

AI 掘金头条-新闻模块

项目工程结构、新闻分类、新闻列表、新闻详情



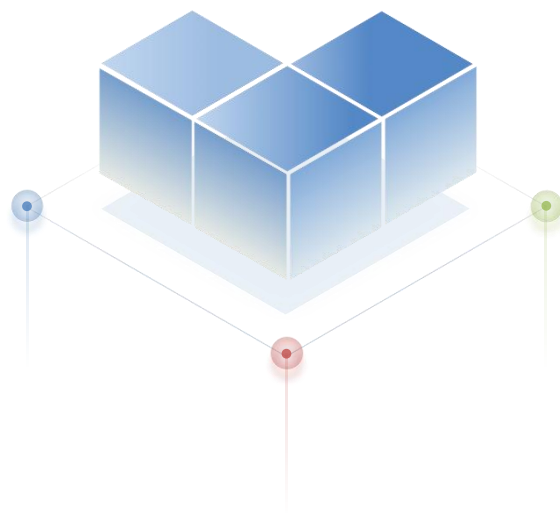
黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

项目一：AI 掘金头条

项目功能

新闻模块
用户模块
收藏和浏览历史
...



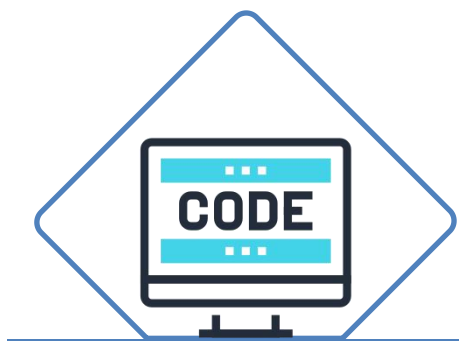
配套物料

前端代码
接口规范文档
数据库 SQL 文件

设计说明文档

用来系统说明整个后端系统的架构、模块划分、数据设计和接口规范

配套物料



前端项目代码

项目前端工程采用 Vue 框架，实现了页面结构、组件交互和接口调用等核心逻辑，能够在浏览器端呈现完整界面并与后端服务对接，从而展示整个应用的完整功能。



接口规范文档

项目的接口规范文档，定义了各接口的请求方式、参数格式、响应结构和错误码等关键规则，为前后端提供统一的交互标准，确保系统各模块能够准确对接并稳定协同工作。



数据库 SQL 文件

项目的 MySQL SQL 文件，包含了数据库的表结构定义、初始化数据和必要的约束规则，用于在数据库中快速搭建完整的数据存储环境，为后端服务提供稳定可靠的数据基础。

运行前端项目

01

安装 Node

运行构建工具 (Vite/Webpack) 、
开启开发服务器、编译代码

02

运行项目

1. 项目根目录打开终端
2. `npm run dev`

工程结构



模块化路由

```
toutiao_backend/  
├── crud/  
│   ├── news.py  
│   └── users.py  
├── models/  
│   ├── news.py  
│   └── users.py  
├── routers/ ←----- 路由层（按模块划分）  
│   ├── news.py  
│   └── users.py  
├── schemas/  
│   ├── news.py  
│   └── users.py  
├── utils/  
├── config/  
│   └── db_conf.py  
├── main.py  
└── test_main.http
```

模块化路由

```
@app.get("/users")
def get_users(): ...

@app.post("/users")
def create_user(): ...

@app.get("/news_categories")
def get_categories(): ...

@app.get("/news_list")
def get_news_list(): ...

@app.get("/news_detail")
def get_news_detail(): ...
```

```
@app.get("/token")
def get_token(): ...

@app.get("/history_list")
def get_list(): ...

@app.get("/favorite_list")
def get_list(): ...

@app.get("/user_list")
def get_user_list(): ...

@app.put("/update_user")
def update_user(): ...
```

