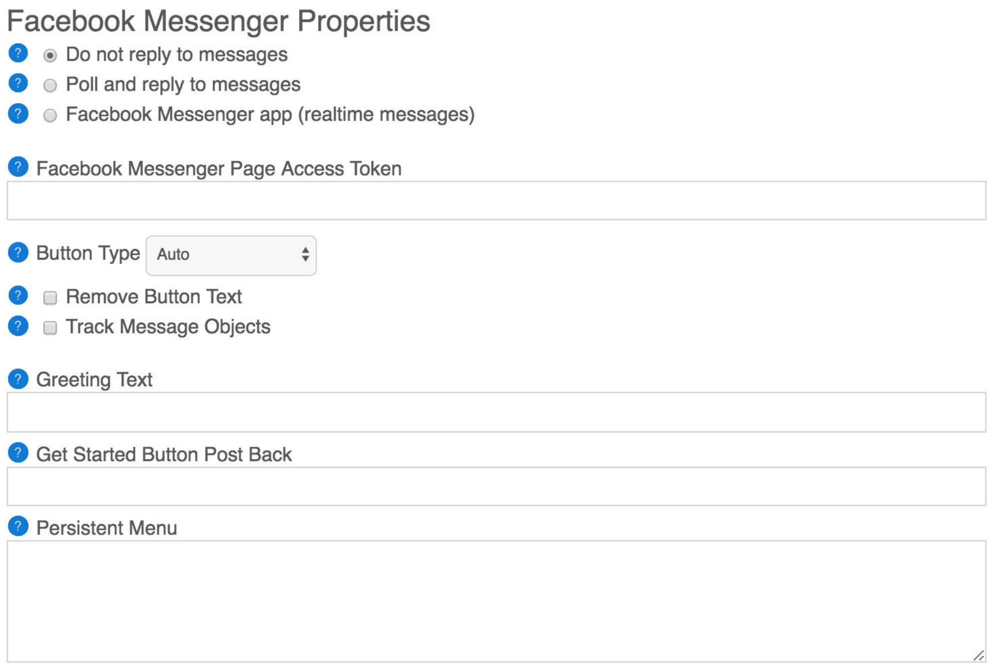
Facebook App ID

Facebook App Secret

Webhook Callback URL: https://www.botlibre.com/rest/api/facebook/83527011264878965415/27396421

Properties

Properties	Description
Authorize	Authorize your bot to access a facebook account.
Connecct	Connect your bot to a facebook account.
Facebook User	You do not need to enter this, just click authorize.
Facebook Access Token	You do not need to enter this, just click authorize.
Page	Select the page you wish the bot to monitor.
Facebook App ID	Advanced: Only for user developing their own facebook app.
Facebook App Secret	Advanced: Only for user developing their own facebook app.
Webhook Callback URL	A webhook is only required for realtime Facebook Messenger page messages. You must submit your webhook to your facebook app.



Facebook Messenger Properties

● Do not reply to messages

○ Poll and reply to messages

○ Facebook Messenger app (realtime messages)

Facebook Messenger Page Access Token

Button Type: Auto

□ Remove Button Text

□ Track Message Objects

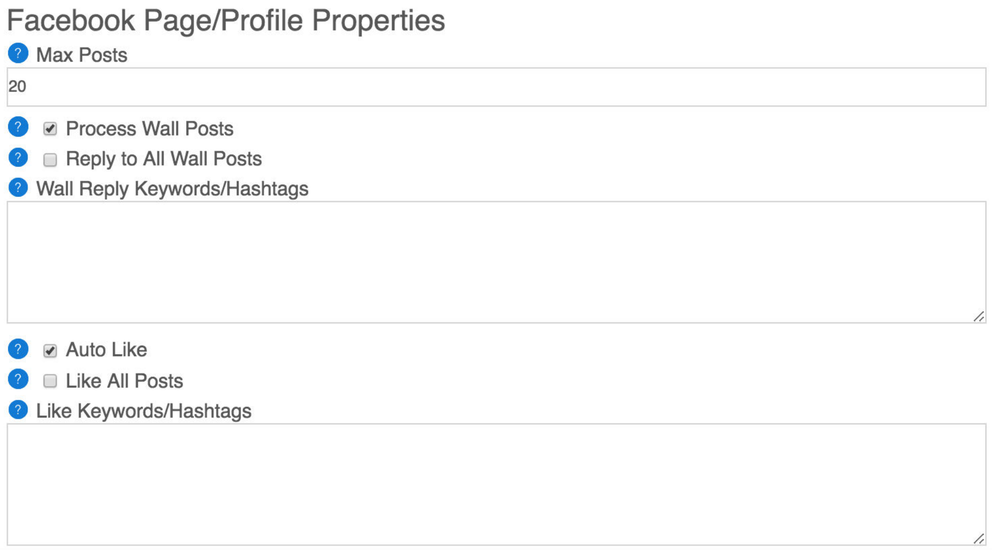
Greeting Text

Get Started Button Post Back

Persistent Menu

Facebook Messenger Properties

Facebook Messenger Properties	Description/th>
Do not reply to messages	Configure if the bot should not reply when someone sends it a message.
Poll and reply to messages	Configure if the bot should poll its messages and reply when someone sends it a message.
Facebook Messenger app (realtime messages)	Configure Facebook Messenger support for a Facebook page.
Facebook Messenger Page Access Token	Configure Facebook Messenger support for a Facebook page.
Button Type	Facebook Messenger supports two button types, 'buttons', and 'quick replies'.
Remove Button Text	Remove the button text from the message as it will be displayed in the Facebook Messenger button.
Greeting Text	Configure Facebook Messenger greeting text.
Get Started Button Post Back	Configure Facebook Messenger get started button post back.
Persistent Menu	Set of menu items for the Facebook Messenger 'Persistent Menu'.



Facebook Page/Profile Properties

Max Posts: 20

☑ Process Wall Posts

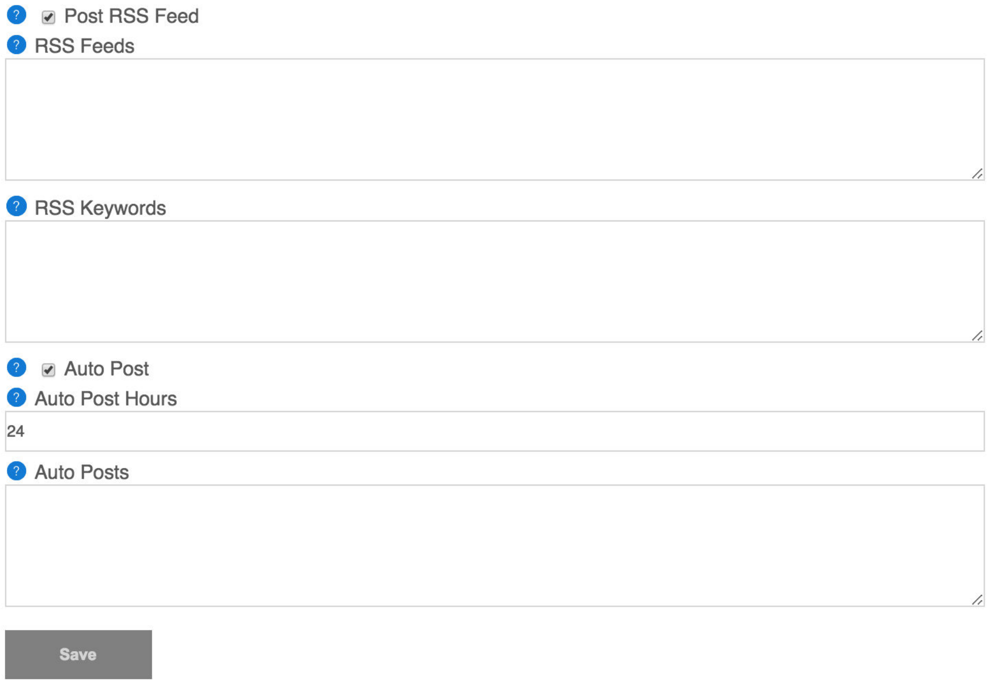
□ Reply to All Wall Posts

Wall Reply Keywords/Hashtags

☑ Auto Like

□ Like All Posts

Like Keywords/Hashtags



☑ Post RSS Feed

RSS Feeds

RSS Keywords

☑ Auto Post

Auto Post Hours

24

Auto Posts

Facebook Page/Profile Properties

Facebook Page/Profile Properties	Description
Max Posts	Maximum posts to process per cycle (max is 20). Larger limits are available for commercial accounts at, www.botlibre.biz
Process Wall Posts	Configure if the bot should read (and possibly respond to) its wall/page's posts. Facebook only allows access to wall posts to authorized Page accounts, or if you use your own Facebook app key.
Reply to All Wall Posts	Configure if the bot should reply to all posts to its wall/page.
Wall Reply Keywords/Hashtags	Only posts that contains 'all' of one of the keyword/tag sets will be replied to.
Auto Like	Automatically like posts shared on its wall.
Like All Posts	Configure if the bot should like all posts shared on its wall.
Like Keywords/Hashtags	Configure if the bot should like all posts shared on its wall.
Post RSS Feed	Automatically post content from an RSS feed.
RSS Feeds	Automatically post content from the RSS feeds. List each feed separated by a new line. You can include a prefix and/or a suffix to append to the RSS title.
RSS Keywords	Only post RSS feeds that contain one of the keywords set in their title. Keywords must be separated by a space (not a comma), each keyword set must be separated by a new line.
Auto Post	Configure if the bot should post automatically every set number of hours.
Auto Post Hours	The number of hours to wait between auto posts.
Auto Posts	Set of posts to auto post. List each post separated by a new line. Self and AIML templates can be used.

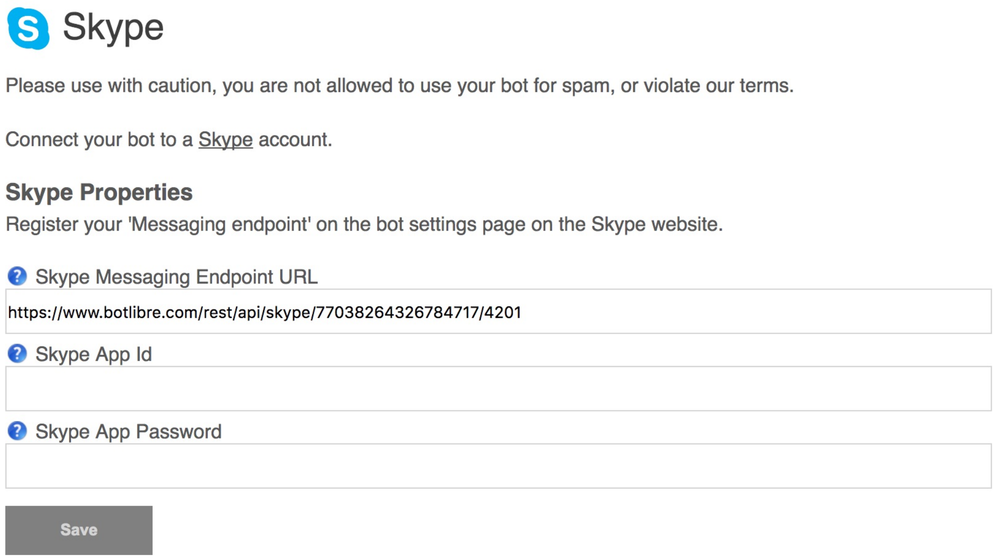
See Also

- Automating your Facebook presence using a Facebook bot
- Video: How To Connect a Bot to Facebook Messenger
- Video: How To Connect a Bot to a Facebook Page
- How To Connect a Bot to Facebook Messenger
- Facebook: Terms of Service

Skype

Overview

You can connect your bot to Skype to answer questions or monitor a conversation.



Skype

Please use with caution, you are not allowed to use your bot for spam, or violate our terms.

Connect your bot to a Skype account.

Skype Properties

Register your 'Messaging endpoint' on the bot settings page on the Skype website.

Skype Messaging Endpoint URL

https://www.botlibre.com/rest/api/skype/770382643267847174201

Skype App Id

Skype App Password

Properties

Properties	Description
Skype Messaging Endpoint URL	Set this URL on the bot settings page on the Microsoft Bot Framework website to enable replying to messages on Skype.
Skype App Id	Enter the App Id from the bot settings page on the Microsoft Bot Framework website.
Skype App Password	Enter the App Password from the bot settings page on the Microsoft Bot Framework website.

