**Python-Tornado Framework: Eine Einführung**

**Was ist Tornado?**

Tornado ist ein leistungsstarkes Webframework für Python, das auf Nichtblockierung und Geschwindigkeit ausgelegt ist. Es wurde von FriendFeed entwickelt und ist besonders gut für Webanwendungen geeignet, die eine hohe Skalierbarkeit und Echtzeitfähigkeit erfordern. Mit Tornado können Entwickler robuste Webserver und Websockets implementieren.

**Warum Tornado verwenden?**

**Hohe Leistung:** Tornado ist bekannt für seine Leistungsfähigkeit und Geschwindigkeit, insbesondere bei der Verarbeitung von vielen gleichzeitigen Verbindungen.

**Nichtblockierende Architektur**: Dank seiner nichtblockierenden Architektur kann Tornado viele Verbindungen effizient verarbeiten, ohne dass dafür viele Ressourcen benötigt werden.

**Websockets-Unterstützung**: Tornado bietet eingebaute Unterstützung für Websockets, was Echtzeitkommunikation zwischen Client und Server ermöglicht.

**Skalierbarkeit**: Tornado eignet sich gut für skalierbare Anwendungen, die viele gleichzeitige Benutzer verarbeiten müssen.

Kernkonzepte von Tornado

**RequestHandler**

Ein RequestHandler ist eine Klasse, die HTTP-Anfragen verarbeitet. Du kannst benutzerdefinierte RequestHandler erstellen, um bestimmte URLs in deiner Anwendung zu bedienen.

**Python Beispiel Code:**

**import tornado.web**

**class MainHandler(tornado.web.RequestHandler):**

**def get(self):**

**self.write("Hello, World!")**

**Application**

Eine Tornado-Anwendung ist eine Instanz der tornado.web.Application-Klasse, die eine Sammlung von Routen definiert und den Webserver startet.

**Python Beispiel Code:**

import tornado.ioloop

import tornado.web

def make\_app():

return tornado.web.Application([

(r"/", MainHandler),

])

if \_\_name\_\_ == "\_\_main\_\_":

app = make\_app()

app.listen(8888)

tornado.ioloop.IOLoop.current().start()

**Asyncio-Unterstützung**

Tornado unterstützt die asynchrone Programmierung mit async und await. Dies ermöglicht es, asynchrone Operationen effizient zu verarbeiten, ohne dass die Ausführung blockiert wird.

class MainHandler(tornado.web.RequestHandler):

async def get(self):

await self.some\_async\_function()

self.write("Async operation completed!")

async def some\_async\_function(self):

pass

**Python Beispiel Code:**

**Wie man Tornado verwendet**

Installation: Installiere Tornado mit pip:

pip install tornado

# **Zusammenfassung**

Tornado ist ein leistungsstarkes Webframework für Python, das sich ideal für hochleistungsfähige Webanwendungen und Echtzeitkommunikation eignet. Mit seinen nichtblockierenden Operationen und der Websockets-Unterstützung ist Tornado eine gute Wahl für Entwickler, die nach einer schnellen und skalierbaren Lösung suchen.