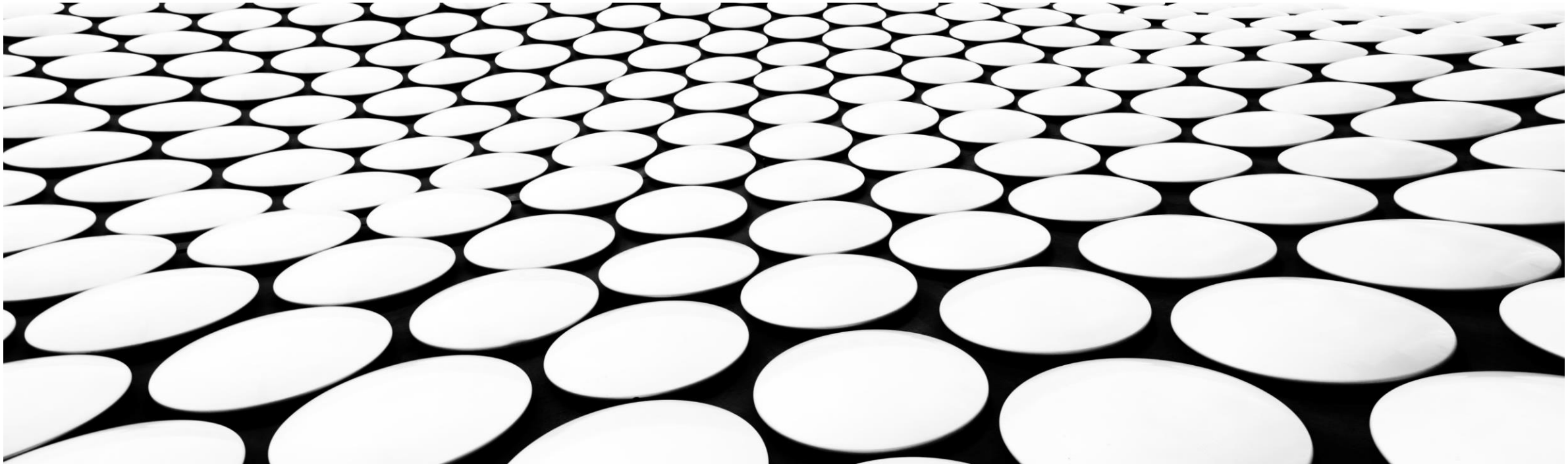


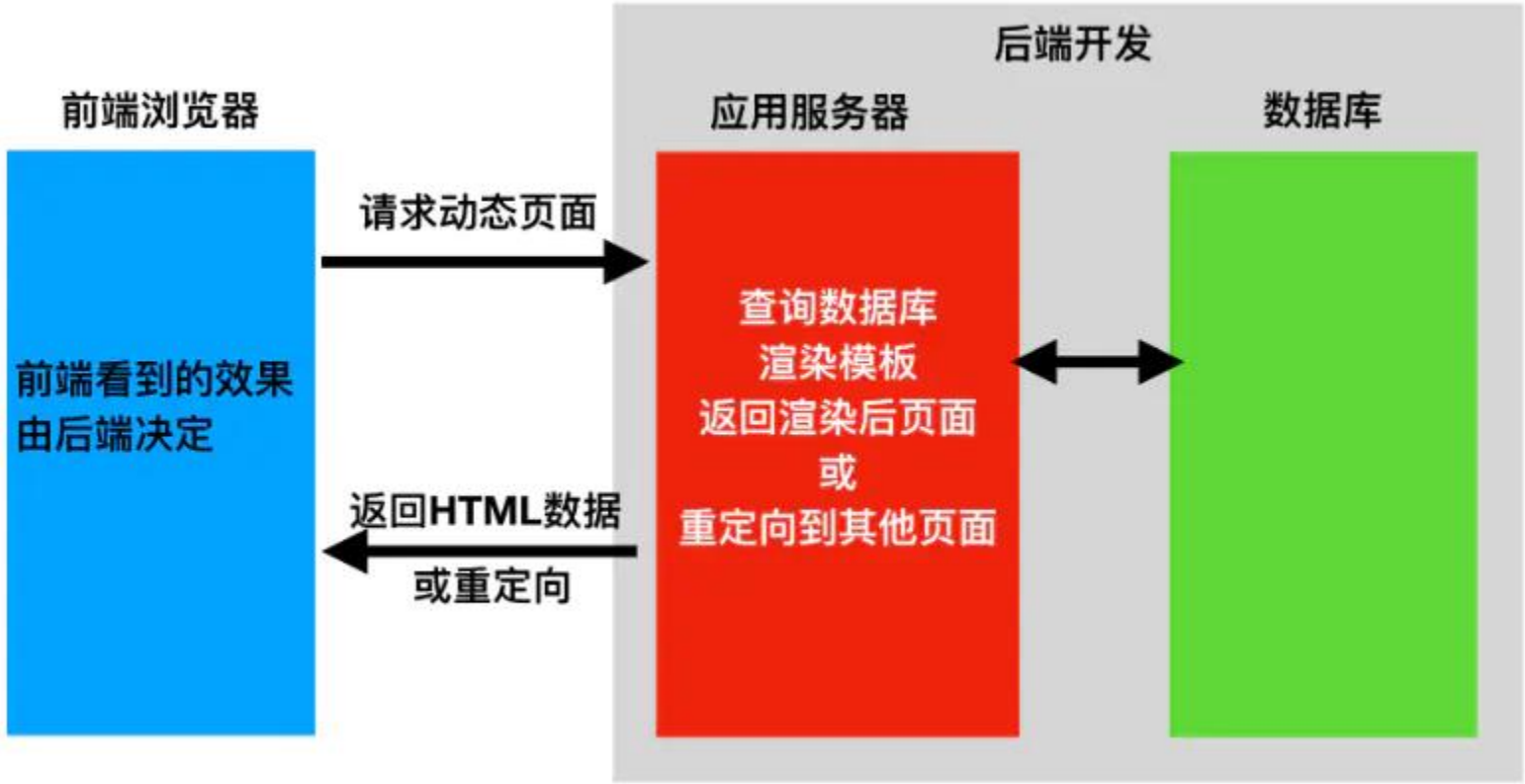
服务端开发-REST API开发

陶召胜

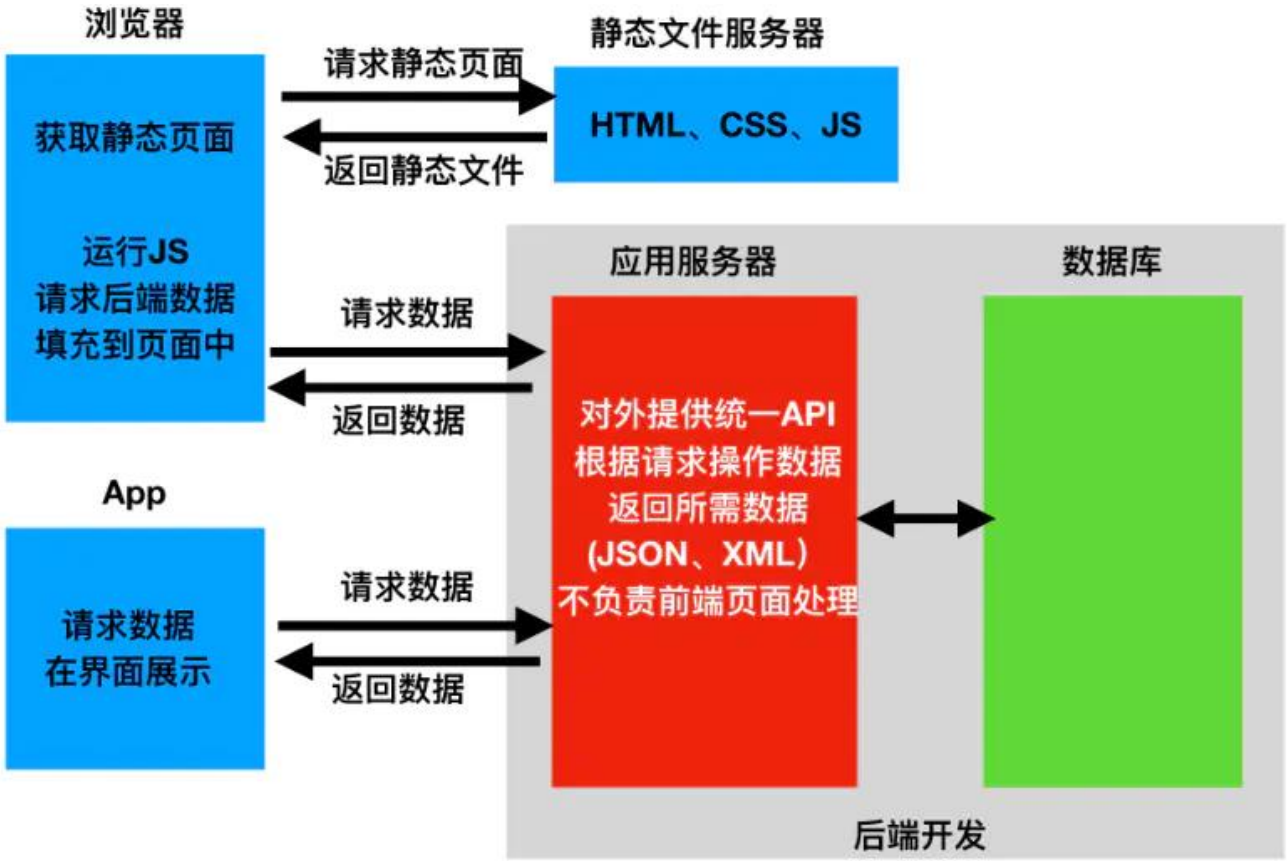
内容