吃个鸡腿压压惊



模块化路由

模块化路由就是把每个业务功能的接口拆分到独立文件里，再统一挂载到主应用中。



接口按模块拆分，不会混在一起，让整个项目结构更直观

项目结构更清晰



每个模块都负责自己对应的接口，便于快速查找和定向修改

更易维护



把接口拆分出去后，main.py 就只负责启动应用，不再堆满业务代码

避免 main.py 爆炸

模块化路由

```
app/  
└─ routers/  
    ├── news.py  
    └─ users.py
```

模块化目录结构

```
app.include_router(news.router)
```

在 main.py 中挂载路由



编写独立路由模块

```
from fastapi import APIRouter  
  
# 创建 APIRouter 实例  
router = APIRouter(prefix="/api/news", tags=["news"])  
  
@router.get("/categories")  
async def get_categories():  
    return {"msg": "获取分类成功"}
```

◆ 什么是模块化路由？有什么优势？

模块化路由就是把每个业务功能的接口拆分到独立文件里，再统一挂载到主应用中。

优势：项目结构更清晰、项目更易维护

◆ 说出下方代码的含义

```
from fastapi import APIRouter

# 创建 APIRouter 实例
router = APIRouter(prefix="/api/news",tags=["news"])

@router.get("/categories")
async def get_categories():
    return {"msg": "获取分类成功"}
```

◆ 如何注册路由？

在 `main.py` 中添加 `app.include_router()`

```
app.include_router(news.router)
```



```
from fastapi import APIRouter

router = APIRouter(prefix="/api/news",tags=["news"])

@router.get("/xx")
async def xx_xx():
```

1. 数据库操作: 增、删、改、查
2. 响应结果

建库



database.sql

配置 ORM



操作数据



配置 ORM

- sqlalchemy[asyncio]
- aiomysql (异步数据库驱动)

安装

1. `create_async_engine` 创建异步引擎
2. 定义模型基类 (继承 `DeclarativeBase`) 和 模型类
3. `run_sync(Base.metadata.create_all)` 建表

建表

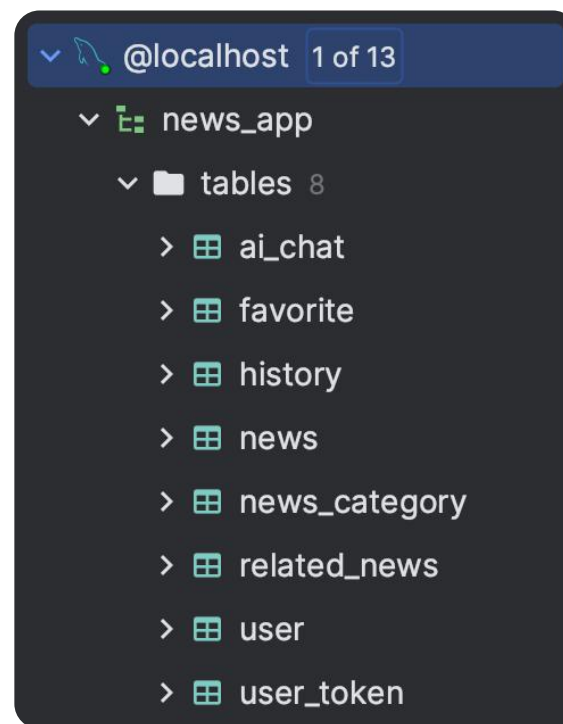
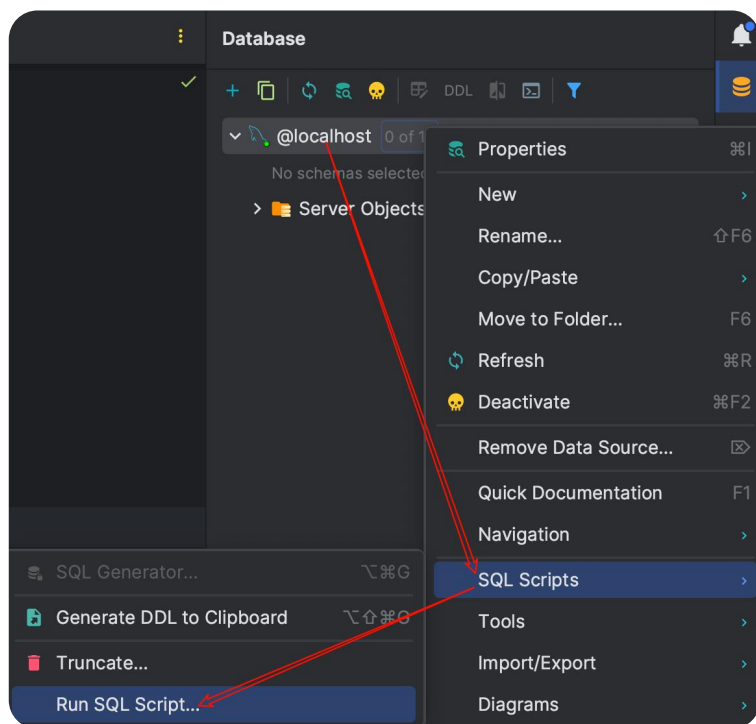
1. 路由处理函数注入数据库会话依赖项 `Depends`
2. 查询数据: `select()`
3. 增加数据: `add()`
4. 更新数据: 重新赋值
5. 删除数据: `delete()`

