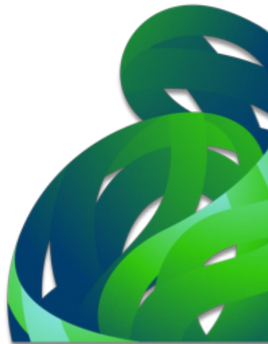
小微团队的业务中台解决方案

陈晓耀（青铜）

之江实验室 • 天枢人工智能开源平台
2020-08-20



总体流程

01

需求背景

问题描述：小微团队定义 & 面临业务挑战

02

研发流程

流程优化：研发流程拆解 & 产品需求定义

03

实现方案

解决方案：基于 strapi 的业务中台方案

04

QA

持续优化：业务落地 & 问题答疑

小微团队定义

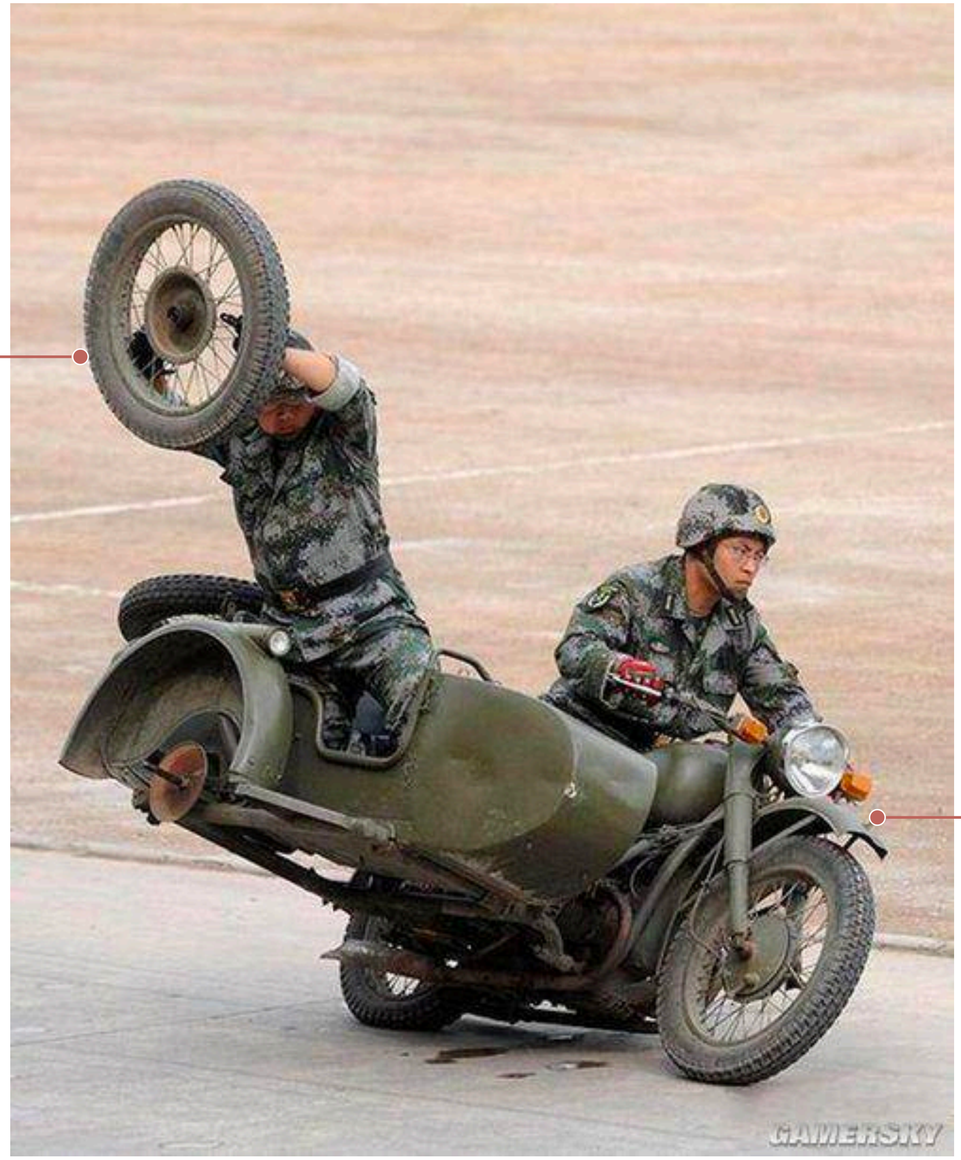
- 团队规模较小，整体水平较弱
- 基础设施薄弱，项目流程混乱
- 缺少技术沉淀，项目从头开始

面临业务痛点

- 新需求层出不穷，老代码难以复用
- 历史债务越滚越多，难以持续维护
- 新功能需要快速交付，可行性验证

现实情况是...

