# 应用场景

高并发

websocket

# 一、单文件开发流程

1、pip install tornado #安装

2、在指定目录创建文件manage.py

3、manage.py引入tornado

import tornado.web

import tornado.ioloop

import tornado.options #获取参数

1. 在main函数中编写

if \_\_name\_\_ == “\_\_main\_\_”:

tarnado.options.parse\_command\_line() #获取命令行参数

app = tornado.web.Application(

[(r“/”, IndexOne), #配置路由

(r”/one”, IndexTwo)] #配置的路由

)

app.listen(8000) #监听端口

tornado.ioloop.IOLoop,current(),start() #开启项目

1. 编写一个类，响应路由方法

class IndexOne(tronado.web.RequestHandler): #路由配置调用

def get(self) #对应Get请求

self.write(“这里是返回的数据”)

def post(self): # 对应http的post请求方式

self.write("Hello Itcast!")

# 系统类

tornado.web 基础web请求框架

|->RequestHandler 包含所有请求方法,当没有对应请求方式的成员方法

| 时，会返回“405: Method Not Allowed”错误。

|->Application 核心应用类，基础配置

|->start **单进程模式**

|->listen **单进程模式**

tornado.ioloop 核心循环模块

|->current() 当前线程的实例

|->start() 开启当前线程的实例

tornado.httpserver 服务器模块 **多进程注意端口**

|->HTTPServer( web应用 ) 实例一个应用

|->bind() 绑定端口 **多进程**

|->start( num ) num代表开启几个进程 **多进程**

tornado.options 配置项

# 自定义目录结构

目录结构自己确定

# POST与GET

tornado.web.RequestHandler包含所有的请求参数

def get(self): 、def post(self):

#接收到的信息

self.request.method #获取当前请求方法

self.request.host #请求主机名

self.request.version #使用的http版本

self.request.url #输出url地址

self.request.headers #请求头，字典

self.request.body #请求body数据

self.request.files # 上传数据

self.request.remote\_ip # 客户端的请求

#输出的信息

self.write() #返回到客户端的数据

self.set\_header(“Content-Type”,”application/json;charset=UTF-8”)

self.status(200,reson=none)

self.redirect(url) # 跳转

self.get\_argument(key, 'Hello') #获取key的值，hello必须有，可为空

-------------------------------------------------------------------

1. **GET/POST**

def get(self): / def post(self):

#获取url地址中key为greeting的值，没有为hello

self.get\_argument('greeting', 'Hello') #获取的数据

self.wirte(“sss”) #返回的数据

1. **输出错误**

self.write\_error(500)

1. ss
2. 试试