操作
数据

配置 ORM

MySQL 导入 SQL 文件:

PyCharm 中 Database 插件 → 数据库连接 → 右键 → SQL Script → Run SQL Script → 浏览 SQL 文件 → 确认



ORM 配置

```
from sqlalchemy.ext.asyncio import async_sessionmaker, AsyncSession, create_async_engine
# 数据库URL
ASYNC_DATABASE_URL = "mysql+aiomysql://root:123456@localhost:3306/news_app?charset=utf8mb4"
# 创建异步引擎
async_engine = create_async_engine(ASYNC_DATABASE_URL, echo=True, pool_size=10, max_overflow=20)
AsyncSessionLocal = async_sessionmaker(
    bind=async_engine,
    class_=AsyncSession,
    expire_on_commit=False
)
# 依赖项，用于获取数据库会话
async def get_db():
    async with AsyncSessionLocal() as session:
        try:
            yield session
            await session.commit()
        except Exception:
            await session.rollback()
            raise
        finally:
            await session.close()
```

新闻模块核心功能

新闻分类

新闻列表



新闻详情
相关推荐

获取新闻分类



接口实现流程



- ✓ 定义 **APIRouter** 实例
- ✓ 注册路由
- ✓ **include_router()**
- ✓ 参照接口规范文档



```
class Base(DeclarativeBase):  
    pass  
  
class Category(Base):  
    __tablename__ = "news_category"  
    .....
```

- ✓ 参照数据库表



- ✓ **select()** (模型类)
- ✓ **add()**
- ✓ **update()**
- ✓ **delete()**



- ✓ **Depends** 注入数据库依赖
- ✓ 调用逻辑
- ✓ 响应结果

获取新闻分类列表

1. 获取新闻分类列表

- 接口地址: GET /api/news/categories
- 请求参数:

参数名	类型	必填	说明
skip	integer	否	跳过的记录数，默认为0
limit	integer	否	返回的记录数限制，默认为100

- 请求示例:

```
GET /api/news/categories
GET /api/news/categories?skip=0&limit=10
```

- 响应示例:

```
{
  "code": 200,
  "message": "success",
  "data": [
    {
      "id": 1,
      "created_at": "2222-02-02T00:00:00",
      "updated_at": "2222-02-02T00:00:00",
      "name": "科技",
      "sort_order": 0
    }
  ]
}
```

```
router = APIRouter(prefix="/api/news", tags=["news"])

@router.get("/categories")
async def get_categories(skip: int=0, limit: int=100):
    return {
        "code": 200,
        "message": "success",
        "data": "新闻分类列表"
    }
```

接口实现流程



- ✓ 定义 **APIRouter** 实例
- ✓ 注册路由
`include_router()`
- ✓ 参照接口规范文档



```
class Base(DeclarativeBase):  
    pass  
  
class Category(Base):  
    __tablename__ = "news_category"  
    .....
```