落后的技术框架升级



快速迭代的业务

开发后台服务都需要经历的流程

1

数据库设计

2

编写 Model 层
封装 Service
设计 API
定义 Router

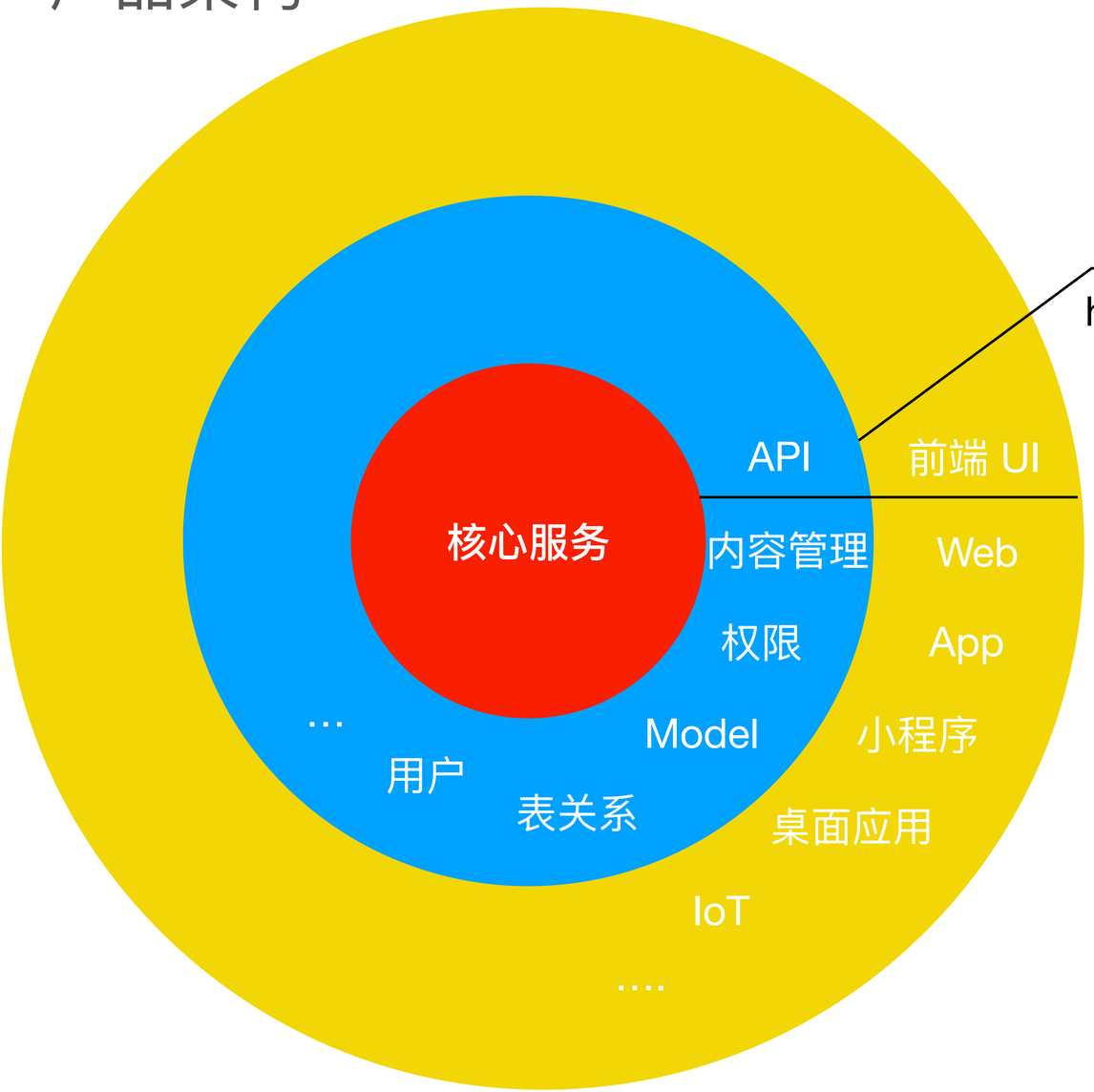
3

权限控制
用户注册、登录

4

前后端联调
产品运营
持续迭代

产品架构



业务中台

headless-CMS

- 数据库设计
- Model 编写
- Service 开发
- API 设计
- 前后端接口约定
- 权限设计
- 用户注册、登陆
- 内容运营
- ……

假设我是一个产品经理：

- 一套数据到 API 的自动生成系统
- 为产品运营人员提供内容管理后台
- 自带权限管理，适应大部分业务场景
- 自带用户体系，包括注册、登陆等
- 面向多终端，快速接入，减少开发
- 方便扩展，没有太多约束

strapi

快速开始

Build apps fast.

Building self-hosted, customizable, and performant content API has never been easier.

Open Source

Forever. The entire codebase is available on GitHub and maintained by hundreds of contributors.

Customizable