1. 使用Spring MVC的控制器创建RESTful端点
2. 将Spring Data存储库暴露为REST端点
3. 测试和保护端点

前、后端不分离的开发模式



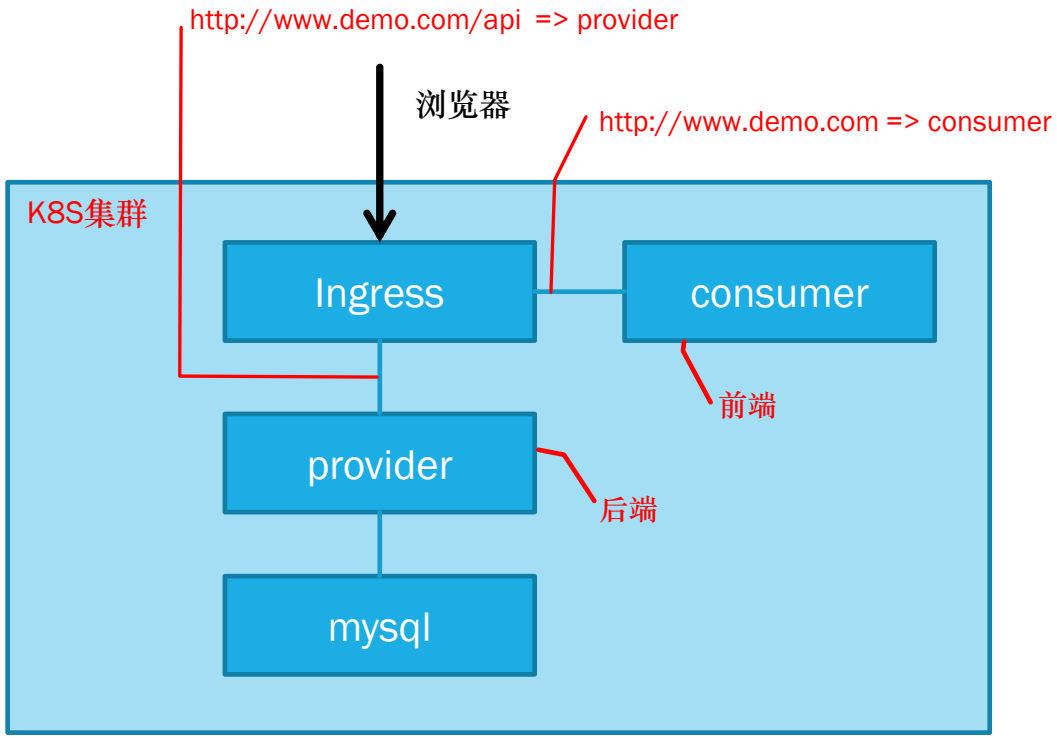
前、后端分离的开发模式


浏览器
静态文件服务器
请求静态页面
获取静态页面
HTML、CSS、JS
返回静态文件
运行JS
请求后端数据
填充到页面中
应用服务器
数据库
请求数据
返回数据
对外提供统一API
根据请求操作数据
返回所需数据
(JSON、XML)
不负责前端页面处理
App
请求数据
请求数据
在界面展示
返回数据
后端开发