- ✓ 参照数据库表



- ✓ `select(模型类)`
- ✓ `add()`
- ✓ `update()`
- ✓ `delete()`

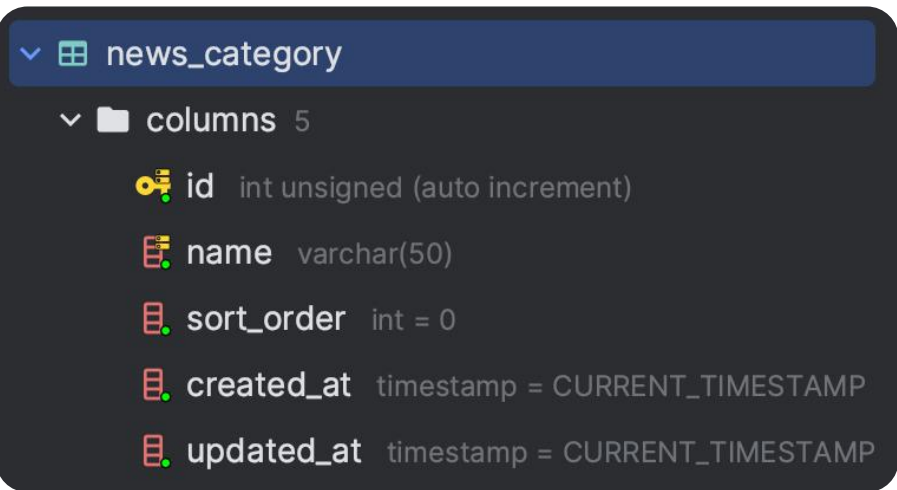


- ✓ **Depends** 注入数据库依赖
- ✓ 调用逻辑
- ✓ 响应结果

获取新闻分类列表 – 定义模型类

模型类规范：

- 基类，继承 `DeclarativeBase`
- 数据库表模型类，继承基类
 - ★ 属性及类型参照数据库表定义



跨域资源共享 CORS

跨域资源共享（CORS）是一种浏览器安全机制，用于允许运行在一个源（Origin）的 Web 应用，通过浏览器向另一个源的服务器发起跨域 HTTP 请求，并在服务器授权的前提下获取资源。

同源的三个条件：

- 协议
- 域名
- 端口

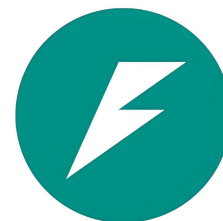
前端	后端	是否跨域
http://localhost:5173 ↗	http://localhost:8000 ↗	✅ 跨域
http://example.com ↗	http://api.example.com ↗	✅ 跨域
http://example.com ↗	http://example.com ↗	❌ 不跨域

跨域资源共享 CORS



前端 Vue

`http://localhost:5173/`



后端 FastAPI

`http://127.0.0.1:8000/`

CORS 中间件

CORS: 让后端主动告诉浏览器: 这个前端“允许访问”。

```
from fastapi.middleware.cors import CORSMiddleware

# 允许的来源 (可以是域名列表)
origins = [
    "http://localhost",
    "http://localhost:3000",
    "https://your-frontend-domain.com"
]

# 添加 CORS 中间件
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],           # 允许访问的源
    allow_credentials=True,       # 允许携带 Cookie
    allow_methods=["*"],          # 允许所有请求方法
    allow_headers=["*"],          # 允许所有请求头
)
```

◆ 分析下图浏览器报错的原因?

✖ Access to XMLHttpRequest at '<http://127.0.0.1:8000/api/news>' from origin '<http://localhost:5173>' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. [home:1](#)

浏览器向另一个源的服务器发起了跨域 HTTP 请求（浏览器的安全机制，只允许同源请求）

◆ 同源的三个条件是什么?

- 协议
- 域名
- 端口

◆ 如何解决跨域问题?

全局配置 CORS 中间件

```
from fastapi.middleware.cors import
CORSMiddleware

# 允许的来源 (可以是域名列表)
origins = [
    "http://localhost",
    "http://localhost:3000",
    "https://your-frontend-domain.com"
]

# 添加 CORS 中间件
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],      # 允许访问的源
    allow_credentials=True,  # 允许携带 Cookie
    allow_methods=["*"],     # 允许所有请求方法
    allow_headers=["*"],     # 允许所有请求头
)
```



获取新闻列表

2. 获取新闻列表

- 接口地址: GET /api/news/list

- 请求参数:

参数名	类型	必填	说明
categoryId	integer	是	分类ID
page	integer	否	页码, 默认为1
pageSize	integer	否	每页显示的新闻数量, 最大值为100, 默认为10