Easily customize the admin panel as well as the API. Extend your content management with custom plugins, in seconds.

RESTful or GraphQL

Consume the API from any client (React, Vue, Angular), mobile apps or even IoT devices, using REST or GraphQL.

Self-hosted

Don’t give up on data privacy or lock yourself in. Keep control of your data and your costs at all time.

需求背景 研发流程 实现方案 QA 环节

3min 快速上手

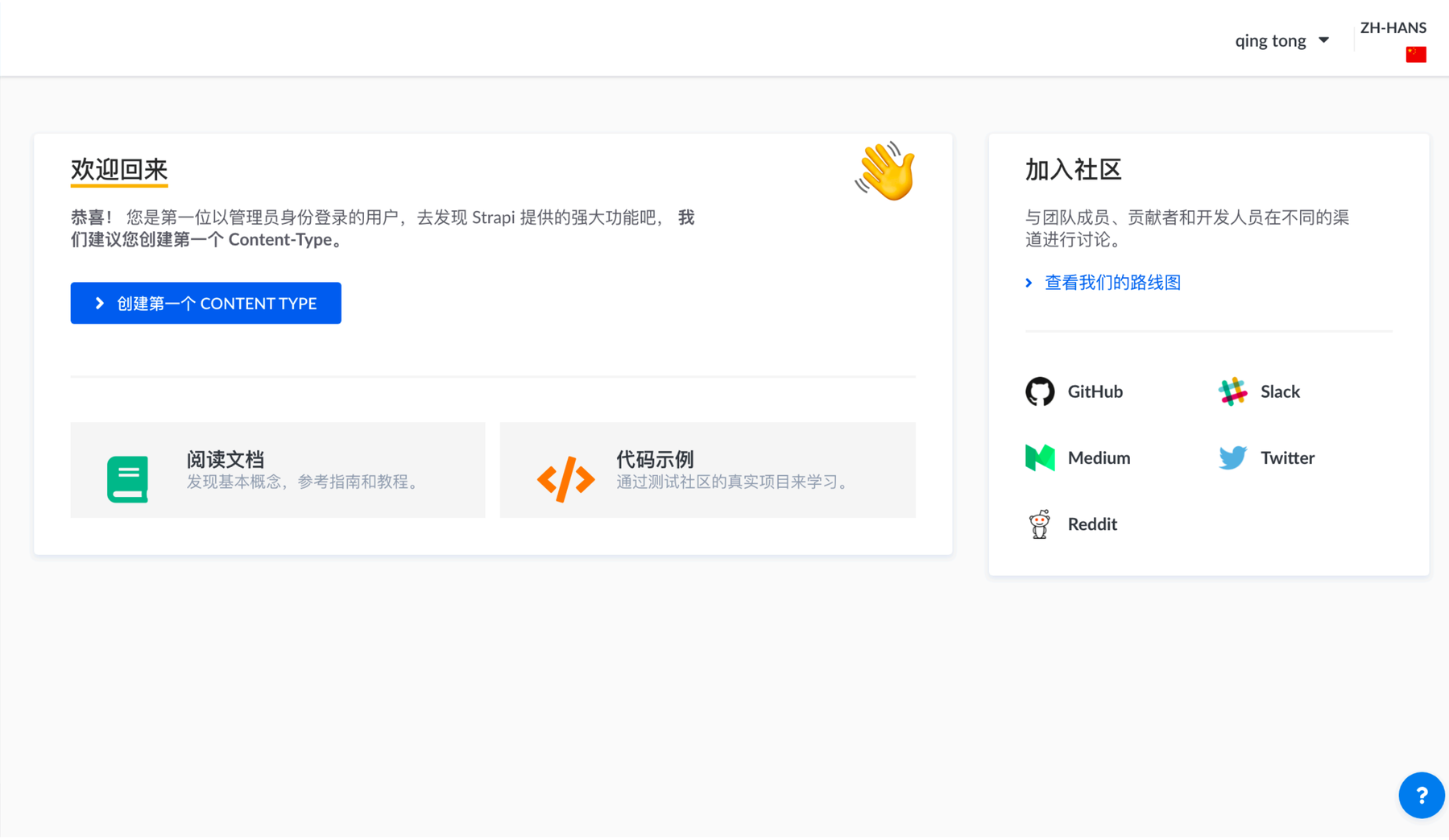
```
# 首先需要有 node.js 环境
> brew install node
> # 拉取 strapi 脚手架
> yarn create strapi-app my-project --quickstart
> # or use npx
> npx create-strapi-app my-project --quickstart
> # start
> cd my-project
> yarn develop
```