See Also

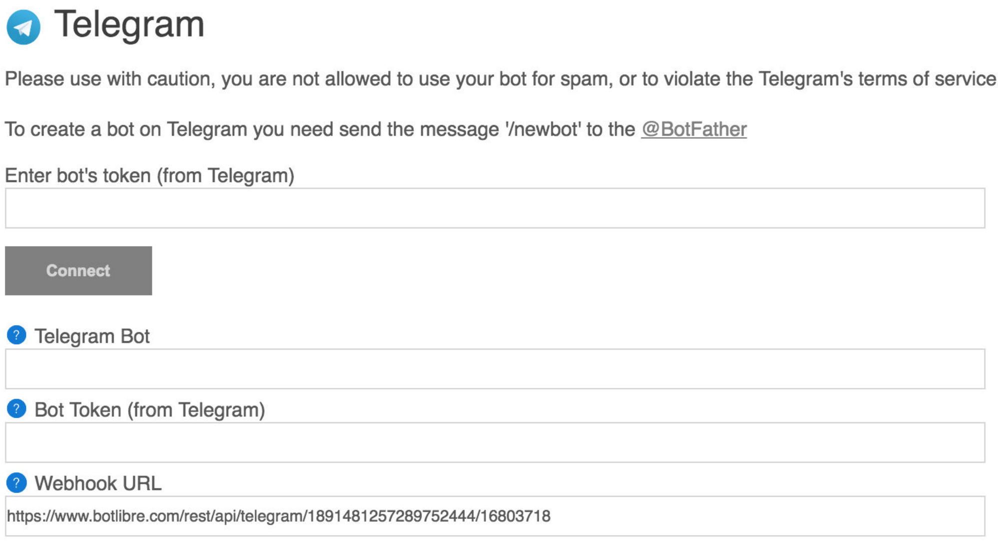
- How To Connect a Bot to Skype and the Microsoft Bot Framework
- Video: How To Connect a Bot to Skype and the Microsoft Bot Framework
- Skype
- Microsoft Bot Framework

Telegram

Overview

You can connect your bot to a Telegram chat account and chat with your bot on mobile, or have your bot manage your Telegram channel.

The Telegram tab allows you to connect your bot to Telegram.



Telegram

Please use with caution, you are not allowed to use your bot for spam, or to violate the Telegram's terms of service.

To create a bot on Telegram you need send the message '/newbot' to the @BotFather

Enter bot's token (from Telegram)

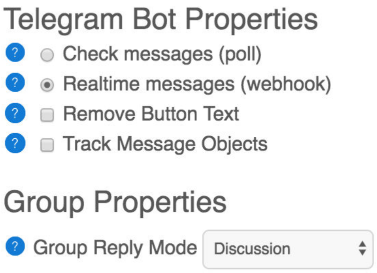
Telegram Bot

Bot Token (from Telegram)

Webhook URL: https://www.botlibre.com/rest/api/telegram/1891481257289752444/16803718

Properties

Properties	Description
Connect	Connect the bot.
Check Messages	Have the bot check its telegram messages.
Telegram Bot	You do not need to enter this, just click Connect.
Bot Token (from Telegram)	You do not need to enter this, just click Connect.
Webhook URL	Enable realtime messages and then click Connect to automatically send your webhook to Telegram.



Telegram Bot Properties

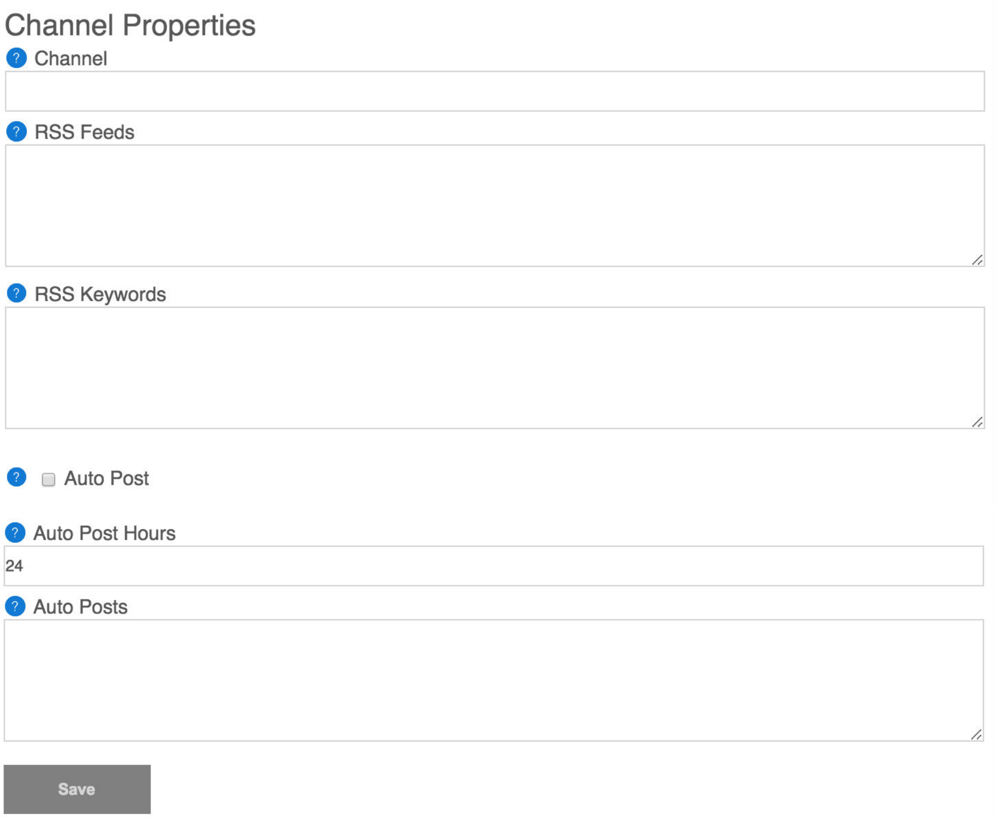
○ Check messages (poll)
● Realtime messages (webhook)
□ Remove Button Text
□ Track Message Objects

Group Properties

Group Reply Mode: Discussion

Telegram Bot Properties

Telegram Bot Properties	Description
Check messages (poll)	Configure if the bot should poll for new messages and reply.
Realtime messages (webhook)	Configure if the bot should use a webhook to reply to messages in realtime.
Remove Button Text	Configure if the bot should remove the button text from the message.
Track Messages	Stores message in bot's input and conversation logs.
Group Reply Mode	Choose bot's reply mode (Ignore, Listening Only, Listening, Discussion, and Conversation) to the group.



Channel Properties

Channel

RSS Feeds

RSS Keywords

□ Auto Post

Auto Post Hours

24

Auto Posts

Channel Properties

Channel Properties	Description
Channel	You can have your bot manage a channel, and post from an RSS feed, or auto post. Enter your channel's name, and ensure your bot has been added as a channel administrator.
RSS Feeds	Automatically post content from the RSS feeds to a channel. List each feed separated by a new line. You can include a prefix and/or a suffix to append to the RSS title.
RSS Keywords	Only post RSS feeds that contain one of the keywords set in their title. Keywords must be separated by a space (not a comma), each keyword set must be separated by a new line.
Auto Post	Configure if the bot should post automatically every set number of hours.
Auto Post Hours	The number of hours to wait between auto posts.
Auto Posts	Set of posts to auto post. List each post separated by a new line. Self and AIML templates can be used.

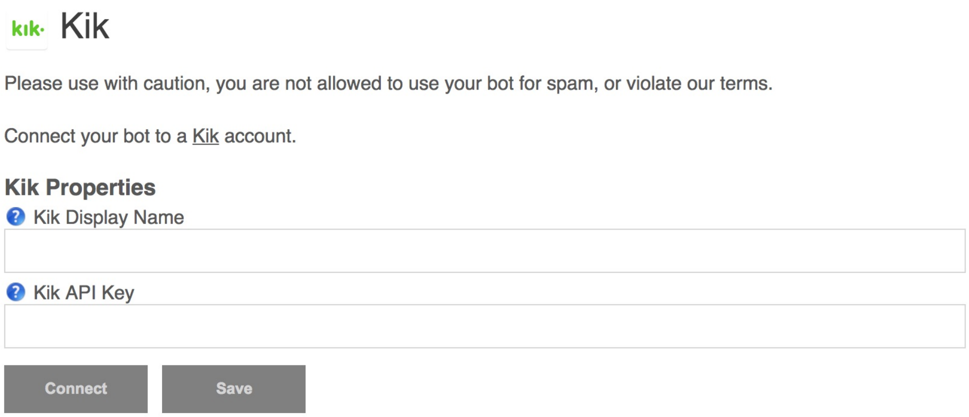
See Also

- Automating your Mobile Presence with a Telegram Bot
- Brain Bot
- Julie
- Bot Libre's Help Bot
- Bot Libre's Channel

Kik

Overview

You can connect your bot to Kik to answer questions or monitor a conversation.



Kik

Please use with caution, you are not allowed to use your bot for spam, or violate our terms.

Connect your bot to a Kik account.

Kik Properties

Kik Display Name

Kik API Key

Properties

Properties	Description
Kik Display Name	Enter the Kik Display Name from the bot configuration page on the Kik Dev website.
Kik API Key	Enter the Kik API Key from the bot configuration page on the Kik Dev website.

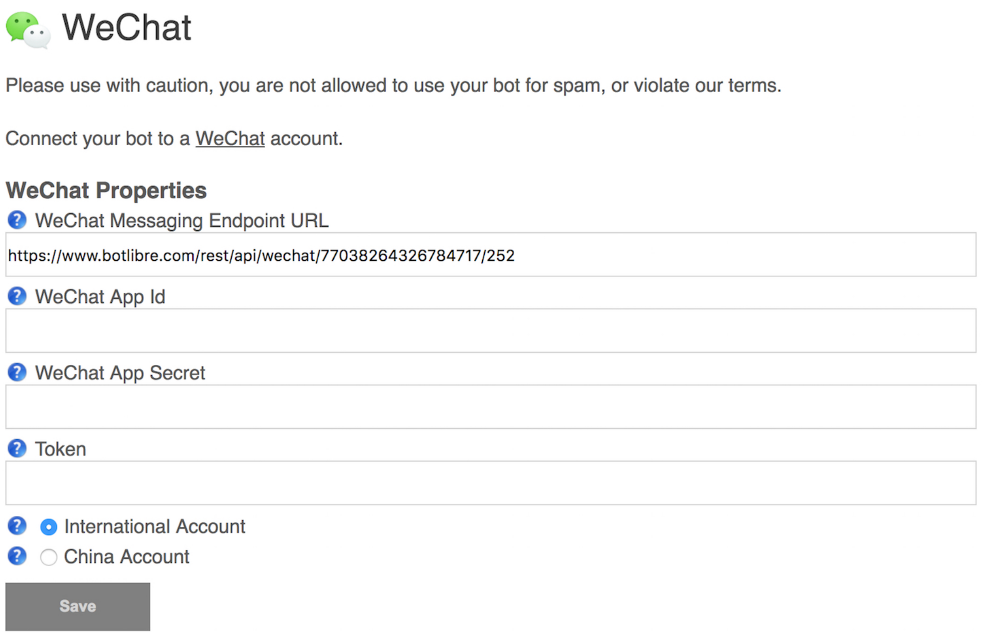
See Also

- How To Connect a Bot to Kik
- Kik

WeChat

Overview

You can connect your bot to WeChat to answer questions or monitor a conversation.



WeChat

Please use with caution, you are not allowed to use your bot for spam, or violate our terms.

Connect your bot to a WeChat account.

WeChat Properties

WeChat Messaging Endpoint URL

https://www.botlibre.com/rest/api/wechat/77038264326784717/252

WeChat App Id

WeChat App Secret

Token

● International Account

○ China Account

Properties

Properties	Description
WeChat Messaging Endpoint URL	Enter this URL on the WeChat Developer Center to enable replying to messages on WeChat.
WeChat App Id	Enter the app id from the WeChat Developer Center.
WeChat App Secret	Enter the app secret from the WeChat Developer Center.
Token	Enter the token from the WeChat Developer Center.
International Account	Check this box if using International Account.
China Account	Check this box if using China Account.

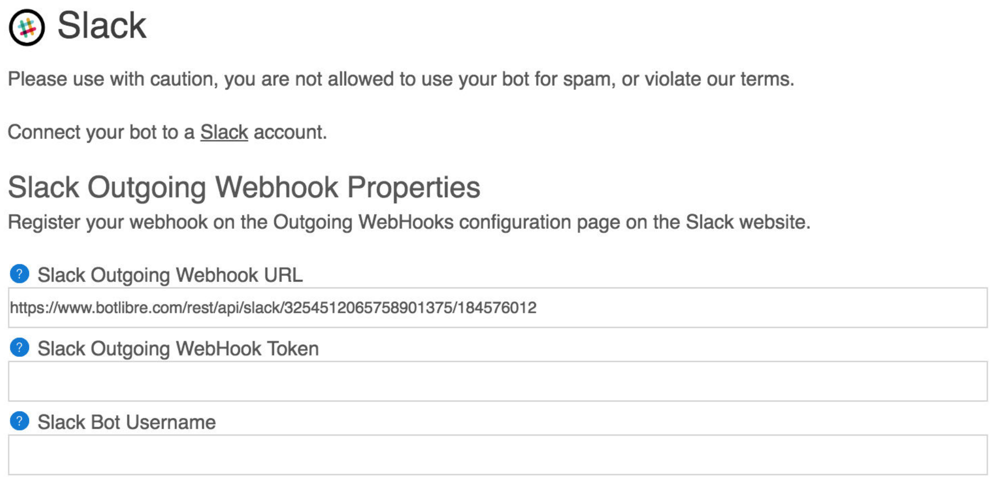
See Also

- How To Connect a Bot to WeChat
- WeChat
- WeChat Admin
- WeiXin

Slack

Overview

You can connect your bot to a Slack group to answer questions, post a feed, or monitor the conversation.