- 请求示例:

```
GET /api/news/list?categoryId=1
GET /api/news/list?categoryId=1&page=2&pageSize=20
```

- 响应示例:

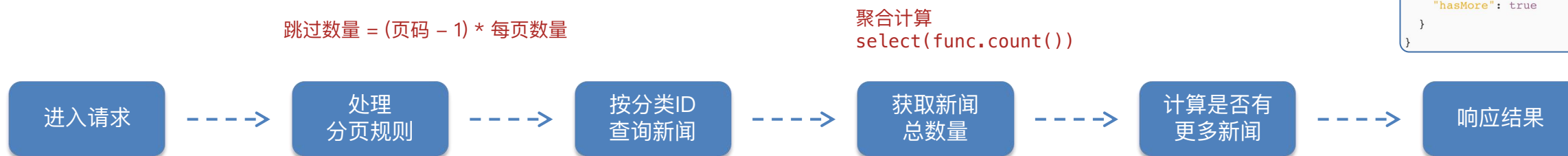
```
{
  "code": 200,
  "message": "success",
  "data": {
    "list": [
      {
        "id": 1,
        "category": null,
        "title": "新闻标题",
        "description": "新闻简介",
        "content": "新闻内容",
        .....
      }
    ],
    "total": 100,
    "hasMore": true
  }
}
```

```
@router.get("/list")
async def get_news(
    category_id: int = Query(..., alias="categoryId"),
    page: int = 1,
    page_size: int = Query(10, le=100, alias="pageSize")
):
    return {
        "code": 200,
        "message": "success",
        "data": {
            "list": "news_list",
            "total": "total_count",
            "hasMore": "has_more"
        }
    }
```

获取新闻列表 – 查询功能

查询功能响应结果：当前分类新闻列表、新闻总量、是否还有更多新闻

```
{
  "code": 200,
  "message": "success",
  "data": {
    "list": [
      {
        "id": 1,
        "category": null,
        "title": "新闻标题",
        "description": "新闻简介",
        "content": "新闻内容",
        .....
      }
    ],
    "total": 100,
    "hasMore": true
  }
}
```



参数

- 分类 id: categoryId
- 页码: page
- 每页数量: pageSize

分页查询

select().offset(跳过数量).limit(pageSize)

是否还有更多新闻

(offset + 当前新闻列表条数) < 总量

获取新闻详情

新闻分类

新闻列表

新闻资讯

头条 社会 国内 国际 娱乐 体育 科技 更多>

国家主席发表2024年新年贺词

主席总结过去一年成就，展望未来发展蓝图，强调高质量发展。

新华社 12517 阅读

全国两会将于3月初在京召开

十四届全国人大二次会议和全国政协十四届二次会议议程公布。

人民日报 8910 阅读

我国成功发射通信技术试验卫星十一号

卫星顺利进入预定轨道，将用于开展多频段、高速率通信试验。

央视新闻 11230 阅读

2023年我国GDP同比增长5.2%

经济总量超126万亿元，国民经济回升向好，高质量发展扎实推进。

经济日报 15300 阅读

央行宣布降准0.5个百分点

释放长期资金约1万亿元，保持流动性合理充裕，支持实体经济。

财经网 14200 阅读

C919国产大飞机开启东南亚演示飞行

首次飞抵新加坡航展，展示我国民用航空工业新成就。

新华每日电讯 9800 阅读

首页 AI问答 我的

新闻详情

我国成功发射通信技术试验卫星十一号

央视新闻 2024-01-10T18:20:00 11231 阅读



2024年1月10日15时03分，我国在西昌卫星发射中心使用长征二号丁运载火箭，成功将通信技术试验卫星十一号发射升空。该卫星主要用于开展Ka频段宽带通信、新型天线展开等关键技术验证，可为下一代通信技术积累数据。此次任务是长征系列火箭的第456次飞行。

相关推荐

2023年我国GDP同比增长5.2%

新闻详情
相关推荐

获取新闻详情

3. 获取新闻详情

- 接口地址: GET /api/news/detail
- 请求参数:

参数名	类型	必填	说明
id	integer	是	新闻ID

- 请求示例:

```
GET /api/news/detail?id=1
```

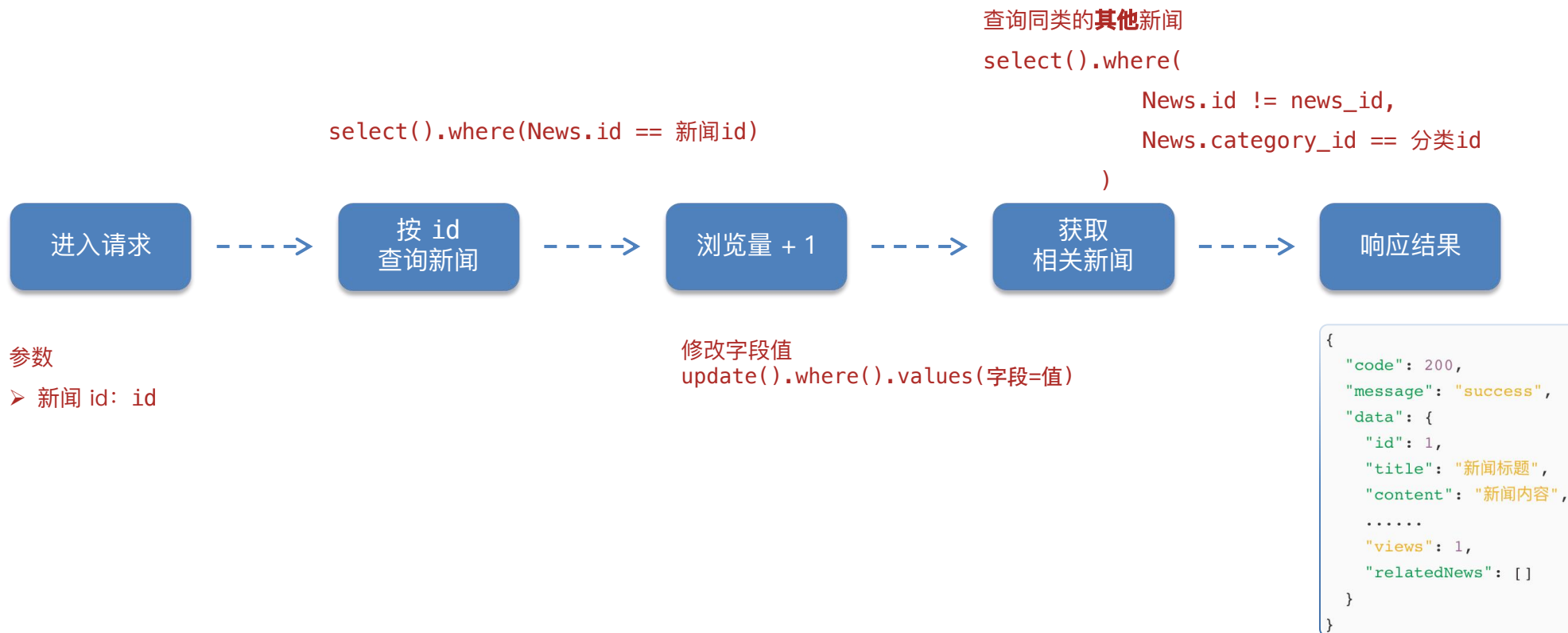
- 响应示例:

```
{
  "code": 200,
  "message": "success",
  "data": {
    "id": 1,
    "title": "新闻标题",
    "content": "新闻内容",
    .....
    "views": 1,
    "relatedNews": []
  }
}
```

```
@router.get("/detail")
async def read_news_detail(news_id: int=Query(..., alias="id")):
    return {
        "code": 200,
        "message": "success",
        "data": {
            "id": "新闻id",
            "title": "新闻标题",
            "content": "",
            .....
            "relatedNews": "相关新闻"
        }
    }
```