- Default database client: sqlite3
- Filename: .tmp/data.db

界面展示

数据管理

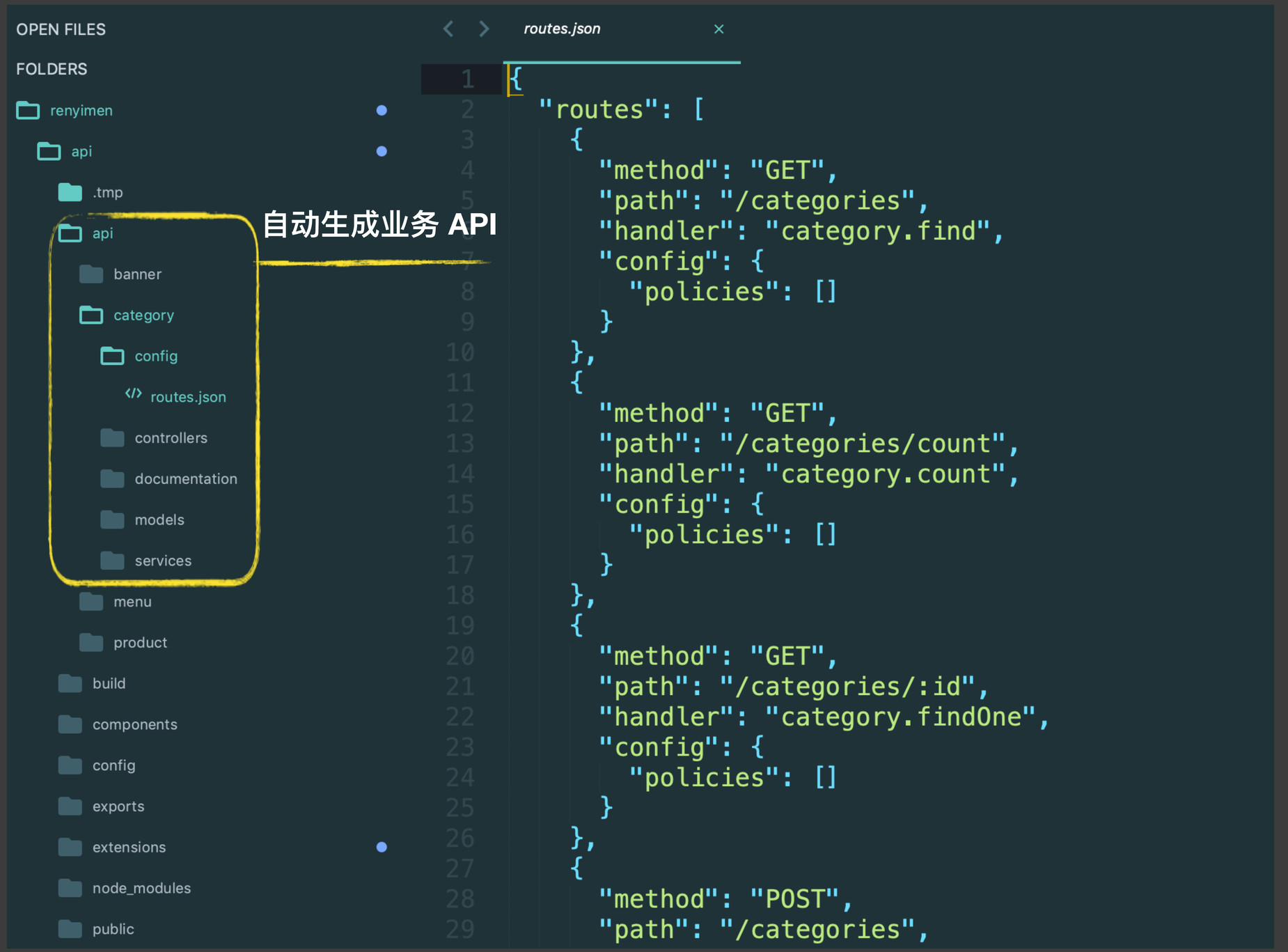
数据类型定制



目录说明


数据文件
api
.tmp
业务代码
api
build
components
配置信息
config
exports
平台扩展
extensions
public
scripts
.editorconfig
.env
.eslintignore
.eslintrc
.gitignore
favicon.ico
package.json
部署配置
process.json
README.md
服务入口
server.js

业务代码


OPEN FILES
FOLDERS
renyimen
api
.tmp
api
banner
category
config
routes.json
controllers
documentation
models
services
menu
product
build
components
config
exports
extensions
node_modules
public
自动生成业务 API
routes.json
{
"routes": [
{
"method": "GET",
"path": "/categories",
"handler": "category.find",
"config": {
"policies": []
}
},
{
"method": "GET",
"path": "/categories/count",
"handler": "category.count",
"config": {
"policies": []
}
},
{
"method": "GET",
"path": "/categories/:id",
"handler": "category.findOne",
"config": {
"policies": []
}
},
{
"method": "POST",
"path": "/categories",

RESTFul API

HTTP 请求	路径	示例	说明
GET	/{content-type}	/menus	获取 menu 全部列表
