DISCORD BOTS ERSTELLEN - INTERAKTIV - LIVE - FÜR ANFÄNGER - PRÄSENTATION - OFAHRT 2025

(Keine FOPBots)

THEORIE

WIE FUNKTIONIEREN DISCORD BOTS?

- Eine Form von Discord Applications
- Können auf Servern installiert werden
- Euer Code interagiert mit der Discord API, damit der Bot coole Sachen macht

CODE-GRUNDLAGEN

- Als Library nutzen wir JDA
- Es existieren Libraries für jede erdenkliche Sprache

CODE-GRUNDLAGEN

- Als Library nutzen wir JDA (außer Nikola, der macht Racket)
- Es existieren Libraries für jede erdenkliche Sprache

LIBRARIES (SYMBOLBILD)

available!
- Discord.Net – An unofficial .Net wrapper for the Discord API (https://discord.com/)
- Discord.Net-Labs **(archived)** – An experimental fork of Discord.Net that implements the newest discord features for testing and development to eventually get merged into Discord.Net
- DiscordUnity – A DiscordAPI made for Unity and only usable for Unity. It contains special features to make it al work.
- Discore – A light-weight .NET library for creating Discord bots.
- Disqord – Asynchronous Discord API wrapper and bot framework for .NET.
- DSharpPlus – A .NET library for making bots using the Discord API.
- NetCord – A modern, lightweight, and customizable C# Discord library with Native AOT support, immutable caching, voice capabilities, and complete API coverage.
- Remora.Discord – A data-oriented C# Discord library, focused on high-performance concurrency and robust design.

C++
- aegis.cpp **(archived)** – Discord C++ library for interfacing with the API. Join our server: https://discord.gg/w7Y3Bb8
- DiscordCoreAPI – A bot library for Discord, written in C++, and featuring explicit multithreading through the usage of custom, asynchronous C++ CoRoutines.
- discordpp – A Modularized C++ Library for the Discord API
- DisCPP **(archived)** – Simplified, but feature rich Discord API wrapper written in modern C++.
- DPP – C++ Discord API Bot Library – D++ is Lightweight and scalable for small and huge bots!
- sleepy-discord – C++ library for the Discord chat client.

Clojure
- discljord – A Clojure wrapper library for the Discord API, with full API coverage (except voice, for now), and high scalability
- discord.clj – A Clojure library for creating Discord bots
- ring-discord-auth – Fast and secure functions and ring middleware to verify ED-25519-signed Discord interactions

Common Lisp
- lispcord – A client library for the discordapp bot api

Crystal
- discordcr **(archived)** – Minimalist Discord library for Crystal. (Still WIP, but usable)

D
- dscord **(archived)** – Discord Library for D

Dart
- nyxx – Wrapper around Discord API for Dart

Elixir
- alchemy **(archived)** – A discord library for Elixir
- concoction – Concoction is a Discord library for Elixir.
- coxir – A modern Elixir wrapper for Discord.
- crux **(archived)** – Repository creating the umbrella documentation for all crux libraries
- nostrum – Elixir Discord Library

Go
- arikawa – A Golang library and framework for the Discord API.
- discord-interactions-go – discord interactions
- discordgo – (Golang) Go bindings for Discord
- disgo – A modular Golang Discord API Wrapper
- disgord **(archived)** – Go module for interacting with the documented Discord's bot interface; Gateway, REST requests and voice
- goscord – A Discord API wrapper written in Golang.

Haskell
- calamity – A library for writing discord bots in haskell
- discord-haskell – Haskell library for writing Discord bots
- Discord.hs **(archived)** – Have you heard of our lord and savior Haskell?

Haxe

- litcord – A Lua wrapper for Discord.

Nim
- dimscord – A Discord Bot & REST Library for Nim.
- discordnim – Discord library for nim
- nimcord **(archived)** – Memory optimized, simple, and feature rich Discord API wrapper written in Nim.

PHP
- discord-interactions-php – PHP utilities for building Discord Interaction webhooks
- DiscordPHP – An API to interact with the popular messaging app Discord
- DiscordPHP-Slash **(archived)** – PHP server and client for Discord slash commands.
- Nyasmin – Dedicated to maintaining the Yasmin core used by Palace Bot #9203
- restcord – Discord REST API Client

Python
- disco **(archived)** – Discord Python library for people that like to dance
- discord-interactions-python – Useful tools for building interactions in Python
- discord.py – An API wrapper for Discord written in Python.
- dislash.py **(archived)** – A Python wrapper for discord slash-commands and buttons, designed to extend discord.py.
- disnake – An API wrapper for Discord written in Python.
- dispike – An independent, simple to use, powerful framework for creating interaction-based Discord bots. Powered by FastAPI
- enhanced-discord.py **(archived)** – An API wrapper for Discord written in Python.
- flask-discord-interactions – A Flask extension to enable declarative definitions for Discord slash commands.
- hata – Async Discord API wrapper.
- hikari – A Discord API wrapper for Python and asyncio built on good intentions.
- interactions.py – A highly extensible, easy to use, and feature complete bot framework for Discord
- NAFF **(archived)** – A Python API wrapper for Discord
- nextcord – A Python wrapper for the Discord API forked from discord.py
- pycord – Pycord is a modern, easy to use, feature-rich, and async ready API wrapper for Discord written in Python

Racket
- racket-cord – A discord library for racket

Raku
- raku-api-discord – Raku module for interacting with the Discord API.