Slack

Please use with caution, you are not allowed to use your bot for spam, or violate our terms.

Connect your bot to a Slack account.

Slack Outgoing Webhook Properties

Register your webhook on the Outgoing WebHooks configuration page on the Slack website.

Slack Outgoing Webhook URL

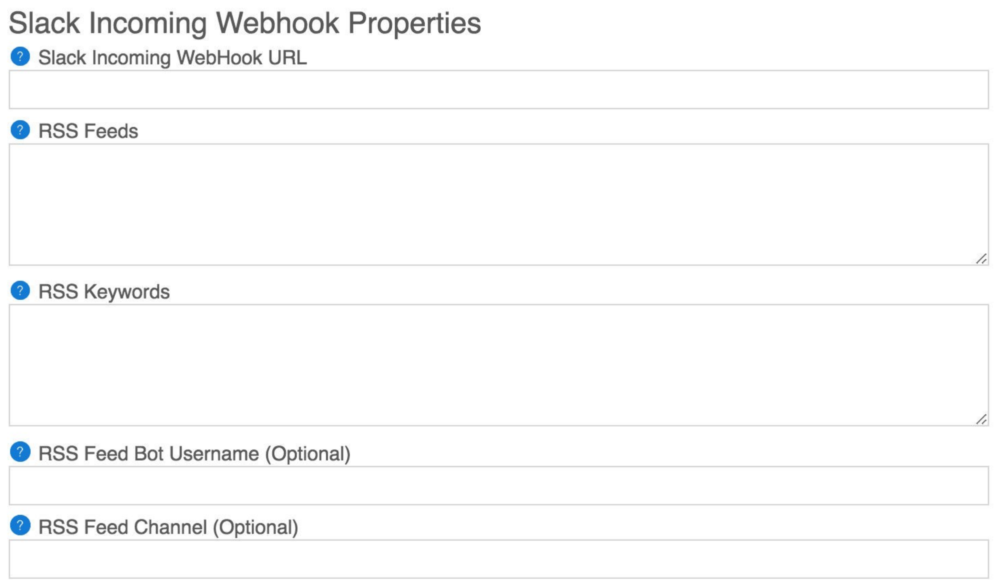
https://www.botlibre.com/rest/api/slack/3254512065758901375/184576012

Slack Outgoing WebHook Token

Slack Bot Username

Properties

Properties	Description
Slack Outgoing WebHook URL	Set this URL on the Outgoing WebHooks configuration page on the Slack website to enable replying to messages on Slack.
Slack Outgoing WebHook Token	Enter the Token from the Outgoing WebHook configuration page on the Slack website.
Slack Bot Username	Enter the username from the Outgoing WebHook configuration page on the Slack website.



Slack Incoming Webhook Properties

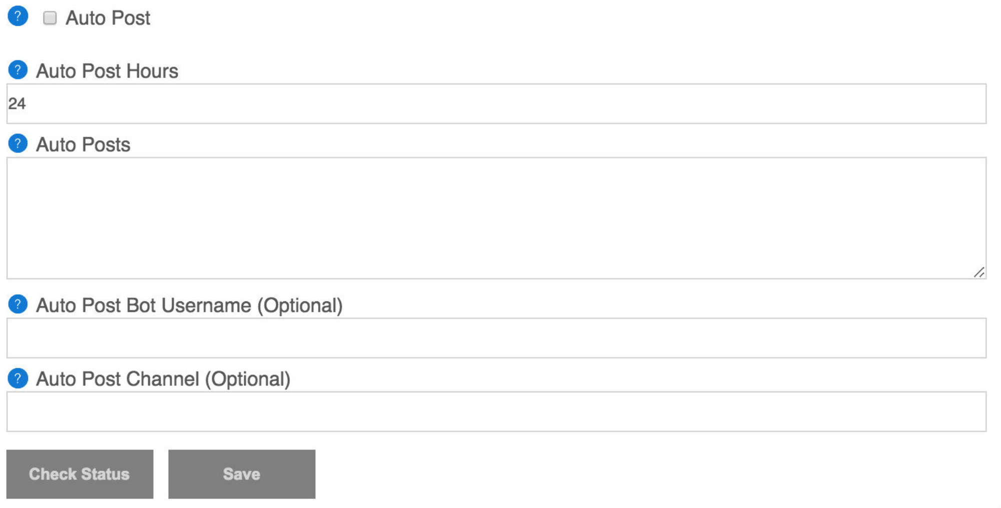
Slack Incoming WebHook URL

RSS Feeds

RSS Keywords

RSS Feed Bot Username (Optional)

RSS Feed Channel (Optional)



□ Auto Post

Auto Post Hours

24

Auto Posts

Auto Post Bot Username (Optional)

Auto Post Channel (Optional)

Slack Incoming Webhook Properties

Slack Incoming Webhook Properties	Description
Slack Incoming WebHook URL	Enter the WebHook URL from the Incoming WebHook configuration page on the Slack website.
RSS Feeds	Automatically post content from the RSS feeds to a channel. List each feed separated by a new line. You can include a prefix and/or a suffix to append to the RSS title.
RSS Keywords	Only post RSS feeds that contain one of the keywords set in their title. Keywords must be separated by a space (not a comma), each keyword set must be separated by a new line.
RSS Feed Bot Username (Optional)	Override the bot username to use when posting an rss feed.
RSS Feed Channel (Optional)	Override the channel to post the rss feed to.
Auto Post	Configure if the bot should post automatically every set number of hours.
Auto Post Hours	The number of hours to wait between auto posts.
Auto Posts	Set of posts to auto post. List each post separated by a new line. Self and AIML templates can be used.
Auto Post Bot Username (Optional)	Override the bot username to use when posting an auto post.
Auto Post Channel (Optional)	Override the channel to post the auto post to.

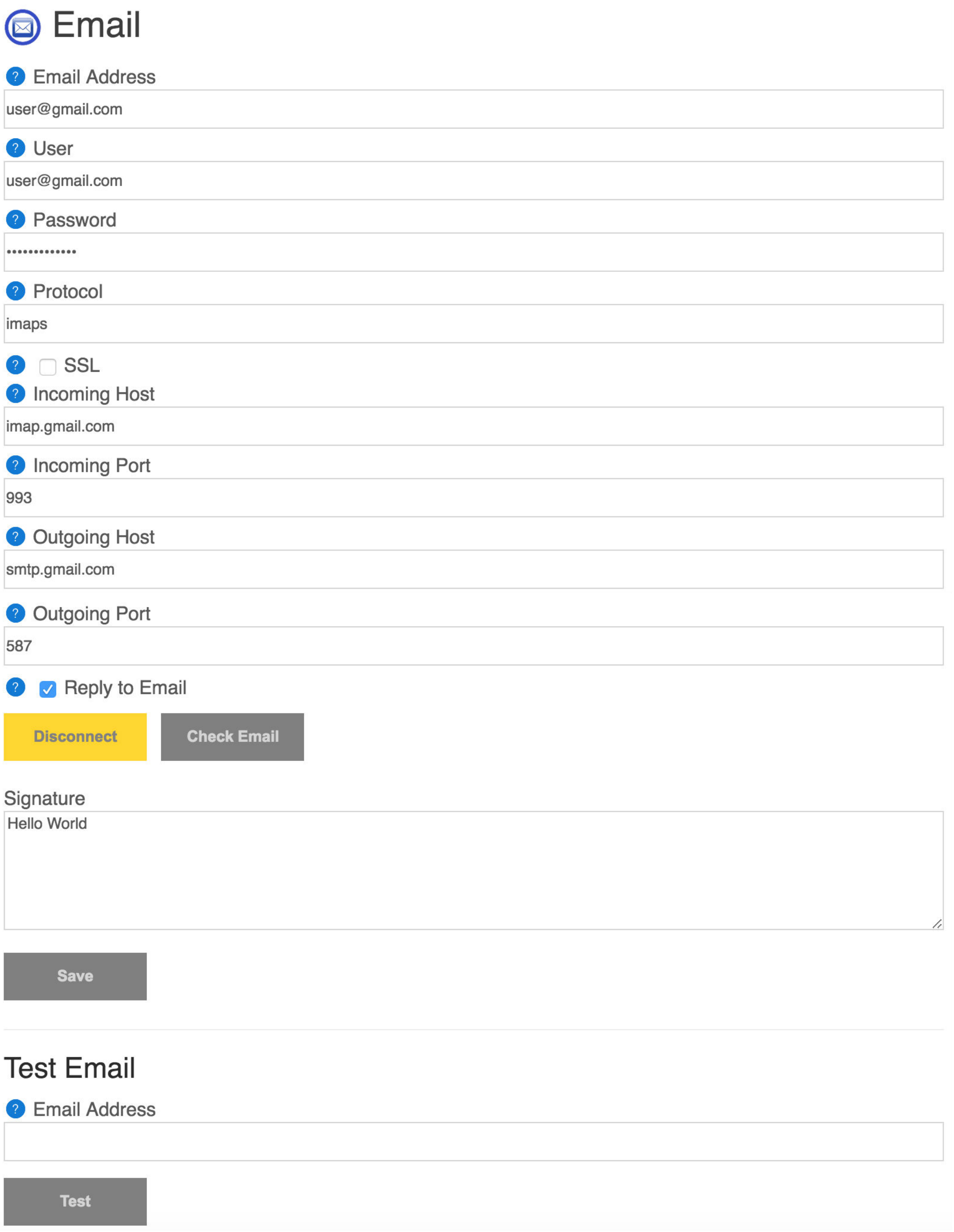
See Also

- How To Connect a Bot to Slack
- Slack

Email

Overview

The email tab allows you to connect your bot to an email account, and monitor and reply to messages.



Email

Email Address: user@gmail.com

User: user@gmail.com

Password: •••••••••••••

Protocol: imaps

□ SSL

Incoming Host: imap.gmail.com

Incoming Port: 993

Outgoing Host: smtp.gmail.com

Outgoing Port: 587

☑ Reply to Email

Signature: Hello World

Test Email

Email Address:

Properties

Properties	Description
Email Address	For the user email address.
User	For the user email address.
Password	Unsecure password.
Protocol	Protocol name.

Properties	Description
SSL	To use search sockets.
Incoming Host	Incoming host email.
Incoming Port	Incoming port number.
Outgoing Host	Outgoing host email.
Outgoing Port	Outgoing port number.
Reply to Email	Bot will reply to all the incomming emails.
Disconnect	Disconnect bot from current email address
Check Email	Bot checks email for the newly received emails.
Signature	Signature of email that the bot will use when replying to emails.
Save	Save current email settngs.
Test	Sends a test email to the above email address.

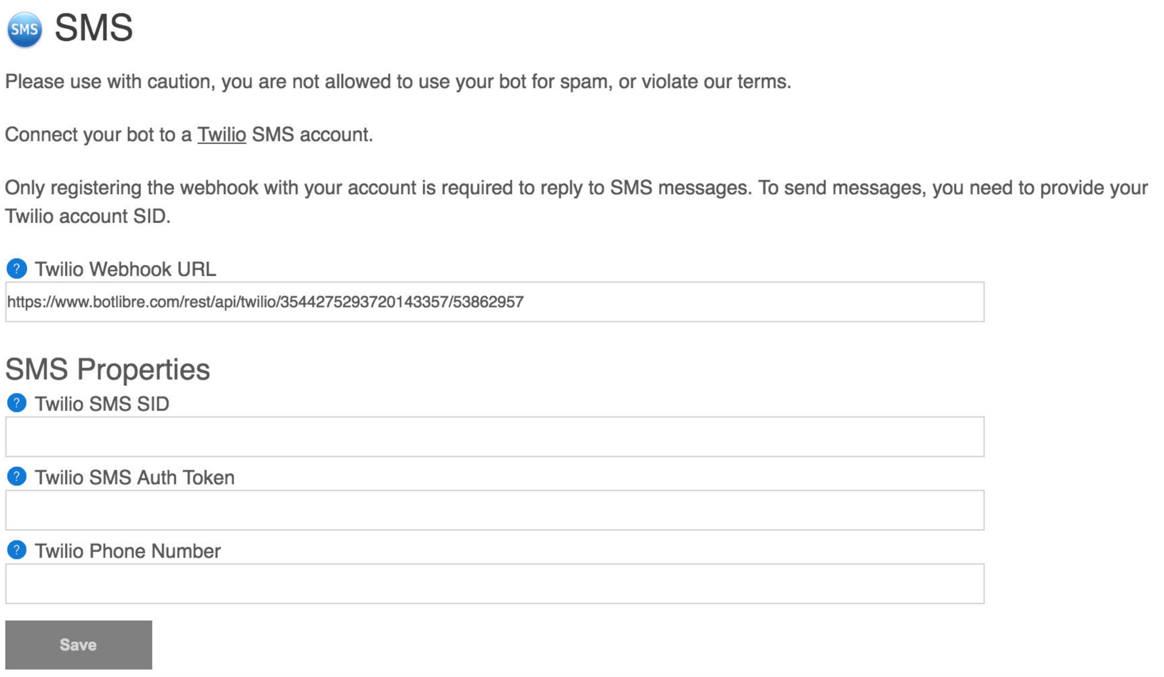
See Also

- email.botlibre.com

SMS

Overview

The SMS tab allows you to connect your bot to SMS messaging.