GET	/{content-type}/count	/menus/count	获取 menu 列表数量
GET	/{content-type}/:id	/menus/1	获取 id 为 1 的 todo 记录
POST	/{content-type}	/menus	添加一个 menu 记录
DELETE	/{content-type}/:id	/menus/2	删除 id 为 2 的 menu 记录
PUT	/{content-type}/:id	/menus/3	更新 id 为 3 的 menu 记录

RESTFul API — Filter

Filters

Filters are used as a suffix of a field name:

- Not suffix or `eq` : Equals
- `ne` : Not equals
- `lt` : Lower than
- `gt` : Greater than
- `lte` : Lower than or equal to
- `gte` : Greater than or equal to
- `in` : Included in an array of values
- `nin` : Isn't included in an array of values
- `contains` : Contains
- `ncontains` : Doesn't contain
- `containss` : Contains case sensitive
- `ncontainss` : Doesn't contain case sensitive
- `null` : Is null/Is not null

相等匹配（默认）

```
GET /menus?name=hello
```

相等匹配

```
GET /todos?name_eq=hello
```

大于等于

```
GET /students?score_gte=60
```

id in (3, 6, 8)

```
GET /todos?id_in=3&id_in=6&id_in=8
```

name 包含 foo 或者 name 包含 bar

```
GET /products?name_contains=foo&name_contains=bar
```