获取新闻详情

响应结果：当前新闻详情 + 增加1次浏览量 + 相关新闻（同分类 id 的新闻）



新闻模块

新闻分类

新闻列表

新闻资讯

头条 社会 国内 国际 娱乐 体育 科技 更多>

国家主席发表2024年新年贺词

主席总结过去一年成就，展望未来发展蓝图，强调高质量发展。

新华社 12517 阅读

全国两会将于3月初在京召开

十四届全国人大二次会议和全国政协十四届二次会议议程公布。

人民日报 8910 阅读

我国成功发射通信技术试验卫星十一号

卫星顺利进入预定轨道，将用于开展多频段、高速率通信试验。

央视新闻 11230 阅读

2023年我国GDP同比增长5.2%

经济总量超126万亿元，国民经济回升向好，高质量发展扎实推进。

经济日报 15300 阅读

央行宣布降准0.5个百分点

释放长期资金约1万亿元，保持流动性合理充裕，支持实体经济。

财经网 14200 阅读

C919国产大飞机开启东南亚演示飞行

首次飞抵新加坡航展，展示我国民用航空工业新成就。

新华每日电讯 9800 阅读

首页 AI问答 我的

新闻详情
相关推荐

< 返回 新闻详情

我国成功发射通信技术试验卫星十一号 ☆

央视新闻 2024-01-10T18:20:00 11231 阅读



2024年1月10日15时03分，我国在西昌卫星发射中心使用长征二号丁运载火箭，成功将通信技术试验卫星十一号发射升空。该卫星主要用于开展Ka频段宽带通信、新型天线展开等关键技术验证，可为下一代通信技术积累数据。此次任务是长征系列火箭的第456次飞行。

相关推荐

 2023年我国GDP同比增长5.2%

本章总结

这都是知识啊



本章总结



本章总结



- ✓ 定义 **APIRouter** 实例
- ✓ 注册路由
- ✓ **include_router()**
- ✓ 参照接口规范文档



```
class Base(DeclarativeBase):  
    pass  
  
class Category(Base):  
    __tablename__ = "news_category"  
    .....
```

- ✓ 参照数据库表



- ✓ **select()** (模型类)
- ✓ **add()**
- ✓ **update()**
- ✓ **delete()**



- ✓ **Depends** 注入数据库依赖
- ✓ 调用查询逻辑
- ✓ 响应结果

本章总结

- 模块化路由

```
from fastapi import APIRouter

# 创建 APIRouter 实例
router = APIRouter(prefix="/api/news",tags=["news"])

@router.get("/categories")
async def get_categories():
    return {"msg": "获取分类成功"}

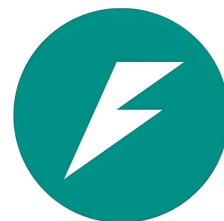
# main.py 注册路由
app.include_router(news.router)
```

本章总结



前端 Vue

`http://localhost:5173/`



后端 FastAPI

`http://127.0.0.1:8000/`

本章总结

- 跨域资源共享（CORS）是一种浏览器安全机制，用于允许运行在一个源（Origin）的 Web 应用，通过浏览器向另一个源的服务器发起跨域 HTTP 请求，并在服务器授权的前提下获取资源。

- 同源的三个条件：

- 协议
- 域名
- 端口

```
from fastapi.middleware.cors import CORSMiddleware

# 允许的来源（可以是域名列表）
origins = [
    "http://localhost",
    "http://localhost:3000",
    "https://your-frontend-domain.com"
]

# 添加 CORS 中间件
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],           # 允许访问的源
    allow_credentials=True,       # 允许携带 Cookie
    allow_methods=["*"],          # 允许所有请求方法
    allow_headers=["*"],          # 允许所有请求头
)
```

本章总结

- 模型类中，如何为字段创建索引？

```
class News(Base):  
    .....  
    # 创建索引：提升查询速度  
    __table_args__ = (  
        Index('fk_news_category_idx', 'category_id'),  
        Index('idx_publish_time', 'publish_time')  
    )
```

- ORM 如何将查询的数据排序？
 - `select().order_by(News.views.desc())`
 - 默认升序，`desc()` 表示降序

本章总结

- 使用 ORM 修改数据的方法是什么?

➤ `update().where().values()`

```
async def increase_news_views(db: AsyncSession, news_id: int):  
    stmt = update(News).where(News.id == news_id).values(views=News.views + 1)  
    result = await db.execute(stmt)  
    await db.commit()  
    # 更新 → 检查数据库是否真的命中了数据 → 命中了返回True  
    return result.rowcount > 0
```



传智教育旗下高端IT教育品牌