一个前、后端开发的例子

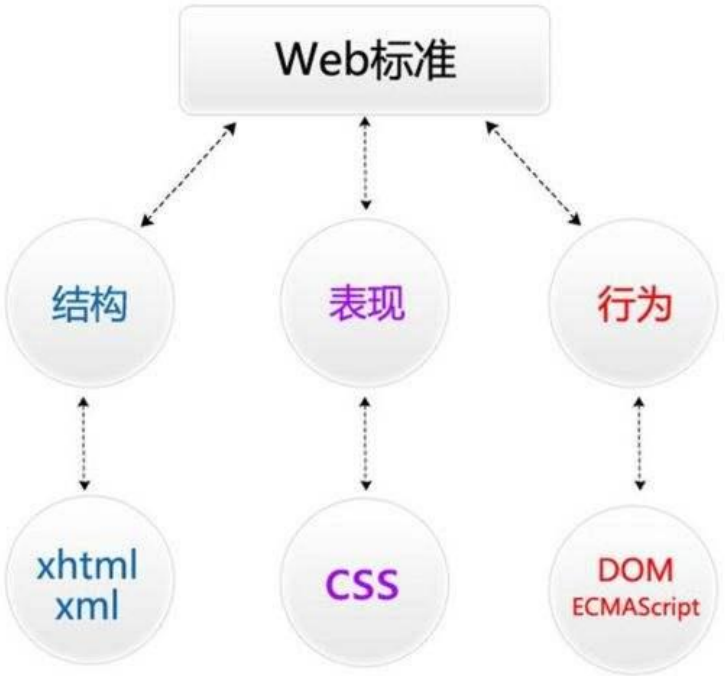
http://www.demo.com

部署图


http://www.demo.com/api => provider
浏览器
http://www.demo.com => consumer
K8S集群
Ingress
consumer
前端
provider
后端
mysql

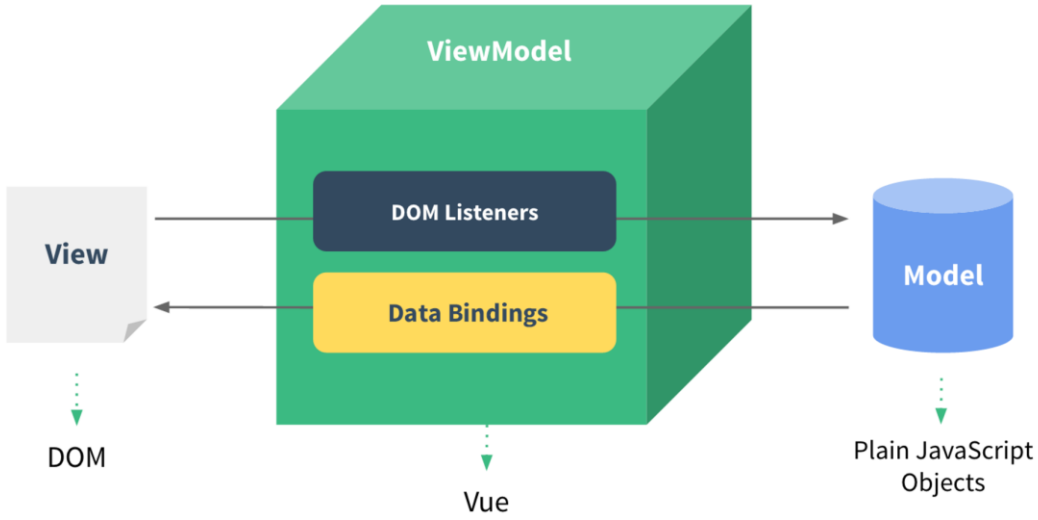
前端开发的基础

- HTML、CSS 和 JavaScript
- Node.js，是一个Javascript运行环境。它让Javascript可以开发后端程序，实现几乎其他后端语言实现的所有功能
- NPM，全称是Node Package Manager，https://www.npmjs.com/，是一个NodeJS包管理和分发工具，已经成为了非官方的发布Node模块（包）的标准