RESTFul API — sort、limit、start

按 id 正序排序（默认）

```
GET /products?_sort=id:ASC
```

按 id 倒序排序

```
GET /products?_sort=id:DESC
```

组合排序

```
GET /users?_sort=email:asc,dateField:desc
```

组合排序（大小写不敏感）

```
GET /users?_sort=email:DESC,username:ASC
```

分页

```
GET /products?_start=10&_limit=10
```

GraphQL

GraphQL 是一种基于类型系统（type）的 API 查询语言，

相比于 REST，GraphQL 是一种新的前后端交互约定。

后端定义接口类型

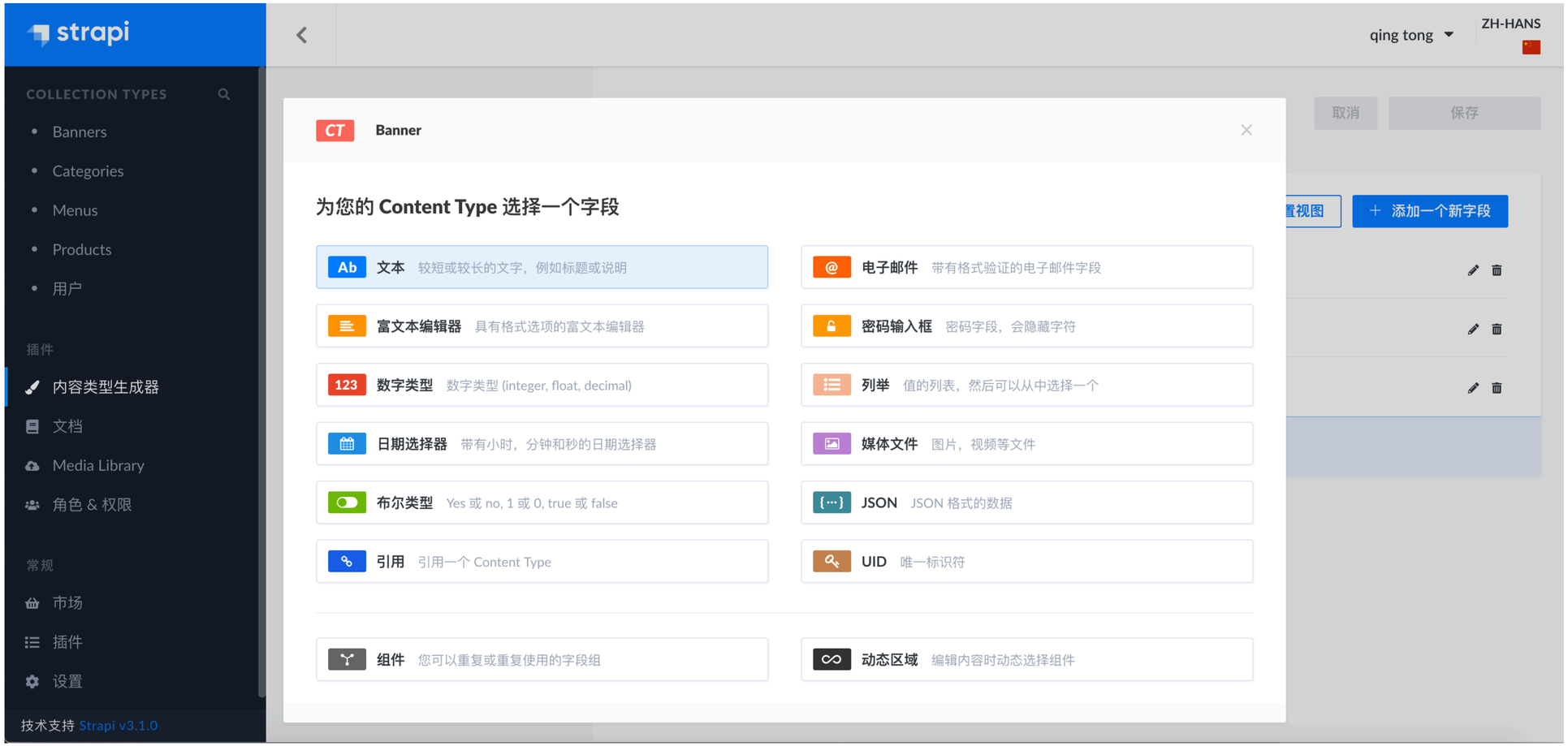
```
type Post {
  _id: String
  created_at: String
  updated_at: String
  title: String
  content: String
  published: Boolean
}

type Query {
  posts(sort: String, limit: Int, start: Int, where: JSON): [Post]
  post(id: String!): Post
}
```