SMS

Please use with caution, you are not allowed to use your bot for spam, or violate our terms.

Connect your bot to a Twilio SMS account.

Only registering the webhook with your account is required to reply to SMS messages. To send messages, you need to provide your Twilio account SID.

Twilio Webhook URL: https://www.botlibre.com/rest/api/twilio/3544275293720143357/53862957

SMS Properties

Twilio SMS SID:

Twilio SMS Auth Token:

Twilio Phone Number:

SMS Properties

Properties	Description
Twilio Webhook URL	Set this URL in your twillo account to enable replying to SMS messages.
Twilio SMS SID	To send SMS messages, enter your Twilio SMS account SID.
Twilio SMS Secret	To send SMS messages, enter your Twilio SMS account secret.
Twilio Phone Number	To send SMS messages, enter your Twilio SMS account phone number. Use the full number, i.e. +16131234567

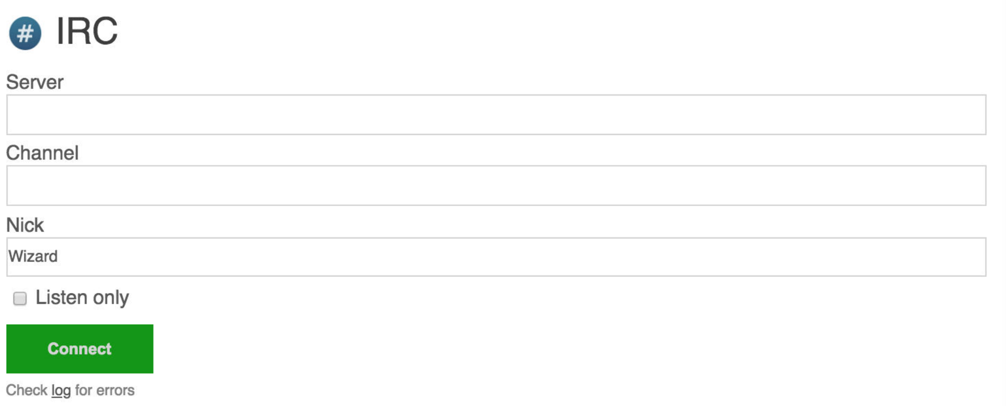
See Also

- How To Connect your Bot to SMS Text Messaging
- Video: How to create a chat bot for Twilio and SMS
- twillo.com

IRC

Overview

IRC is Internet Relay Chat, a chat room standard. You can create an IRC bot. The IRC tab allows you to connect your bot to an IRC chat room, so it can chat or listen and learn from others.



IRC

Server

Channel

Nick: Wizard

□ Listen only

Check log for errors

Properties

Properties	Description
Server	
Channel	
Nick	
Listen only	

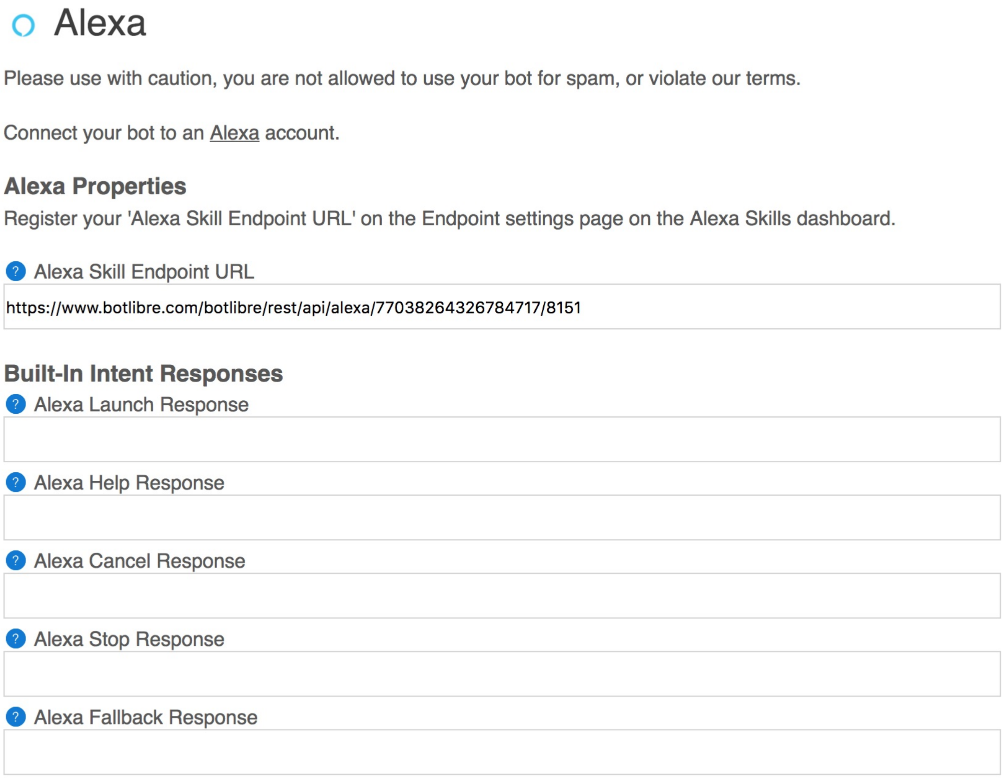
See Also

- IRC

Alexa

Overview

You can connect your bot to Alexa to answer questions or hold a conversation on an Amazon Echo device.



Alexa

Please use with caution, you are not allowed to use your bot for spam, or violate our terms.

Connect your bot to an Alexa account.

Alexa Properties

Register your 'Alexa Skill Endpoint URL' on the Endpoint settings page on the Alexa Skills dashboard.

Alexa Skill Endpoint URL

https://www.botlibre.com/botlibre/rest/api/alexa/77038264326784717/8151

Built-In Intent Responses

Alexa Launch Response

Alexa Help Response

Alexa Cancel Response

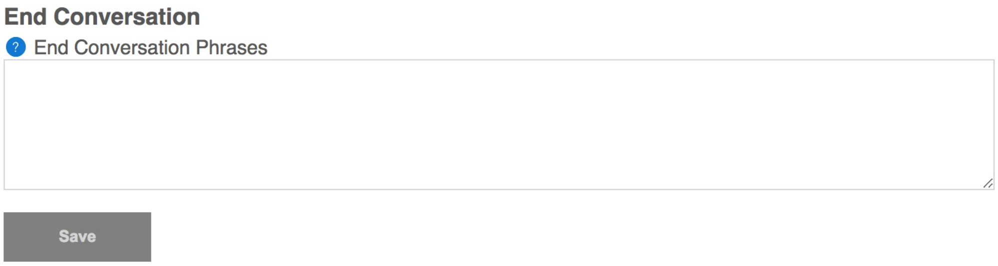
Alexa Stop Response

Alexa Fallback Response

Built-In Intent Responses will be required to pass Amazon's Alexa Skill certification. These will only be active before the conversation with your bot begins. After the conversation has begun, your bot will handle all responses.

Properties

Properties	Description
Alexa Skill Endpoint URL	Enter this URL into the Endpoint settings page of the Alexa Skills dashboard.
Alexa Launch Response	Enter the response Alexa will say after the user launches your Skill. This should prompt the user to begin the conversation.
Alexa Help Response	Enter the response Alexa will say after the user asks for help.
Alexa Cancel Response	Enter the response Alexa will say after the user cancels interacting with your Skill.
Alexa Stop Response	Enter the response Alexa will say after the user stops interacting with your Skill.
Alexa Fallback Response	Enter the response Alexa will say if it does not understand the user's command.



End Conversation

End Conversation Phrases

End Conversation Properties

Properties	Description
End Conversation Phrases	Enter a set of phrases the user can say to end the chat session, separated by a new line.

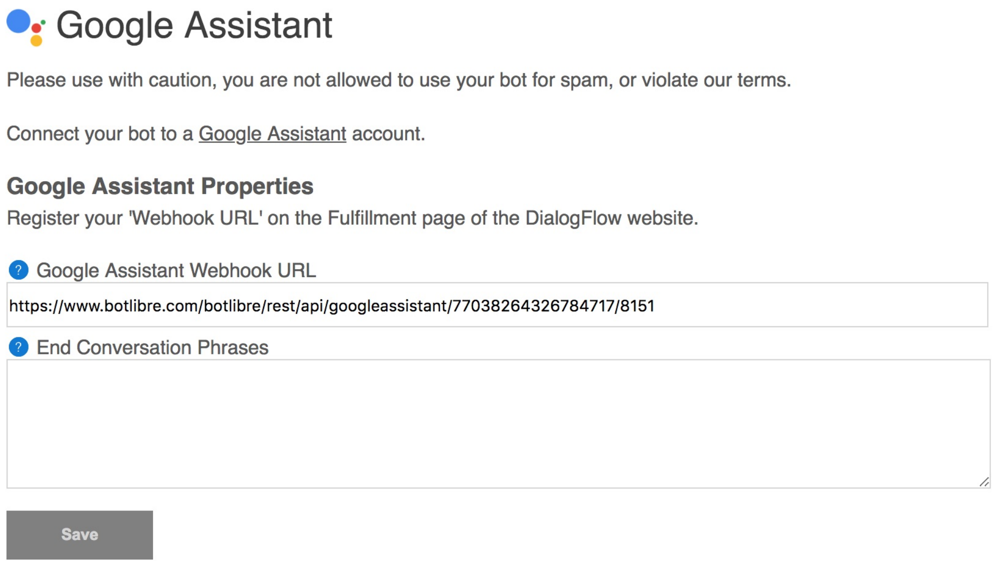
See Also

- How to create a bot for Amazon Alexa
- Alexa

Google Assistant

Overview

You can connect your bot to Google Assistant to answer questions or hold a conversation on a Google Home or other Google Assistant enabled device.



Google Assistant

Please use with caution, you are not allowed to use your bot for spam, or violate our terms.

Connect your bot to a Google Assistant account.

Google Assistant Properties

Register your 'Webhook URL' on the Fulfillment page of the DialogFlow website.

Google Assistant Webhook URL

https://www.botlibre.com/botlibre/rest/api/googleassistant/77038264326784717/8151

End Conversation Phrases

Properties

Properties	Description
Google Assistant Webhook URL	Set this URL on the Fulfillment page on the DialogFlow website to enable replying to messages.
End Conversation Phrases	Enter a set of phrases the user can say to end the chat session, separated by a new line.

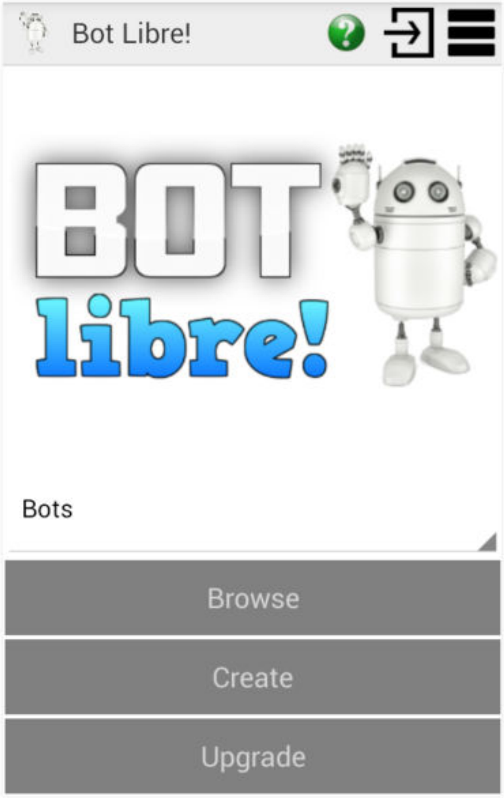
See Also

- How to create a bot for Google Home and Google Assistant
- Actions on Google
- Google Assistant

Android

Overview

You can download the Bot Libre app for Android, or use the Bot Libre Android SDK or web API to make your own Android app.



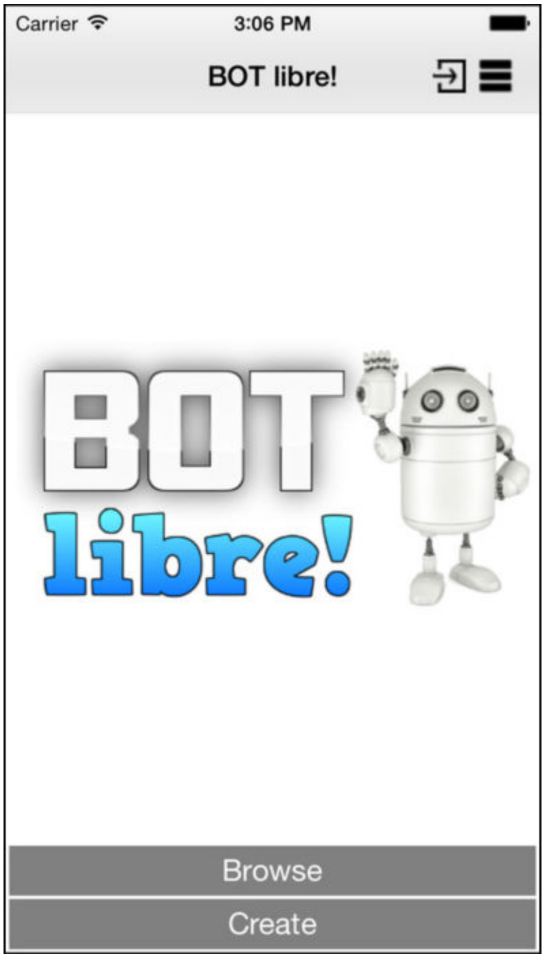
See Also

- SDK
- Bot Libre on Google Play

iOS

Overview

Bot Libre has an open source iPhone and iOS SDK that lets you build your own app or integrate our services into your own app.