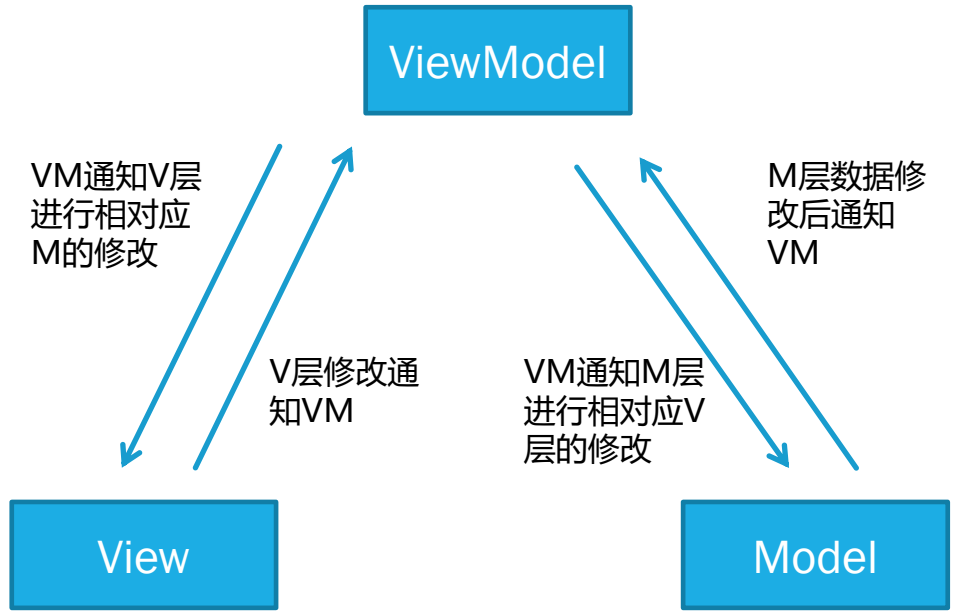
前端开发框架：Vue.js

- 官网： https://v3.cn.vuejs.org/
- Vue.js是一款流行的JavaScript前端框架，旨在更好地组织与简化Web开发
- Vue所关注的核心是MVC模式中的视图层，同时，它也能方便地获取数据更新，并通过组件内部特定的方法实现视图与模型的交互



前端开发的MVVM架构模式

- View，视图，就是DOM，它负责将数据模型转化成UI展现出来
- Model，数据模型，就是Vue组件里的data
- ViewModel，监听模型数据的改变和控制视图行为、处理用户交互
- View和Model之间并没有直接的联系，而是通过ViewMode进行交互，Model和ViewModel之间的交互是双向的，因此View数据的变化会同步到Model中，而Model数据的变化也会立即反应到View上



单文件组件(single-file components)