前端决定返回内容

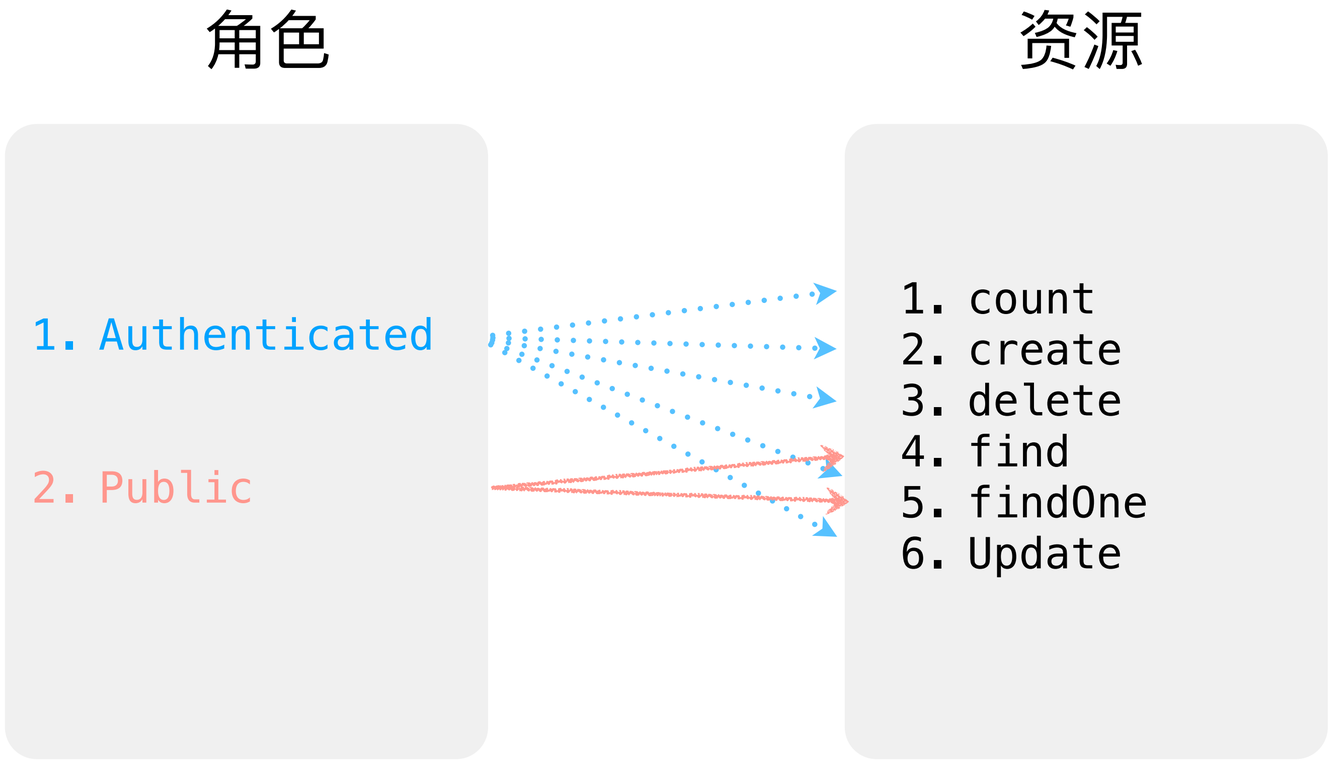
```
query {
  posts(limit: 10, start: 10, sort: "title:asc", where: {
    title_contains: "foo"
  }) {
    _id,
    title,
    content,
    published
  }
}
```

内容类型

- String
- Number
- Boolean
- Rich Text
- Date
- Password
- Media
- Enum
- Email
- JSON
- **Relation**
- **Component**
- **Dynamic Zone**