Ruby
- discordrb – Discord API for Ruby
- vox – Discord library in ruby

Rust
- accord **(archived)** – Discord API client to power Discord API clients via the power of love, friendship, and HTTP 💞
- automate **(archived)** – An asynchronous library to interact with Discord API
- discord-rs – Rust library for the Discord chat client API
- serenity – A Rust library for the Discord API.
- songbird – An async Rust library for the Discord voice API
- twilight – Powerful, flexible, and scalable ecosystem of Rust libraries for the Discord API.

Scala
- AckCord – A Discord library for Scala using Akka

Shell
- discord.sh – Write-only command-line Discord webhooks integration written in 100% Bash script

Swift
- SwiftDiscord – Discord API Client for Swift
- Sword **(archived)** – Discord library for Swift

TypeScript
- arcscord – A discord.js framework, slash command first and full typescript !

CODE-GRUNDLAGEN

- Als Library nutzen wir JDA
- Es existieren Libraries für jede erdenkliche Sprache

```
public static void main(String[] arguments) throws Exception {
    JDA api = JDABuilder.createDefault(BOT_TOKEN).build();
}
```

- Hiermit wird eine Verbindung zur Discord API aufgebaut
- Der Bot ist jetzt online

EXKURS: .ENV FILE

- Wir wollen nicht den Bot Token direkt in unseren Code schreiben
- Deshalb erstellen wir eine Datei namens “.env” in unserem Projekt
- Und greifen dann über eine Library auf den Token zu

.env:

```
token=FHLIUADS38z92dsh32HIU98dsf2ijf893t298zfwejhiod98G82fhd
```

- Und diese comitten wir bitte nicht
  - Sonst müssen wir in der Zeit reisen

GATEWAY INTENTS

- Gateway Intents legen fest, welche Events der Bot empfangen kann
- Zum Beispiel:
  - GUILD_MESSAGES: Der Bot kann Nachrichten in Servern empfangen
  - GUILD_MESSAGE_TYPING: Der Bot kann sehen, wenn jemand eine Nachricht schreibt
  - etc.

GATEWAY INTENTS

- Einige Intents erfordern, dass euer Bot von Discord verifiziert wird
  - Allerdings nur, wenn der Bot auf mehr als 100 Servern ist
- Deshalb aktivieren wir heute einfach alle Intents

```
public static void main(String[] arguments) throws Exception {
    JDA bot = JDABuilder
            .createDefault(BOT_TOKEN)
            .enableIntents(List.of(GatewayIntent.values()))
            .build();
}
```

LISTENERS

- Listeners sind Klassen, die auf Events reagieren
  - Beispiel: Listener, der auf Nachrichten reagiert

```
public class MessageListener extends ListenerAdapter
{
    @Override
    public void onMessageReceived(MessageReceivedEvent event)
    {
       ...
    }
}
```

LISTENERS

- Damit diese Klasse auch auf Events reagieren kann, muss sie dem Bot hinzugefügt werden

```
public static void main(String[] arguments) throws Exception {
    JDA bot = JDABuilder
            .createDefault(BOT_TOKEN)
            .enableIntents(List.of(GatewayIntent.values()))
            .addEventListeners(new MessageListener())
            .build();
}
```

REST ACTIONS

- Rest Actions beschreiben, was der Bot tun soll, bevor er es tut
- Beispiel: Nachricht senden

```
Message message = event.getMessage();
RestAction<Message> action = message.reply("abc");
```

- Diese Aktion kann dann ausgeführt werden

```
action.queue();
```

REST ACTIONS

- Wenn wir mit Rest Actions arbeiten, können wir auch auf das Ergebnis reagieren

```
action.queue(message ->
    message.editMessage("cba").queue());
```

- Hiermit wird die Nachricht, die wir gerade gesendet haben, bearbeitet

REST ACTIONS

- Statt queue() können wir auch submit() oder complete() verwenden
- Wie die funktionieren, steht in der JDA-Dokumentation

WEITERE FEATURES

- JDA hat noch viele weitere Features
- Hier sind ein paar Beispiele:
  - Slash Commands
  - Buttons
  - Modals
  - Interaktionen
  - etc.
- Diese Features sind alle in der JDA-Dokumentation beschrieben

DER COOLE PART

RPS BOT

Live Presentation

BOT APPLICATION ERSTELLEN

Live Presentation

https://discord.com/developers/applications

CODE SCHREIBEN

Live Presentation

https://git.burg.ofahrt.com/Dennis/Discord-Bot

https://discord.gg/eNSkhE3d

ENDE

✅ Positives Feedback:

- Der Stil ist super locker und verständlich.
- Die Slides sind klar strukturiert.
- Live-Coding-Slots sind *chef's kiss*.
- Humorvolle Notizen wie „außer Nikola, der macht Racket" machen's sympathisch.