- 模板、逻辑和样式在一个组件里，组件更加内聚且更可维护
- 组件是一种抽象，允许我们使用小型、独立和通常可复用的组件构建大型应用

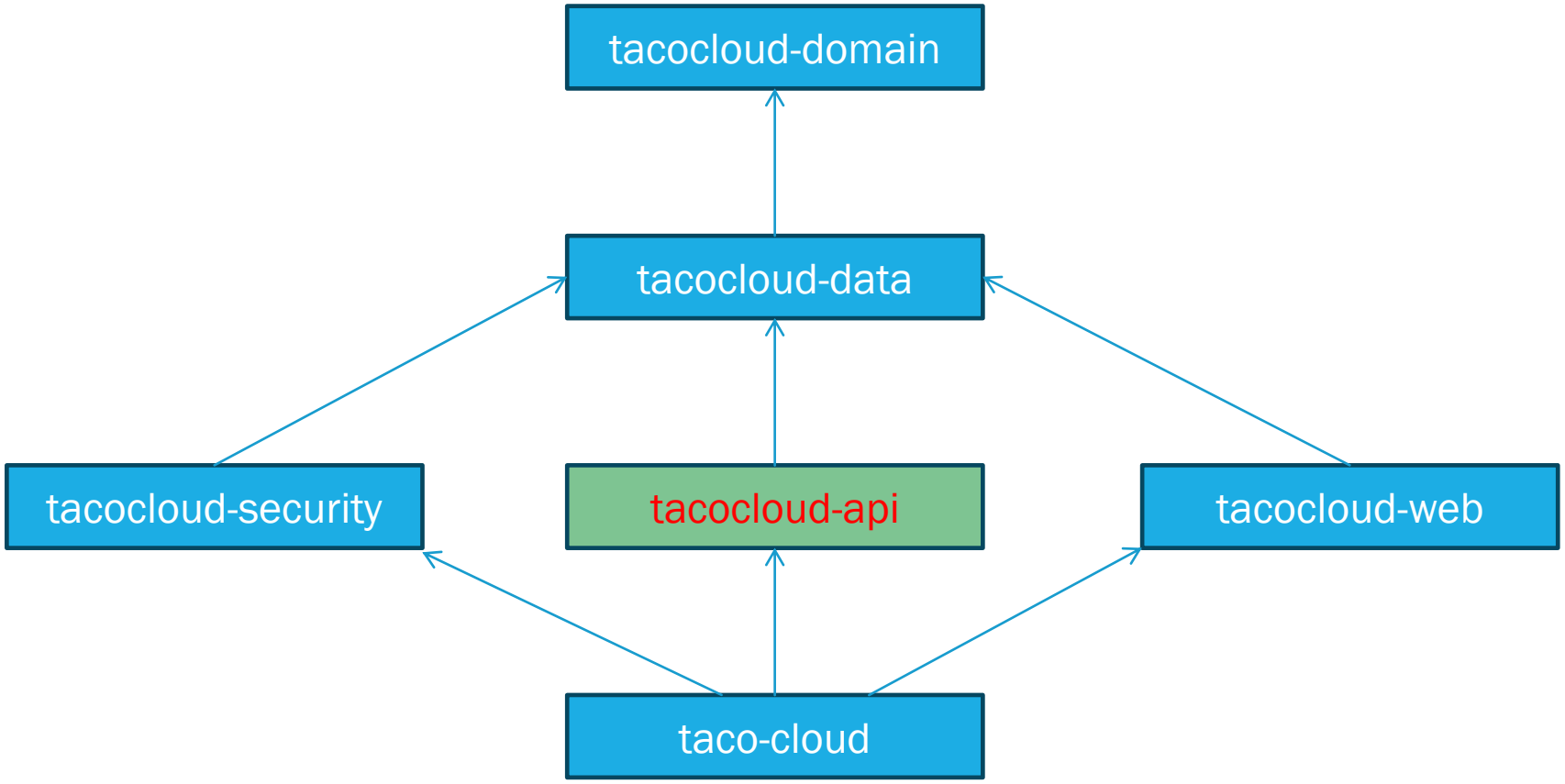
curl工具

- curl 是常用的命令行工具，用来请求 Web 服务器。它的名字就是客户端（client）的 URL 工具的意思。它的功能非常强大，命令行参数多达几十种。如果熟练的话，完全可以取代 Postman 这一类的图形界面工具
- 安装：https://curl.se/download.html

Rest原则

- Representational State Transfer，表现层状态转移
- 资源（Resources），就是网络上的一个实体，标识：URI
- 表现层（Representation）：json、xml、html、pdf、excel
- 状态转移（State Transfer）：服务端--客户端
- HTTP协议的四个操作方式的动词：GET、POST、PUT、DELETE
  - CRUD：Create、Read、Update、Delete
- 如果一个架构符合REST原则，就称它为RESTful架构

组件（模块）依赖关系


tacocloud-domain
tacocloud-data
tacocloud-security
tacocloud-api
tacocloud-web
taco-cloud

配置本地域名

- Windows：C:\Windows\System32\drivers\etc\hosts
- Linux：/etc/hosts

  127.0.0.1 tacocloud

  127.0.0.1 authserver

RESTful控制器实现

- REST API以面向数据的格式返回，JSON或XML
- 这些注解继续有用

  @RequestMapping

  @GetMapping

  @PostMapping

  @PutMapping

  @DeleteMapping

  @PatchMapping
- @RestController，@ResponseBody，或返回ResponseEntity对象，TacoController
- @RequestMapping的produces属性

请求头与请求体

- 请求头：请求头由 key/value 对组成，每行为一对，key 和 value 之间通过冒号(:)分割。请求头的作用主要用于通知服务端有关于客户端的请求信息。
  - ✓ User-Agent：生成请求的浏览器类型
  - ✓ Accept：客户端可识别的响应内容类型列表；星号* 用于按范围将类型分组。*/*表示可接受全部类型，type/*表示可接受 type 类型的所有子类型。
  - ✓ Accept-Language: 客户端可接受的自然语言
  - ✓ Accept-Encoding: 客户端可接受的编码压缩格式
  - ✓ Accept-Charset： 可接受的字符集
  - ✓ Host: 请求的主机名，允许多个域名绑定同一 IP 地址
  - ✓ connection：连接方式（close 或 keepalive）
  - ✓ Cookie: 存储在客户端的扩展字段
  - ✓ Content-Type:标识请求内容的类型
  - ✓ Content-Length:标识请求内容的长度
- 请求体：请求体主要用于 POST 请求，与 POST 请求方法配套的请求头一般有 Content-Type和 Content-Length

Accept取值

- text/html： HTML格式
- text/plain：纯文本格式
- text/xml： XML格式
- image/gif：gif图片格式
- image/jpeg：jpg图片格式
- image/png：png图片格式
- video/mpeg：视频
- vedio/quicktime：视频
- application/xhtml+xml：XHTML格式
- application/xml： XML数据格式
- application/atom+xml ：Atom XML聚合格式
- application/json： JSON数据格式
- application/pdf：pdf格式
- application/msword： Word文档格式
- application/octet-stream： 二进制流数据（如常见的文件下载）
- application/x-www-form-urlencoded： < form encType=””>中默认的encType，form表单数据被编码为key/value格式发送到服务器（表单默认的提交数据的格式）

