python怎么与html交互 ？python前后端交互？

Python与HTML可以通过Web框架（如Django，Flask等）进行交互。这些框架允许您在Python中编写后端逻辑，然后使用模板引擎将数据渲染到HTML页面中。此外，您可以使用Python的requests库从HTML页面中获取数据，或使用BeautifulSoup库解析HTML文档。

python后端与web前端进行交互通信解决方案：

1.Djson,Flask前后端框架，打包数据流

2.JS调用命令行cmd执行python文件， JS保存txt文件，python读取txt文件

3.PyWebIO:

[PyWebIO——免 HTML、JS 快速构建简单 Web 应用的 Python 库 - V2EX](https://staging.v2ex.com/t/746349?p=1)

[PyWebIO — PyWebIO 1.7.0 文档](https://pywebio.readthedocs.io/zh_CN/latest/)

Github： <https://github.com/wang0618/PyWebIO>

HTTP请求发起：requests, urllib,urllib3库

python多线程threading实现并行操作 join()实现线程同步

，join所完成的工作就是线程同步，即主线程任务结束之后，进入阻塞状态，一直等待其他的子线程执行结束之后，主线程在终止

join 会卡住主线程，并让当前已经 start 的子线程继续运行，直到调用.join的这个线程运行完毕。

**Q：**我想使用pysyncobj把不同端口上的sqlite数据库联系起来

**A：**pysyncobj是一个Python库，可以用于实现分布式应用程序的数据同步。它可以将多个节点上的数据同步，使得这些节点上的数据保持一致性。如果您想使用pysyncobj将不同端口上的SQLite数据库联系起来，可以按照以下步骤进行操作：

安装pysyncobj库。可以使用pip命令进行安装：

pip install pysyncobj

编写一个SQLite数据库同步的示例代码，示例代码如下：

import sqlite3

from pysyncobj import SyncObj

class SyncedSqliteDatabase(SyncObj):

def \_\_init\_\_(self, selfNodeAddr, partnerNodeAddrs):

super(SyncedSqliteDatabase, self).\_\_init\_\_(selfNodeAddr, partnerNodeAddrs)

self.conn = sqlite3.connect('mydb.db')

def set(self, key, value):

cursor = self.conn.cursor()

cursor.execute("INSERT INTO mytable (key, value) VALUES (?, ?)", (key, value))

self.conn.commit()

def get(self, key):

cursor = self.conn.cursor()

cursor.execute("SELECT value FROM mytable WHERE key = ?", (key,))

row = cursor.fetchone()

if row is not None:

return row[0]

else:

return None

def sync(self, other):

cursor = self.conn.cursor()

cursor.execute("SELECT \* FROM mytable")

rows = cursor.fetchall()

for row in rows:

key = row[0]

value = row[1]

other.set(key, value)

在这个示例代码中，我们定义了一个名为SyncedSqliteDatabase的类，它继承自SyncObj类。这个类中包含了三个方法：

set(self, key, value): 向数据库中插入一条记录，其中key为键，value为值。

get(self, key): 获取数据库中指定键对应的值。

sync(self, other): 将本地数据库的数据同步到其它节点。

在\_\_init\_\_方法中，我们使用

**Q：**继续

**A：**SQLite库连接到本地的数据库文件mydb.db。在sync方法中，我们首先获取本地数据库中的所有记录，然后通过set方法将这些记录同步到其它节点。

在主程序中启动多个SyncedSqliteDatabase实例，将它们绑定到不同的端口上：

if \_\_name\_\_ == '\_\_main\_\_':

node1 = SyncedSqliteDatabase('localhost:4321', ['localhost:4322', 'localhost:4323'])

node2 = SyncedSqliteDatabase('localhost:4322', ['localhost:4321', 'localhost:4323'])

node3 = SyncedSqliteDatabase('localhost:4323', ['localhost:4321', 'localhost:4322'])

node1.set('key1', 'value1')

node2.set('key2', 'value2')

node3.set('key3', 'value3')

print(node1.get('key2'))

print(node2.get('key3'))

print(node3.get('key1'))

在这个示例代码中，我们创建了三个SyncedSqliteDatabase实例，分别绑定到4321、4322、4323端口上，并将它们互相连接起来。然后，我们向这些数据库中插入了一些记录，并从其它节点获取了一些记录。这样，就可以实现不同端口上的SQLite数据库之间的数据同步了。

需要注意的是，这个示例代码仅仅是一个演示，实际使用时还需要考虑一些细节问题，比如节点的启动顺序、数据同步的一致性等。

**Q:**请把 刚才你说的这一部分 ‘在\_\_init\_\_方法中，我们使用’说完

**A:**在\_\_init\_\_方法中，我们使用了sqlite3库连接到本地的数据库文件mydb.db。这里的sqlite3.connect方法会创建一个sqlite3.Connection对象，该对象代表了一个数据库连接。如果指定的数据库文件不存在，sqlite3.connect方法会自动创建一个新的数据库文件。

在SyncedSqliteDatabase类中，我们只创建了一个数据库连接，这是因为SQLite是一种轻量级的数据库，它可以在同一个进程中同时访问多个数据库。如果需要在不同的进程中同时访问多个数据库，可以创建多个SyncedSqliteDatabase对象，每个对象连接到不同的数据库文件。在这种情况下，需要确保不同的进程之间可以互相访问对方的数据库文件。

sqlite3库的cursor.execute()使用方法

cursor.execute()方法是在SQLite中执行SQL语句的主要方法。它的使用方法如下：

cursor.execute(sql, [parameters])

其中，sql是要执行的SQL语句，可以是任何合法的SQL语句，包括SELECT、INSERT、UPDATE和DELETE等操作。parameters是可选的参数，用于传递SQL语句中的参数。如果SQL语句没有参数，可以省略这个参数。

cursor.execute()方法返回一个cursor对象，该对象可以用于获取SQL语句的执行结果。对于SELECT语句，可以使用cursor.fetchall()方法获取所有结果；对于INSERT、UPDATE和DELETE语句，可以使用cursor.rowcount属性获取受影响的行数。

下面是一个例子，展示了如何使用cursor.execute()方法执行一个SELECT语句：

import sqlite3

conn = sqlite3.connect('mydb.db')

cursor = conn.cursor()

cursor.execute("SELECT \* FROM mytable")

rows = cursor.fetchall()

for row in rows:

print(row)

conn.close()

在这个例子中，我们首先使用sqlite3.connect()方法连接到本地的数据库文件mydb.db，然后使用conn.cursor()方法创建一个cursor对象。接着，我们使用cursor.execute()方法执行一个SELECT语句，获取mytable表中的所有记录。最后，我们使用cursor.fetchall()方法获取所有结果，并遍历输出每一条记录的内容。最后，我们调用conn.close()方法关闭数据库连接。