See Also

- SDK
- Bot Libre on iTunes

SDK

Overview

The Bot Libre SDK provides an API for Android and iOS. The mobile SDK also provides a set of reusable interface components, and examples apps that make it easy for you to develop your own app.

See Also

- SDK
- Create your own bot app with the Bot Libre SDK

Response List

Overview

A Response List is Bot Libre's text file format for defining a bot's responses. A response list is a list of question/response pairs. Each phrase is separated by a new line, and each question/response list is separated by an empty line. You can also tag responses with meta data such as keywords to influence when the response is used. Response lists are the recommended way to train a bot. The bot will automatically find the best matching response for any question, the questions do not need to be exact matches, only sufficiently similar, or include a keyword.

Response Tags

Tag	Description
question:	Defines the question. This is not required, as the first line after a blank line is assumed to be a new question.
pattern:	Defines a question as a pattern. Patterns allow the wildcard * and other wild cards to be used.
response:	Defines a response. This is not required, as any lines after the question as assumed to be responses.
template:	Defines a response template. This lets you embed Self code inside {} brackets for dynamic responses.
sentiment:	You can associate sentiment (good/bad) with the phrase.
condition:	Optionally you can give a condition in self code that must evaluate to true for the response to be use.
think:	Optionally you can give a 'think' code in self that is evaluated when the response is used.
command:	Optionally you can give a JSON command (Self code) that is evaluated and returned to the client to support games and virtual assistance.
topic:	Optionally you can give a topic to categorize the response under.
require topic:	If the topic is required, the response will only be used when the topic is active.
exclusive topic:	If the topic is exclusive, only response that share the topic will be used while the topic is active.
label:	Optionally you can give a label to reuse the response as.
keywords:	Optionally you can give keywords from the question that will influence a response match.
required:	Optionally you can give required words from the question that will be required for a response match.
emotions:	You can associate an emotion with the response.
actions:	You can associate an action with the response.
poses:	You can associate a pose with the response.
on repeat:	Optionally you can give a response to used if the current response has already been used in this conversation.
no repeat:	Require that the response or phrase only be used once.
previous:	Optionally you can give a previous response to give a response a context.
require previous:	Require that the response only be used if the previous response matches one of the previous responses.
confidence:	Confidence can be used if a response is not certain. The matching response with the highest confidence will be used.

question:

The response question can be a phrase that will be automatically matched with similar questions, or can be a Pattern. Patterns are matched literally in the order of the words in the pattern, and can have wildcard characters such as * and can also include Regular Expressions. Normally it is best to use a phrase, and let the bot decide if a question should match, you can also add keywords and required words to improve when a question is matched. To enter a pattern use the pattern: tag.

```
what is your name

question: what is your name

pattern: What is *
```

response:

The response can be a phrase, can contain HTML or rich content including images, video, and buttons, or a response can be a Template. A template is a response that can include some Self code. Any code inside {} in a template will be executed, and the result printed into the response. To enter a template use the template: tag and enter Self code inside {} brackets.

```
My name is Julie.

My name is <b≷Julie</b≷.

response: My name is Julie.
```

```
template: My name is {#self.name}.

template: The date is {Date.date()}.
```

keywords:

Keywords are important words in the question that should influence when the response is used.

```
// Keywords are used to allow this response to be used for any questions on hockey or spinach.
question: do you like hockey
response: Hockey is my favorite sport.
keywords: hockey

question: what do you love
response: I love hockey!
keywords: love luv like adore

question: do you like field hockey
response: I only play ice hockey.
keywords: "field hockey"

question: i love spinach
response: Spinach is is my favorite food.
keywords: spinach
```

A keyword will be used by the bot to select the best response. Keywords let some words count for more than other words. In the example, the bot knows two responses, how will it respond to the phrase "I love hockey"? Without the keywords the bot would probably match this question with "I love spinach", but since hockey is a keyword, it will instead match it with "Do you like hockey?".

If the response has no defined required words, then the keywords will also be required for the response match. So "I love hockey" would not be matched with "I love spinach." because it is missing the keyword.

If you keywords has many similar words. You can list them all as keywords to match any of the words. You could also define these words a synonyms, which may be a better option.

You can also use compound words are keywords. A compound keyword must have all the words in the order to be used. For a compound keyword just wrap the words in "quotes".

Be careful using too many keywords. If every word is a keyword, then their value is diminished. Never make common words like "the", "a", "he", "what" keywords, if you require one of these words for a match use a required word instead.

You can also review all of your bot's keywords by searching for 'words' and restrict to 'keywords'. This lets you remove a word from being a keyword if you added it by mistake.

required:

A required word is a word that is required to be in the question for a specific response.

```
// This example requires the words 'like' and 'hockey'.
question: do you like hockey
response: Yes, I like hockey.
required: like hockey

// This example requires the compound word 'ice hockey'.
question: do you like ice hockey
response: Yes, I like ice hockey.
required: "ice hockey"

// This example requires one of the words 'like' or 'love', and 'hockey'.
question: do you like ice hockey
response: Yes, I like ice hockey.
required: (like love) hockey

// This example requires a pattern.
```

```
question: are you okay
response: Yes, I am okay.
required: Pattern("^ are you (ok okay)")
```

A required word can be used to ensure the bot does not choose a response if the required word is missing from the question. For example the question "Do you play hockey?" would normally be matched to this response, but because it is missing the required word "like", it will not be considered as a match.

All of the required words must be contained in the question in order for the response to be used. Required words also supports lists, compound words, and patterns. To require one of a set of words a list can be used using brackets i.e. (like love). To require a compound word quotes are used i.e. "hello world". To require a pattern enter the pattern i.e. Pattern("what is *").

Previous

A previous responses is the bot's preceding response. Previous responses can be used to give a response a context.

```
// This example uses previous to give different answers to 'yes' depending on the context.
question: yes
response: Great, wanna go out sometime?
require previous: Are you single?

question: yes
response: I am please to hear that.
require previous: Are you happy?
```

The question "Yes" has multiple responses. The bot can use the previous response to choose the best response. You can also invalidate a previous response to ensure the bot does not use a response if the previous response was preceding.

A previous response can either be 'required' or optional. If required, the response will never be used unless one of the previous response matches. If optional (default) the response is given precedence if the previous response matches, but can still be used if the previous does not match.

topic:

A topic is the general topic category for the response. For example a response on a hockey related question would belong to the "hockey" topic. You can use topics to categories your responses and give the bot additional context. If the bot has multiple responses to a question, it will use the one that matches the current topic.

Topics can be either optional or required. The response topic is optional by default, it you click on 'Require Topic', then the response will only be used if that topic is active in the conversation. If optional the topic will be set as active for the conversation, until a new topic is set.

Topics can also be defined as exclusive. An exclusive topic defines an isolated conversational space. For a normal topic that is not exclusive the bot can still use responses that do not have the same topic, but responses with the active topic will be given precedence over responses without a topic. For exclusive topics only response that have the same topic as the active topic will be used. No other responses will be used (other than possibly the default response if there is no topic default).

Topics can also be set on greetings to start the conversation with a topic. Topics can be set on default responses to provide a different response when that topic is active.

```
// Categorizes the product response.
question: what are your products
response: Our products include ACME CRM and ACME ERP.
topic: products

// Use an exclusive topic to define a joke conversational dialog.
question: joke
response: Knock knock.
exclusive topic: joke

question who is there
response: Boo.
require topic: joke
```

```
question: boo who
response: Don't cry. lol
require topic: joke
think: conversation.topic = null;

default: Say 'Who is there?'
require topic: joke

default: Say 'Boo who?'
require topic: joke
previous: boo
```

label:

A response can be given an intent label to let you reuse the same response in other questions. You can also use labels in a response's previous.

```
// Use a label to reuse the common products response.
question: products
response: Our products include ACME CRM and ACME ERP.
label: #products

question: what are your products
response: #products

question: what software do you sell
response: #products

question: are they good
response: Our products are the best in the world.
previous: #products
```

on repeat:

If the user asks the same question, or similar questions multiple times, you may want your bot to give a different response the second time. On repeat lets you set one or many responses to use the second time that response is triggered in the same conversation. You can also set 'No Repeat' to never repeat the response in the same conversation.

```
// Offer to escalate the issue if the user repeatedly asks for help.
question: help
response: How can I help you?
on repeat: What is the issue that you are having?
on repeat: Would you like me to have a support staff contact you?

question: goodbye
response: Before you go, would you like to take a survey?
no repeat:

question: goodbye
response: Goodbye.
```

condition:

Conditions let you enter Self code that must evaluate to true for the response to be used. This is an advanced properties as it requires you understand Self (a dialect of JavaScript), but can be used to do very powerful things.

```
// Response to goodbye differently based on the time of day.
question: goodbye
response: Goodnight.
condition: Date.get(#hour, Date.time()) > 18

question: goodbye
response: Goodbye.
```

think:

Think let you enter Self code that is executed when the response is used. This is an advanced properties as it requires you understand Self (a dialect of JavaScript), but can be used to do very powerful things.

```
// Track the user's name.
pattern: my name is *
template: Pleased to meet you {star}.
think: speaker.name = star;

question: what is my name
template: Your name is {speaker.name}.

question: can we change the subject
response: Sure, what would you like to talk about?
think: conversation.topic = null;
```

confidence:

You can enter multiple responses for the same question. The bot will use the response that it is the most confident in for the current conversational context. By default a response is given a confidence level of 90%, but you can customize the value. Note that in a group discussion such as when the bot is added to a chatroom, or on Twitter, the bot will only use responses that have >=90% confidence.

sentiment:

You can associate a user question or phrase, or individual words with sentiment (good vs bad). Sentiment can be used to track how the user is feeling about the current conversation. The sentiment statistic is tracked by your bot daily and can be graphed from its Analytics page.

```
question: thank you
response: You are most welcome.
keywords: thank
sentiment: good

question: you suck
response: Sorry, I am doing my best. Would you like me have someone contact you via email?
keywords: suck
sentiment: bad

word: suck
sentiment: bad

word: great
sentiment: great
```

Next and Conversation Flows

Next questions in a response can be used to handle follow-up questions and conversation flows. To start a conversation flow, indent the next follow-up question to the response. You can use tabs or 4 spaces for indents. You can also indent multiple levels to define a conversation tree.

A respones next question defines an isolated conversation space. The bot will only choose its next response from the provided responses. To define a default response use the default: tag. If the bot cannot match the user's question with a next question, it will search the responses parent, otherwise use the bot's default response.

Next is similar to previous, but unlike previous is isolated to the response. In the context of a next response, the bot will never use any of its responses outside the context of the next response. This makes is easy to define isolated and nested conversations.

```
question: help
response: Do you want help with <button>web</button> or <button>mobile</button>
keywords: help

        question: web
        response: Is you issue with <button>HTML</button> or <button>JavaScript</button>?

                question: html
                response: See https://www.w3schools.com/html

                question: javascript
                response: See https://www.w3schools.com/js

        question: mobile
        response: Are you using <button>Android</button> or <button>iOS</button>?

                question: android
                response: See https://www.android.com

                question: ios
                response: See https://developer.apple.com/ios

        question: quit
        response: Okay. Let me know if you need further help.

        default: Please specify <button>web</button> or <button>mobile</button>, or type
<button>quit</button> if you do not need help.
```

Greetings

A greeting is the bot's first response in a new conversation. Greetings can have most of the same tags as responses. To define a greeting the greeting: tag is used.

```
greeting: Welcome to my website. I am Julie, how may I be of service?
```

Default Responses

If the bot does not find any good matching question to a user's input, it uses its default response. Default responses can have most of the same tags as responses. To define a default response the default: tag is used.

```
default: Sorry, I do not understand.

default: Sorry, I do not understand. Please email sales@acme.com for more information.
topic: Sales
```

Phrases

Phrases can be defined to associate sentiment, emotions, action, or poses to a phrase.

```
thanks for the help
sentiment: good

you are not helping me
sentiment: bad
```

Words

Word can be defined to associate sentiment, emotions, and synonyms. A word can also be a keyword, a topic, or an exclusive topic.

```
word: good
sentiment: good
synonyms: great wonderful cool amazing

word: bad
```

```
sentiment: bad
synonyms: sucks terrible horrible

word: aiml
keyword: true
topic: true
```

Word Tags

Tag	Description
word:	Define a word.
sentiment:	Optionally you can associate a sentiment with the word.
emotions:	Optionally you can associate an emotion with the word.
keyword:	Define the word to be a keyword.
synonyms:	Optionally you can give a synonym to the word, so that similar words in a question to trigger the same response.
topic:	Define the word to be a topic.
exclusive topic:	Define the word to be an exclusive topic.