响应头与响应体

- 状态行：由 HTTP 协议版本、状态码、状态码描述三部分构成，它们之间由空格隔开。
- 状态码：由 3 位数字组成，第一位标识响应的类型，常用的5大类状态码如下：
  - ✓ 1xx：表示服务器已接收了客户端的请求，客户端可以继续发送请求
  - ✓ 2xx：表示服务器已成功接收到请求并进行处理
  - ✓ 3xx：表示服务器要求客户端重定向
  - ✓ 4xx：表示客户端的请求有非法内容
  - ✓ 5xx：标识服务器未能正常处理客户端的请求而出现意外错误
- 响应头
  - ✓ Location：服务器返回给客户端，用于重定向到新的位置
  - ✓ Server： 包含服务器用来处理请求的软件信息及版本信息Vary：标识不可缓存的请求头列表
  - ✓ Connection: 连接方式， close 是告诉服务端，断开连接，不用等待后续的请求了。 keep-alive 则是告诉服务端，在完成本次请求的响应后，保持连接
  - ✓ Keep-Alive: 300，期望服务端保持连接多长时间（秒）
- 响应内容：服务端返回给请求端的文本信息。

消息转换器

- 使用注解@ResponseBody或类级@RestController，作用：指定使用消息转换器
- 没有model和视图，控制器产生数据，然后消息转换器转换数据之后的资源表述。
- spring自动注册一些消息转换器（HttpMethodConverter），不过类路径下要有对应转换能力的库，如：Jackson Json processor、JAXB库
- 请求传入，@RequestBody以及HttpMethodConverter

@CrossOrigin注解

- CORS， Cross Origin Resource Sharing

@GetMapping

- recentTacos、tacoById方法实现
- http://tacocloud:8080/api/tacos/3
- @PathVariable
- 如果未查询到元素，返回状态码200，body返回null，如果不使用Optional类型，则返回状态码500
- 返回ResponseEntity

@PostMapping

- postTaco方法实现
- consumes属性
- @RequestBody
- @ResponseStatus，指定返回状态码

更多

- @PutMapping，putOrder方法实现，”将数据放到这个URL上“
- @PatchMapping，patchOrder方法实现，局部更新
- @DeleteMapping，deleteOrder方法实现，HttpStatus.NO_CONTENT，body不需要返回数据

接口设计

- 使用标准HTTP动词：GET、PUT、POST、DELETE，映射到CRUD
- 使用URI来传达意图
- 请求和响应使用JSON
- 使用HTTP状态码来传达结果

内容

1. 使用Spring MVC的控制器创建RESTful端点
2. 将Spring Data存储库暴露为REST端点
3. 测试和保护端点

添加依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
```

Spring HEATEOAS项目

- 超媒体作为应用状态引擎（Hypermedia AsThe Engine Of Application State,HEATEOAS）
- 消费这个API的客户端可以使用这些超链接作为指南，以便于导航API并执行后续的请求
- 也会生成POST和PUT请求

设置API基础路径

```
spring:
  data:
    rest:
      base-path: /data-api
```

调整关系名和路径

```
@Data
@Entity
@RestResource(rel="tacos", path="tacos")
public class Taco {
```

分页和排序

- http://tacocloud:8080/data-api/tacos?size=15&page=0&sort=createdAt,desc

内容

1. 使用Spring MVC的控制器创建RESTful端点
2. 将Spring Data存储库暴露为REST端点
3. 测试和保护端点

RestTemplate

- getForObject、getForEntity
- postForObject、postForEntity、postForLocation
- put
- delete
- execute、exchange

使用Feign调用REST API