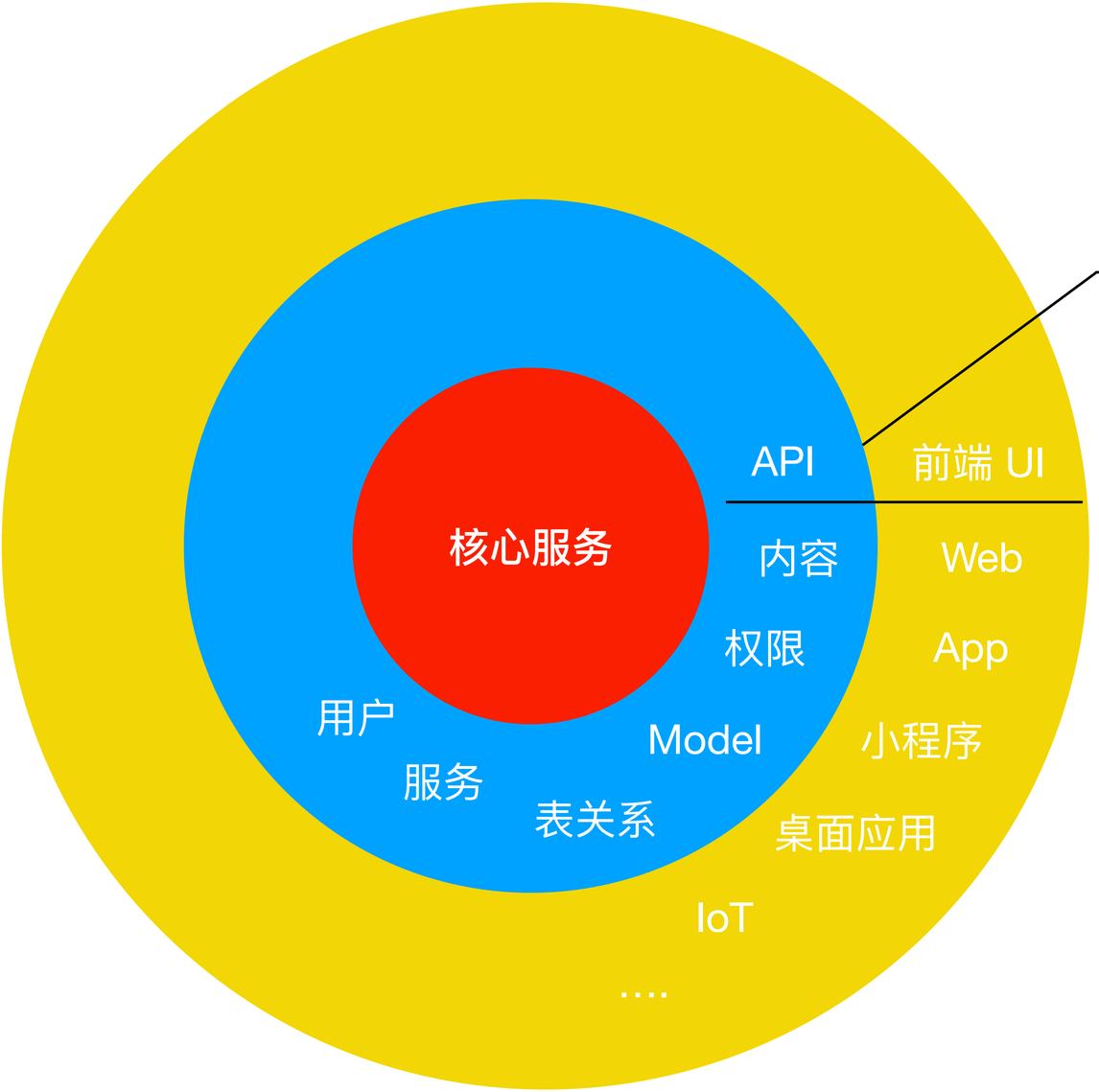


角色 & 资源 & 权限



Authenticated: 已认证系统用户
Public: 普通游客

资源对应每个内容类型对应 API



Strapi

- 数据库设计
- Model 编写
- Service 开发
- API 设计
- 前后端路由设计
- 权限设计
- 用户注册、登陆
- 内容运营
- ……

可能的风险

- 只适用于简单的表查询，无法处理复杂条件联动
- 数据同步机制（数据迁移、导入导出等）
- 社区插件不够丰富

QA 环节