

Маршрутизация

Добавление маршрутов

Основной файл:

```
from fastapi import FastAPI
from todo import todo_router

app = FastAPI()

@app.get("/")
async def welcome() -> dict:
    return {
        "message": "Hello World"
    }

app.include_router(todo_router)
```

Файл дополнительных маршрутов

```
from fastapi import APIRouter

todo_router = APIRouter()

todo_list = []

@todo_router.post("/todo")
async def add_todo(todo: dict) -> dict:
    todo_list.append(todo)
    return {"message": "Todo added successfully"}

@todo_router.get("/todo")
async def retrieve_todos() -> dict:
    return {"todos": todo_list}
```

Автоматическое добавление маршрутов в основной файл *app* из файлов в директории *data/plugins* имеющих шаблон имени объекта *APIRouter* *modulename_router*

```

fpath = os.path.join('data', 'plugins')
flist = os.listdir(fpath)
sys.path.insert(0, fpath)
for fname in flist:
    if fname not in ['__pycache__', '__init__.py']:
        m = os.path.splitext(fname)[0]
        impmod = importlib.import_module(m)
        router_name = f'{m}_router'
        if router_name in dir(impmod):
            router_mod = getattr(impmod, router_name)
            app.include_router(router_mod)

```

Получаемые параметры

Параметры пути:

```

from fastapi import Path

@todo_router.get("/todo/{todo_id}")
async def get_single_todo(todo_id: int = Path(..., title="The ID of the todo to retrieve.")) -> dict:
    for todo in todo_list:
        if todo.id == todo_id:
            return {"todo": todo}
    return {
        "message": "Todo with supplied ID doesn't exist."
    }

```

В Path ... - параметр пути обязательный, None - не обязательный

Параметры запроса (после ? в запросе):

```

async def query_route(query: str = Query(None)):
    return query

```

Передаваемые параметры

При помощи response_model можно фильтровать отдаваемые данные. Т е можно в отдаваемой модели указать неполный набор.

```

from typing import List

class TodoItem(BaseModel):

```

```
item: str
```

```
class TodoItems(BaseModel):  
    todos: List[TodoItem]
```

```
@todo_router.get("/todo", response_model=TodoItems)  
async def retrieve_todo() -> dict:  
    return {  
        "todos": todo_list  
    }
```

У содержащихся в списке словарей будет оставлен только item

Исключения

Класс HTTPException принимает три аргумента:

- status_code: Код состояния, который будет возвращен для этого сбоя
- detail: Сопроводительное сообщение для отправки клиенту
- headers: Необязательный параметр для ответов, требующих заголовков

```
from fastapi import APIRouter, Path, HTTPException, status  
  
@todo_router.get("/todo/{todo_id}")  
async def get_single_todo(todo_id: int = Path(..., title="The ID of the todo to retrieve. ")) -> dict:  
    for todo in todo_list:  
        if todo.id == todo_id:  
            return { "todo": todo }  
    raise HTTPException(  
        status_code=status.HTTP_404_NOT_FOUND,  
        detail="Todo with supplied ID doesn't exist",  
    )
```

Можно переопределить код успешного возврата в декораторе

```
@todo_router.post("/todo", status_code=201)  
async def add_todo(todo: Todo) -> dict:  
    todo_list.append(todo)  
    return { "message": "Todo added successfully." }
```

Revision #6

Created 22 July 2024 13:14:42 by Admin

Updated 28 July 2024 13:14:30 by Admin