```
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-feign</artifactId>
</dependency>
```

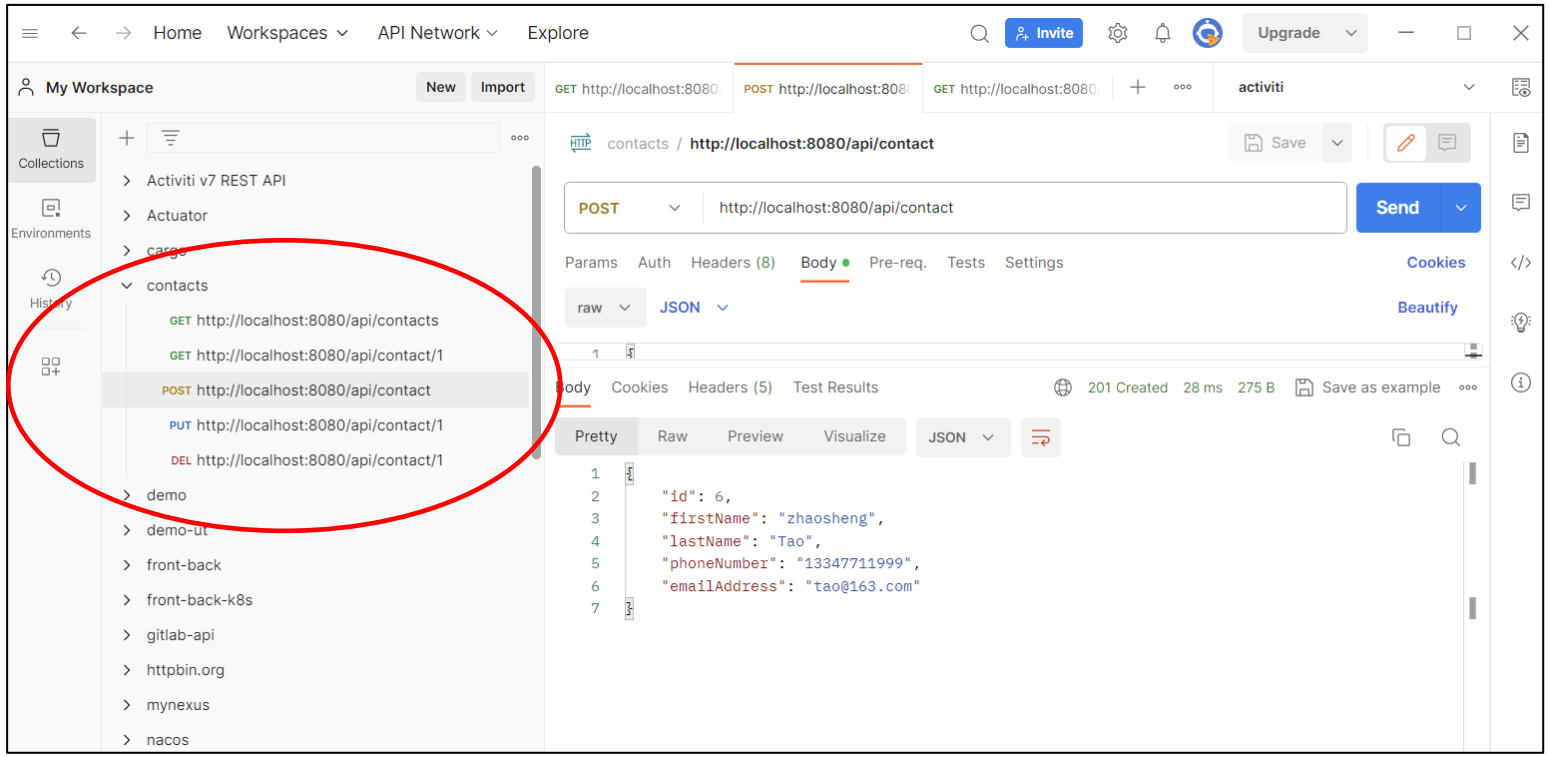
```
@FeignClient("organizationservice")
public interface OrganizationFeignClient {
    @RequestMapping(
            method = RequestMethod.GET,
            value = "/v1/organizations/{organizationId}",
            consumes = "application/json")
    Organization getOrganization(@PathVariable("organizationId") String organizationId);
}
```

作业

- 基于第5次课的作业（使用Spring Data JPA做数据库持久化，数据库使用H2），增加REST API实现
- 要实现的接口如下：
  1. GET：http://localhost:8080/api/contacts，获取所有联系人信息
  2. GET：http://localhost:8080/api/contact/{id}，获取指定id的联系人信息，如果不存在返回状态码404（HttpStatus.NOT_FOUND）
  3. POST：http://localhost:8080/api/contact，创建新联系人信息，返回状态码201（HttpStatus.CREATED）
  4. PUT：http://localhost:8080/api/contact/{id}，更新指定id的联系人信息
  5. DELETE：http://localhost:8080/api/contact/{id}，删除指定id的联系人信息，返回状态码204（HttpStatus.NO_CONTENT）

提交

- 使用postman做好各接口的测试，并提交如下截图
- 所有源代码，压缩成一个文件。不包含编译后的class文件（删除target目录）



谢谢观看！