See Also

- How to train your customer service bot by monitoring its chat logs, using keywords and topics.
- What are the supported response and chat log formats for importing and exporting?
- What are topics?
- What are labels, and how to reuse responses?
- What are previous responses?
- What are required words?
- What are keywords?

Self

Overview

Self is Bot Libre's JavaScript dialect that has been extend to support natural language processing. You can program your bot using the Self scripting language from your bot's Script page. You can also use Self from response templates, thinks, and conditions, and from the AIML "self" tag.

The Self scripting language is a state machine based language for scripting chat bots, it also enables bots to alter and program their own scripts.

Classes

Class	Description
Object	Self defines several object method available to all objects.
String	String processing methods.
Array	Array processing methods.
Language	Natural language processing methods.
Date	Date and time processing methods.
Utils	Common utility methods.
Math	Math operations.td>
Http	HTTP, HTML, XML, JSON and web service processing methods.
JSON	JSON processing methods.
Facebook	Facebook methods.
FacebookMessaging	Facebook messaging methods.
Twitter	Twitter messaging methods.
Telegram	Telegram messaging methods.
Email	Email methods.
Twilio	Twilio & SMS methods.
Vision	Image and vision messaging methods.
Context	Context access methods.
Avatar	Avatar, emotion, poses, action, and command accessing methods.
Mood	Mood and emotion methods.

Operators

Operators	Description	Example
if	If statement.	if (value == null) { ... } else { ... }
for	For statement.	for (word in sentence.word) { ... }
while	While statement.	while (count < 10) { ... }
do	Do statement.	do { ... }
think	The same as *do* but can be used inside a Template to perform some code but not print a value into the response.	think { ... }
return	Return the value.	if (word == "hello") { return "hello there"; }
==	Compare if two values match.	apple == apple
!=	Compare if two values don't match.	apple != orange
<	Compare if a value is less than another.	if (time < 10) { greeting = "Good Morning!"; }
<=	Compare if a value is less or equal than another.	if (time <= 12) { greeting = "Have a Nice Day!"; }
>	Compare if a value is greater than another.	if (time > 12) { greeting = "Good Afternoon!"; }
>=	Compare if a value is greater or equal than another.	if (time >= 5) { greeting = "Nice Day!"; }
!	negates a logical value	negative = !negative;
&&	Logical AND.	if (x < 5 && y > 2) { ... }
||	Logical OR.	if (x < 5 || y > 1) { ... }
=	Variable assignment.	name = "Alice";
++	Increment a variable.	var step; for (step = 0; step < 5; step++) { ... }
--	Decrement a variable.	var step; for (step = 10; step < 5; step--) { ... }
+	Add two numbers or concatenate two strings.	txt1 = "John"; txt2 = "Smith"; txt3 = txt1 + " " + txt2;
-	Subtract two numbers.	var x = 5; var y = 2; var z = x - y;
*	Multiply two numbers.	var x = 5; var y = 2; var z = x * y;

Operators	Description	Example
/	Divide two numbers.	var x = 10; var y = 2; var z = x / y;
new	Construct a new object.	response = new Sentence();
Symbol	Create a new global symbol.	Language.define(word, Symbol(word));
.	Get a relationship from an object.	age = speaker.age;
=	Set a relationship on an object.	response.word[0] = "Hello";
=+	Add a relationship on an object.	sentence.response =+ response;
=-	Remove a relationship on from object.	sentence.response =- response;
random	Select and execute random value.	random("Hello", "Hi", "Hey", "G'day mate");
redirect	Evaluate the response to the phrase.	redirect("what is " + star);
srai	Synonym for redirect (AIML syntax).	srai("hello");
request	Evaluate the response to the phrase using a remote service.	request(song, { service : #wikidata, hint : "performer"});
sraix	Synonym for request (AIML syntax).	sraix(song, { service : #wikidata, hint : "performer"});
learn	Learn a new response.	learn({pattern:"hello", template:"how are you" });
eval	Evaluate code within a learned pattern or template.	learn({pattern:Pattern("what is {eval (star[0])}"), template:Template("{eval (star [1])}") });
debug	Print the arguments to the log.	debug(star); debug (#error, error)

Object

Object Methods	Description	Example
add(key, value)	Add a relationship value to the object.	speaker.add(#name, "Bob")
all(key)	Returns an array of all of the object's relationship values for the type.	speaker.all(#name)
addWithMeta(key, vale, metaType, meta)	Add a relationship value to the object with the relationship meta data.	
append(key, value)	Append the value to the end of the object's relationship.	response.append(#word, ".")
appendWithMeta(key, vale, metaType, meta)	add a relationship value to the object with the relationship meta data	
copy()	Return a shallow copy of the object.	object.copy()
dataType()	Return the primitive data-type of the object or null.	object.dataType() == #String
delete()	Delete the object, (use this with caution).	object.delete()
delete(key, value)	Delete the relationship.	object.delete(#type, value)
deleteAll()	Delete all relationships.	object.deleteAll(),
deleteAll(key)	Delete all the relationships of the type.	object.deleteAll(#type),
findReference()	Inverse references lookup. Returns first reference.	
findReferenceBy(key)	Inverse relationship lookup. Returns first reference.	
findReferences()	Inverse references lookup. Returns array of objects.	
findReferencesBy(key)	Inverse relationship lookup. Returns array of objects.	
get(key) get(key, index)	Get a relationship value from an object, optional index.	speaker.get(#name), sentence.get(#word, 3)
getAccessCount()	Return the object's access count.	object.getAccessCount() > 100
getAccessDate()	Return the object's access date.	object.getAccessDate()
getAccessDate(key, value)	Return the relationship's access date.	object.getAccessDate(#type, value)
getConsciousnessLevel()	Return the object's consciousness level.	object.getAccessDate()
getConsciousnessLevel(key, value)	Return the relationship's consciousness level.	object.getAccessDate(#type, value)
getCorrectness(key, value)	Return the relationship's correctness.	object.getCorrectness(#type, value)
getCreationDate()	Return the object's creation date.	object.getCreationDate()
getCreationDate(key, value)	Return the relationship's creation date.	object.getCreationDate(#type, value)
getGroupId()	Return the object's group id.	object.getGroupId()
getId()	Return the object's unique id.	object.getId()
getId(key, value)	Return the relationship's unique id.	object.getId(#type, value)
getIndex(key value)	Return the relationship's index.	list.getIndex(#element, value)
getKey(value)	Return the related objects relationship key.	object.getKey(value)
getLast(key) getLast(key, start)	Get a relationship value from the end of an ordered relationship.	conversation.getLast(#input, 1)
getName()	Return the object's primitive name.	object.getName()
getWithAssociate(key, associate, associateType)	Get a relationship value from an object most associate to the other value.	
has(key, value)	Return if the relationship exists.	speaker.has(#name, "Bob")
hasAny(key)	Return if the object has any relationship of the type.	object.hasAny(#type)
hasData()	Return if the object has primitive data (String, Number, Date, etc.).	object.hasData()
hasMeta(key, value)	Return if the relationship has a meta value.	object.hasMeta(#type, value)
isArray()	Return if the object is an array.	object.isArray()
isPinned()	Return if the object is pinned.	object.isPinned()

Object Methods	Description	Example
isPinned(key, value)	Return if the relationship is pinned.	object.hasMeta(#type, value)
isSymbol()	Return if the object is a primitive symbol.	object.isPrimitive()
keys()	Return an array of the object relationship keys.	for (key in object.keys()) { .. }
meta(key, value)	Return the relationship's meta value.	object.meta(#type, value)
pin()	Pin the object.	object.pin()
pin(key, value)	Pin the relationship.	object.pin(#type, value)
random(key)	Return a random element of the relationship type.	object.random(#type)
remove(key, value)	Remove the relationship value (this creates an inverse relationship).	speaker.remove(#name, "Bob")
removeWithMeta(key, value, metaType, meta)	Remove the relationship value with the relationship meta data.	
set(key, value) set(key, value, index)	Sets a relationship value on an object.	speaker.set(#age, 44)
setCorrectness(key, value, value)	Sets a relationship value on an object.	speaker.setCorrectness(#gender, #male, 0.5)
size(key)	Returns the number of relationships of the type.	object.size(#type)
toJSON()	Convert the object to a JSON string.	object.toJSON()
toXML()	Convert the object to an XML string.	object.toXML()
toString()	Convert the object to string.	object.toString()
unpin()	Unpin the object.	object.unpin()
unpin(key, value)	Unpin the relationship.	object.unpin(#type, value)
weakAdd(key, value)	Add a relationship value to the object with a low correctness.	
weakAddWithMeta(key, value, metaType, meta)	Add a relationship value to the object with a low correctness with the relationship meta data.	

String

String Methods	Description	Example
charAt(index)	Return the string's character at the index.	"hello".charAt(0) == "h"
concat(text)	Concatenate the two strings.	"hello".concat(" world!") == "hello world!"
endsWith(text)	Return if the string ends with the text.	"hello world".endsWith("world") == true
exec(text)	Convert the string to a regex pattern and evaluates if it matches the text and return the text match.	"/\d+".exec("123") == "123"
indexOf(text)	Return the index of the substring or -1 if missing.	"hello world".indexOf("hello") == 0
lastIndexOf(text)	Return the last index of the substring or -1 if missing.	"hello world hello".indexOf("hello") == 12
length()	Return the string's length.	"hello".length() == 5
match(text)	Convert the string to a regex pattern and return an array of all matching values.	"/\d+".exec("123 plus 456") == ["123" "456"]
replace(token, text)	Replace all occurances of the token string with the text.	"hello world".replace("hello", "goodbye") == "goodbye world"
setCharAt(index, char)	Return a string with the string's character at the index.	"hello".setCharAt(0, "H") == "Hello"
size()	Return the string's length.	"hello".size() == 5
startsWith(text)	Return if the string starts with the text.	"hello world".startsWith("hello") == true
substr(start, end)	Return the string substring.	"hello world".substr(0, 4) == "hell"
substring(start, end)	Return the string substring.	"hello world".substring(0, 4) == "hell"
test(text)	Convert the string to a regex pattern and return if it matches the text.	"/\d+".test("123") == true
toLowerCase()	Return the string as lower case.	"HELLO".toLowerCase() == "hello"
toUpperCase()	Return the string as upper case.	"hello".toUpperCase() == "HELLO"
toNumber	Convert the string to a number.	"123.4".toNumber()
toSymbol	Convert the string to a primitive symbol.	"topic".toSymbol()
trim()	Trim leading and trailing whitespace.	" Hello World! ".trim() == "Hello World!"

Array

Array Methods	Description	Example
add(value)	Add the value to the end of the array's elements.	array.add("hello")
delete(value)	Delete the value from the array's elements.	array.delete("hello")
has(value)	Return if the array contains the element.	array.has("hello")
indexOf(value)	Return the index of the element in the array.	array.indexOf("hello")
indexOf(value, start)	Return the index of the element in the array starting at the index.	array.indexOf("hello", 4)
lastIndexOf(value, start)	Return the index of the element in the array starting at the end.	array.lastIndexOf("hello")
length()	Return the arrays length.	[1, 2, 3].length() == 3
random()	Return a random element of the array.	array.random()
size()	Return the arrays length.	[1, 2, 3].length() == 3

Language

Language Methods	Description	Example
word(text*)	Creates a compound word.	Language.word("ball", "hockey")
sentence(text*)	Creates a sentence.	Language.sentence("How", "are", "you")
define(text, object)	Defines the word as meaning the value.	Language.define("foobar", #foobar)

Date

Date Methods	Description	Example
date()	Current Date object.	var today = Date.date();
date(text)	Convert the text or Timestamp to a Date.	var xmas = Date.date("2017-12-25");
time()	Current Time object.	var time = Date.time();
time(text)	Convert the text or Timestamp to a Time.	var noon = Date.time("12:00:00");
timestamp()	Current Timestamp object.	var now = Date.timestamp();
timestamp(text)	Convert the text or Date to a Timestamp.	var timestamp = Date.timestamp("2017-10-02 16:52:30");
any(text)	Parse any date/time format.	var name = Date.parse("Sept 3, 2014")
add(date, part, time)	Add the date value (#day, #month, #year, #hour, #minute, #second, #millisecond).	var tomorrow = Date.add(Date.date(), #day, 1);
get(date, part)	Return the date part (#day, #month, #year, #hour, #minute, #second, #millisecond).	var day = Date.get(Date.date(), #day);
set(date, part, value)	Set the date part (#day, #month, #year, #hour, #minute, #second, #millisecond).	var day = Date.set(Date.date(), #day, 11);
difference(from, to, part)	Compute the date/time difference from date part (#day, #month, #year, #hour, #minute, #second, #millisecond).	var days = Date.difference(Date.date(), Date.date ("2017-12-25"), #day);
interval(part, from, to)	Compute the date/time interval from date/time formatted strings ("days", "months", "years", "hours", "minutes", "seconds", "milliseconds").	var days = Date.interval("days", "2017-11-20", "2017-12-25")
interval(part, from, to, format)	Compute the date/time interval from date/time Java SimpleDateFormat formatted strings ("days", "months", "years", "hours", "minutes", "seconds", "milliseconds").	var days = Date.interval("days", "2017-11-20", "2017-12-25", "yyyy-MM-dd")
getTimeZone()	Return the default time zone.	var gmt = Date.getTimeZone();
getTimeZone(zone)	Return the timezone object based on its timezone code i.e. "GMT", "GMT-5", "PST", "America/Los_Angeles".	var gmt = Date.getTimeZone("GMT");
setTimeZone(date, timezone)	Set the dates time zone based on its timezone code i.e. "GMT", "GMT-5", "PST", "America/Los_Angeles".	var est = Date.setTimeZone(Date.date(), "EST");
printAIMLDate(date, format)	Print the date using the AIML format.	
printDate(date, format)	Print the date using Java SimpleDateFormat format.	var dateString = Date.printDate(Date.date(), "yyyy-MM-dd")

Utils

Util Methods	Description	Example
encode(text)	Converts to URL encoded character.	Http.requestXML("http://api.com/fetch?name=" + Utils.encode(name))
denormalize(text)	Converts punctuation words back to characters.	Utils.denormalize("That is cool colon) at dash at") == "That is cool :) @-@"
normalize(text)	Converts punctuation characters to the word values.	Utils.normalize("That is cool :) @-@") == "That is cool colon) at dash at"
explode(text)	Expands a string to its character values.	Utils.explode("hello") == "h e l l o"
extract(text, pattern)	Convert the pattern to a regex pattern and extracts the matching string from the text.	Utils.extract("the cost is $123", "/\d+") == "123"
matches(text, pattern)	Convert the pattern to a regex pattern and returns if the text matches.	Utils.matches("123", "/\d+") == true
gender(text)	Converts between 'he' and 'she'.	Utils.gender("he likes her") == "she likes him"
person(text)	Converts between 1st and 2nd person.	Utils.person("I love myself") == "you love yourself"
person2(text)	Converts between 1st and 3rd person.	Utils.person2("I love myself") == "he love himself"
capitalize(text)	Converts the first character to upper case.	Utils.capitalize("hello there") == "Hello there"
sentence(text)	Converts the first character to upper case.	Utils.sentence("hello there") == "Hello there"
formal(text)	Converts the first character to upper case for each word.	Utils.formal("hello there") == "Hello There"
lowercase(text)	Converts to lower case.	Utils.lowercase("HELLO There") == "hello there"
uppercase(text)	Converts to upper case.	Utils.uppercase("hello there") == "HELLO THERE"
program()	Returns the current software name and version.	
size()	Returns to total knowledge base size (total objects).	
version()	Returns the current software version.	
id()	Returns the bot's internal id.	

Math

Math Methods	Description	Example
add(x, y)	Sum of x and y values.	Math.add(2, 3); 5
subtract(x, y)	Subtract y from x value.	Math.subtract(8, 5); 3
multiply(x, y)	Multiply x and y values.	Math.multiply(2, 4); 8
divide(x, y)	Divide x and y values.	Math.divide(18, 9); 2
abs(x)	Returns the absolute value of x.	Math.abs(-7.7); 7.7
floor(x)	Returns the value of x rounded down to its nearest integer.	Math.floor(3.8); 3
ceil(x)	Returns the value of x rounded up to its nearest integer.	Math.ceil(2.2); 3
power(x, y)	Returns the value of x to the power of y.	Math.power(7, 2); 49
round(x)	Returns the value of x rounded to its nearest integer.	Math.round(4.3); 4
log(x)		Math.log(8); 3
ln(x)		
sin(x)	Returns the sine of x (x is in radians).	Math.cos(0); 1
asin(x)		
cos(x)	Returns the cosine of x (x is in radians).	
acos(x)		
tan(x)	Returns the tangent of an angle.	
tanh(x)		

Http

Http	Description	Example
toJSON(object)	Convert an object to JSON text.	
toXML(object)	Convert an object to XML text.	
encode(text)	URL encode the text.	
requestJSON(url)	Fetch the JSON response.	
requestJSON(attribute, url)	Fetch the JSON response.	
requestJSON(attribute, url, headers)	Fetch the JSON response. Pass custom headers as JSON object.	
requestJSONAuth(attribute, url, user, password)	Call authenticated API and fetch the JSON response.	
requestXML(url)	Fetch the XML response.	
requestXML(url, xpath)	Fetch the XML response.	
requestXML(url, xpath, headers)	Fetch the XML response. Pass custom headers as JSON object.	
requestXMLAuth(url, user, password, xpath)	Call authenticated API and fetch the XML response.	
requestHTML(url, xpath)	Scrape the HTML page.	
requestHTML(url, xpath, format)	Scrape the HTML page, return #text, #array, #html, or #object.	
requestHTML(url, xpath, format, subformat)	Scrape the HTML page, return #array of #text, #html, or #object.	
requestCSV(url)	Parse the CSV file.	
requestText(url)	Return the raw text data.	
requestText(url, headers)	Return the raw text data. Pass custom headers as JSON object.	
rss(url)	Fetch the last RSS feed.	
rssFeed(url)	Fetch the entire RSS feed.	
postJSON(url, object)	Post the JSON object to the URL.	
postJSONAuth(url, user, password, object)	Post the JSON object to the URL.	
postXML(url, object)	POST the XML object to the URL.	
postXML(url, object, xpath)	POST the XML object to the URL. Extract the XPath data.	
postXMLAuth(url, user, password, object, xpath)	Post the XML object to the URL. Extract the XPath data.	
postHTML(url, params, xpath)	Post the HTML form parameters to the URL. Extract the XPath data.	
putJSON(object)	PUT the JSON object to the URL.	
delete(url)	Send an HTTP DELETE to the URL.	

JSON

JSON	Description	Example
parse(text)	Parse an object from JSON text.	
stringify(object)	Convert an object to JSON text.	

Facebook

Facebook	Description	Example
post(message)	Post to the bot's Facebook page.	
postComment(comment, postid)	Post to the bot's Facebook page.	
sendMessage(message, userid)	Send a message to a user.	

FacebookMessaging	Description	Example
sendMessage(message, userid)	Send a message to a user.	
sendMessage(message, userid, command)	Send a message to a user with a custom Facebook quick_reply or attachment command.	

Twitter

Twitter	Description	Example
tweet(message)	Tweet to the bot's Twitter account.	
sendMessage(message, userid)	Send a direct message to a user.	

Telegram

Telegram	Description	Example
post(message)	Post to the bot's Telegram channel.	
sendMessage(message, userid)	Send a message to a user.	
sendMessage(message, userid, command)	Send a message to a user with a Telegram custom keyboard reply_markup command.	
postJSON(url, json)	Call a Telegram bot API with the JSON parameter.	

Email

Email	Description	Example
email(address, subject, message)	Send an email from the bot's email account.	Email.email("joe@foo.com", "Welcome", "Thanks for registering.")

Twilio

Twilio	Description	Example
sms(number, message)	Send a SMS messages from the bot's Twilio account.	Twilio.sms("613-123-4567", "Escalation alert")

Vision

Vision	Description	Example
loadImage(url)	Load the image into the bot's memory.	
matchImage(url, tag, error)	Search for the closest matching image for the tag.	

Context

Context	Description	Example
push(object)	Push the object to the current context stack.	
top()	Return the top of the context stack.	
top(index)	Return the nth top of the context stack.	
search(match)	Search the context stack for a matching object.	

Avatar

Avatar	Description	Example
setAction(action)	Set the avatar's action.	
setPose(pose)	Set the avatar's pose.	
setCommand(command)	Set a JSON command object.	

Mood

Mood	Description	Example
setEmotion(emotion, value)	Set the value of the emotion (between -1.0 and +1.0).	Mood.setEmotion(#Happiness, 1.0)

State Operations

State Operation	Description	Example
state	A state defines the current input processing.	pattern "my name is *" template Template("Pleased to meet you {star}");
case	A case can transition to another state if the case variable matches the current input.	case "lol" template "Very funny."
pattern	A pattern can match an input and evaluate a template response.	pattern "hello" template "Hi there";
answer	An answer of a state is evaluated if the input processing is complete.	answer Template("Pleased to meet you {star}");
function	An function can be called from an answer or another function.	function todayResponse() { var response = new (#sentence); response.append(#word, "Hello World!"); return response; }
var	A variable can be matched with the current input, or store context.	var name { instantiation : #name; }

Pattern Operators

Operator	Description
*	Matches one or more words.
_	Matches one or more words, takes priority over all other words and wildcards, except $.
^	Matches zero or more words.
#	Matches zero or more words, takes priority over all other words and wildcards, except $.
$	Pattern priority marker to make a pattern word match highest priority.
()	Optional set of words.
[]	Required set of words.
<set>	Set tag to evaluate a pattern based on words defined in a predefined set.
/	Regular expression pattern.
{}	Self code.

Examples

Here example of a Self script that counts words:

```
// Example script that counts words.
state CountWords {
    pattern "count words in *" answer countWords();

    function countWords() {
        var words = Language.sentence(star);
        var count = 0;
        for (word in words.word) {
            count = count + 1;
        }
        return Template("There are {count} words.");
    }
}
```

Here example of a Self script that repeat words:

```



```

```
// Example script that repeat words.
state Repeat {
    pattern "repeat * by * times" answer repeat();

    function repeat() {
        var response = "";
        var repeat = star[1].toNumber();
        var count = 0;
        while (count < repeat) {
            response = response + " " + star[0];
            count = count + 1;
        }
        return response;
    }
}
```

Here example of a Self script that access an XML web API:

```
// This script searches Wikipedia using an XML HTTP request and an XPath expression
state SearchWikipedia {
    pattern "what is *" answer search();

    function search() {
        return Http.requestXML("https://en.wikipedia.org/w/api.php?action=opensearch&
    }
}

```

Here example of a Self script that uses patterns to perform math:

```
// Script for understanding simple math using patterns.
state SimpleAddition {
    pattern "* + *" answer (Math.add(star[0].toNumber(), star[1].toNumber()));
    pattern "* - *" answer (Math.subtract(star[0].toNumber(), star[1].toNumber()));
    pattern "* / *" answer (Math.divide(star[0].toNumber(), star[1].toNumber()));
    pattern "* x *" answer (Math.multiply(star[0].toNumber(), star[1].toNumber()));
}

```

The state machines are more complex, but allow you to parse arbitrarily complex expressions. Here example of a Self script use a state machine and variables for processing addition:

```
// Language state machine for understanding simple {x+y} addition.
state SimpleAddition {
    // This line is required for any state processing to process each word in the
    case input goto sentenceState for each #word of sentence;

    state sentenceState {
        case left goto numberState;

        // The left variable is a stand-in for any word that represents a number,
        // The meaning is required because in Self words are just words that repr
        var left {
            meaning : number;
        }
        var number {
            instantiation : #number;
```

See Also

- Introducing the Self Scripting Language
- What classes are supported in Self

AIML

Overview

AIML is the Artificial Intelligence Markup Language. It is an XML standard for defining chat bot responses. Bot Libre bots support AIML, but are based on Self and have a knowledge base similar to the human brain.

About AIML

AIML defines how a bot should respond to a question using <pattern> and <template> elements. A pattern represents the user's question, and the template defines the bot's response. The pattern and template elements are grouped in a category element, which can be grouped under topic elements.

List of AIML template tags

Tags	Description
<star index="N"/>	Replaced with value of * in pattern (or Nth *).
<that index="M,N"/>	Replaced with value of bot's previous response (or Mth previous response, and N sentence of the response).
<input index="N"/>	Replaced with value of users's input (or Nth *).
<thatstar index="N"/>	Replaced with value of * in "that" (or Nth *).
<topicstar index="N"/>	Replaced with value of * in topic (or Nth *).
<get name="XXX"/>	Replaced by the value of the conversation specific variable.
<bot name="XXX"/>	Replaced by the value of the bot specific variable.
<sr/>	Short form for <srai><star/><srai> (replaced with response to value of * from pattern).
<person2/>	Converts the text (or <star/>) between 1st and 2nd person (I <-> he, etc.).
<person/>	Converts the text (or <star/>) between 1st and 3rd person (I <-> you, etc.).
<gender/>	Converts the text (or <star/>) between male and female (he <-> she).
<date/>	Replaced with the current date and time, a "format" attribute is also supported.
<id/>	Replaced by the client id.
<size/>	Replaced with the size of bot's memory.
<version/>	Replaced with the AI engine version.
<uppercase>	Converts the text to uppercase.
<lowercase>	Converts the text to lowercase.
<formal>	Converts all words in the text to be capitalized.
<sentence>	Converts the first word in the text to be capitalized.
<condition name="X" value="Y">	Defines an "if" condition based on comparing the value of a variable to a pattern.
<condition name="X">	Case statement.
<condition>	Multi-valued if/else statement.
<random>	Choose on of the nested <li> values at random.
<li name="X" value="Y">	Used in random and condition tags.
<li value="Y">	Used in random and condition tags.
<li>	Used in random and condition tags.
<set name="XXX">	Set the value of a variable.
<gossip>	Logs the text.
<srai>	Recursively evaluates the text and replaces it with the response.
<think>	Evaluates the nested statements but does not output any result.
<learn>	Load external AIML file, this is not currently allowed.
<system>	Executes a OS command, this is not currently allowed.
<javascript>	Executes JavaScript code.
<topic>	Category topic tags can now be set on a category to make it easier to add categories to topics.
<map>	Maps tag to allow the lookup of the element value in a predefined mapping, returning the mapped value.
<loop>	Condition loops tag to loop a conditional statement.
<var>	Local variables attribute for variables scoped to a category.
<sraix>	Remote requests tag to make a remote request of another bot instance, or service.
<normalize> and <denormalize>	Normalization and denormalization tags to convert special character into words, and back again.
<request>	Request tag to return the user's previous input request.
<response>	Response tag to return the bot's previous response.
<learn>, <learnf>, and <eval>	Learning tags to dynamically train a bot with new responses.
<explode>	Explode tag to split a word into its characters.
<oob>	Mobile (out of band) tag to support client and mobile device commands.
(<b>, <a>, <i>, <ul>, <ol>, <li>, <p>, <br>, <button>, <img>, <video>)	Various HTML tags are also allowed.
<self>	Evaluate Self code (Bot Libre extension).

Tags	Description

List of AIML pattern operators

Tags	Description
*	Matches one or more words.
_	Matches one or more words, takes priority over all other words and wildcards, except $.
^	Matches zero or more words.
#	Matches zero or more words, takes priority over all other words and wildcards, except $.
$	Pattern priority marker to make a pattern word match highest priority.
()	Optional set of words (Bot Libre extension).
[]	Required set of words (Bot Libre extension).
<set>	Set tag to evaluate a pattern based on words defined in a predefined set.
/	Regular expression pattern (Bot Libre extension).
{}	Self code (Bot Libre extension).

Examples

Here is a simple 'Hello World' AIML example:
This code will make your bot respond with 'Hello to you too' to the question 'Hello World'.

```
<aiml>
    <category>
        <pattern>HELLO WORLD</pattern>
        <template>Hello to you too</template>
    </category>
</aiml>

```

Here is a simple wildcard example:
This code will make your bot respond with 'Hello to you too' to any question containing the word 'hello'.

```
<aiml>
    <category>
        <pattern>* HELLO *</pattern>
        <template>Hello to you too</template>
    </category>
</aiml>

```

Here is a joke 'that' example:
This code will make your bot tell a joke.

```
<aiml>
    <category>
        <pattern>TELL ME A JOKE</pattern>
        <template>Why did the chicken cross the road?</template>
    </category>
    <category>
        <pattern>* WHY *</pattern>
        <that>Why did the chicken cross the road?</that>
        <template>To get to the other side.</template>
    </category>
</aiml>

```

Here is a complex template example:
This code will make your bot respond to the question 'Do you like apples or do you like oranges' with 'I love apples and I love oranges'.

```
<aiml>
    <category>
        <pattern>* OR *</pattern>
        <template><srai><star/></srai> and <srai><star index="2"/></srai></template
    </category>
    <category>
        <pattern>DO YOU LIKE *</pattern>
        <template>I love <star/></template>
    </category>
</aiml>

```

^ and # Wildcards

```
<category>
    <pattern>^ aiml ^</pattern>
    <template>AIML is cool.</template>
</category>

<category>
    <pattern>^ chat bot^</pattern>
    <template>Yes, I am a chat bot.</template>
</category>

```

$ Priority

```
<category>
    <pattern># sex #</pattern>
    <template>I'm not that kind of bot.</template>
</category>

<category>
        <pattern>$what is your sex</pattern>
        <template>I am female.</template>
</category>

```

() and [] Lists

```
<category>
    <pattern>do [you u] (really) [like love luv lv] me</pattern>
    <template>Yes, I love you.</template>
</category>

```

Regular expressions

```
<category>
    <pattern>my email is /.+\@.+\..+</pattern>
    <template>That is a valid email</template>
</category>
```

Sets

```
<category>
    <pattern>I am <set>name</set></pattern>
    <template>Pleased to meet you <star/>.</template>
</category>

<category>
    <pattern><set>number</set> + <set>number</set></pattern>
    <template><srai>addition <star/> <star index="2"/></srai></template>
</category>

```

AIML template to set a knowledge object's type:

```
<category>
    <pattern>load animals</pattern>
    <template>
        <think>
            <map name="meaning" value="dog">dog</map><map name="instantiation" valu
            <map name="meaning" value="dog">doggy</map>
            <map name="meaning" value="cat">cat</map><map name="instantiation" valu
        </think>
    </template>
</category>

```

Pattern tags

```
<category>
    <pattern><bot name="name"/></pattern>
    <template>Yes, that's my name.</template>
</category>

<category>
    <pattern>I am </pattern>
    <template>That is my age too.</template>
</category>

<category>
    <pattern></pattern>
    <template>You are repeating yourself.</template>
</category>

<category>
```

Attributes as Elements

```
<category>
    <pattern>My * is *</pattern>
    <template>I will remember that your <star/> is <star index="2"/><think><set><name><star/></name><star index="2"/></set></think></te
</category>
<category>
    <pattern>What is my *</pattern>
    <template>Your <star/> is <get><name><star/></name></get></template>
</category>
```

Maps

```
<category>
    <pattern>what is the capital of <set>country</set></pattern>
    <template>The capital of <star/> in <map name="capital"><star/></map>.</template
</category>

```

Bot Libre extends the **map** operation to also allow setting the map value. This will set the value of a relationship on the knowledge object.

```
<category>
    <pattern>the opposite of * is *</pattern>
    <template>Okay, the opposite of <star/> is <star index="2"/>.
        <think><map name="opposite"><value><star index="2"/></value><star/></map></
    </template>
</category>

<category>
        <pattern>what is the opposite of *</pattern>
        <template>the opposite of <star/> is <map name="opposite"><star/></map>
        </template>
</category>

```

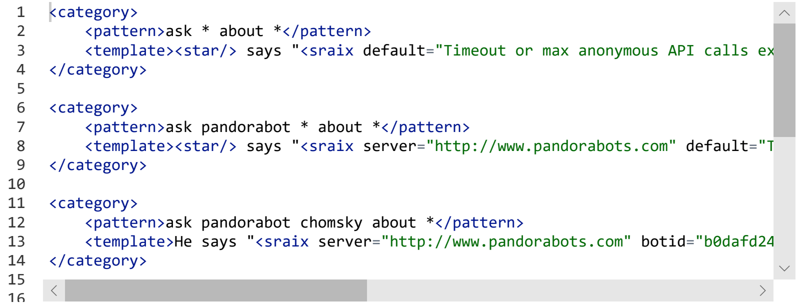
Conditions and Loops

```
<category>
    <pattern>count down from <set>number</set></pattern>
    <template>
    <think><set var="count"><star/></set></think>
    <condition var="count">
        <li value="5"><think><set var="count">4</set></think>5 <loop/></li>
        <li value="4"><think><set var="count">3</set></think>4 <loop/></li>
        <li value="3"><think><set var="count">2</set></think>3 <loop/></li>
        <li value="2"><think><set var="count">1</set></think>2 <loop/></li>
        <li value="1"><think><set var="count">0</set></think>1 </li>
    </condition>
    </template>
</category>

<category>
```

Learning

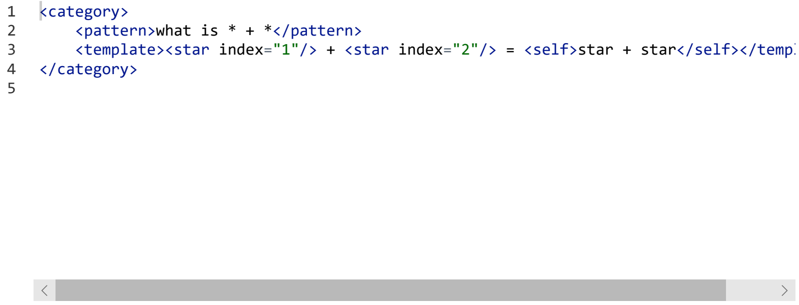
```
<category>
    <pattern>learn question * answer *</pattern>
    <template>
    <think>
        <learn>
            <category>
                <pattern><eval><star/></eval></pattern>
                <template><eval><star index="2"/></eval></template>
            </category>
        </learn>
    </think>
    Okay, I will answer "<star index="2"/>" to "<star/>" next time
    </template>
</category>

<category>
```

Sraix



```
<category>
    <pattern>ask * about *</pattern>
    <template><star/> says "<sraix default="Timeout or max anonymous API calls ex
</category>

<category>
    <pattern>ask pandorabot * about *</pattern>
    <template><star/> says "<sraix server="http://www.pandorabots.com" default="T
</category>

<category>
    <pattern>ask pandorabot chomsky about *</pattern>
    <template>He says "<sraix server="http://www.pandorabots.com" botid="b0dafd24
</category>


```

Self



```
<category>
    <pattern>what is * + *</pattern>
    <template><star index="1"/> + <star index="2"/> = <self>star + star</self></temp
</category>

```

See Also

- Scripting your bot with AIML
- What's new in AIML 2.0

Regular Expressions (Regex)

Overview

Bot Libre supports using Regular Expressions in patterns, templates, and scripts.

Regular Expressions, or Regex defines a pattern syntax for parsing text. Unlike AIML and Bot Libre patterns Regex patterns are character based, not word base, so can match specific types of words and word sequences such as numbers, dates, times, currency, and others.

For example, the following regex matches a number,

```
/\d+
```

and this regex would match a date,

```
/^(19|20)\d\d[-/.](0[1-9]|1[012])[-/.](0[1-9]|[12][0-9]|3[01])$
```

Bot Libre allows regex expressions to be used in AIML patterns, and in Bot Libre response patterns. Bot Libre's scripting language Self also allows regex in patterns and provides extractor functions that allow regex to be used to extract data from a user's input.

To define a regex pattern in an AIML or Bot Libre pattern just start the regex with the "/" character.

AIML

AIML defines pattern wildcards such as * and ^ which can match multiple words in a phrase, but they will match any word, and are not restricted to specific types of words. Bot Libre lets you include regex inside AIML patterns to match specific types of words. Just like the * wildcard the word that was matched by the regex can be accessed in the template using the <star/> tag.

```
<category>
    <pattern>my email is /.+\@.+\..+</pattern>
    <template>Okay, I will email you at <star/></template>
</category>
```

Normally regex is used to match a specific word, but you can also use regex to match and entire phrase if it defines the entire pattern.

For this to work the entire pattern must be the regex, and the pattern can have no other words. The "()" characters in regex define a group which becomes the star variable(s).

```
<category>
    <pattern>/(?i)what\sis\s(.*)</pattern>
    <template>I have no idea what <star/> is.</template>
</category>
```

Patterns

Patterns and regex can also be used in Bot Libre response lists similar to AIML.

```
Pattern("my email is /.+\@.+\..+")
Template("Okay, I will email you at {star}")

Pattern("/(?i)what\sis\s(.*)")
Template("I have no idea what {star} is.")
```

In a response list template you can also use Self extractor functions.

```
I am 22 years old
Template("I will remember that you are { var age = sentence.exec("\d+"); speaker.age = age; age } years
old.")
```

Self

Regex can also be used in Self patterns and functions.

```
state Math {
    pattern "^ /\d+ \* /\d+ ^" template "{star[1].toNumber() * star[2].toNumber()}";
    pattern "^ /\d+ / /\d+ ^" template "{star[1].toNumber() / star[2].toNumber()}";
    pattern "^ /\d+ \+ /\d+ ^" template "{star[1].toNumber() + star[2].toNumber()}";
    pattern "^ /\d+ \- /\d+ ^" template "{star[1].toNumber() - star[2].toNumber()}";
}
```

The following are regex functions in Self:

- Utils.matches(text, regex) - return if the regex matches the text

  ```
  Utils.matches("12345", "\d+") == true
  ```

- String.test(text) - return if the regex string it matches the text

  ```
  "\d".test("hello 123") == true
  ```

- String.exec(text) - extract the subtext matching the regex string from the text

  ```
  "\d+".exec("hello 123") != "123"
  ```

- String.match(text) - returns an array of all values matching the regex string extracted from the text

  ```
  var values = "hello 123 456".match("\d+"); values[1] == "456"
  ```

Bot Libre also defines several symbols for common regex patterns.
These include:

- #number
- #date
- #email
- #url

These symbols can be used in place of regex patterns in Self and patterns.

```
<category>
    <pattern>my email is #email</pattern>
    <template>Okay, I will email you at <star/></template>
</category>
```

```
state Email {
    pattern "^ #email ^" topic "email" template "Thank you, I will remember your email. { think {
speaker.email = Utils.extract(sentence, #email); conversation.topic = null; } }";
    pattern "*" topic "email" template "Please enter a valid email.";
}
```

See Also

- Parsing natural language using Regular Expression patterns and extractors
- https://www.w3schools.com/jsref/jsref_obj_regexp.asp
- https://regexr.com/