

马建勋
AU

/thoughtworks

1 业务分析和建模

/thoughtworks

业务分析和建模

业务分析

/thoughtworks

业务背景

本页概括描述业务需求

任你行差旅服务公司 希望可以为中小企业提供差旅服务：

- 公司通过服务企业获取服务费
- 公司通过企业用户提高客流量和交易量

现有主要业务场景：

- 企业通过该平台，需向平台支付一定的固定服务费，从而获取更优质的差旅服务。
- 企业员工通过平台预订差旅无需自费，平台和企业会按月结算。
- 平台对特定的差旅服务收取服务费，包括：机票改签，取消等。

企业差旅服务 - 合约分析

寻找凭证与合同，分析合约上下文



企业差旅服务 - 合约文本

明确业务的权责关系：权利方的权利、责任方的责任以及履约时限等

甲方：任你行差旅服务平台

乙方：TW 公司

权责说明：

- 甲方要求乙方在申请服务之前，支付甲方5000元的固定服务费。
- 乙方要求甲方在收到乙方支付的固定服务费后的24 小时内联系乙方，协助开通服务。
- 乙方要求甲方每个月3日之前，生成乙方上月的消费明细，供乙方对账支付。
- 甲方要求乙方在每个月10 之前对乙方上个月产生的账单进行支付
- 乙方要求甲方在完成支付24小时内，生成并发送乙方对应的发票
- 甲方要求乙方在退改机票的时候，每单应付给甲方20元服务费

差旅预订 - 合约分析

寻找凭证与合同，分析合约上下文

RFP

Proposal

Contract

Fulfillment

差旅价格

差旅订单

订单提交凭证

订单改签凭证

订单取消凭证

合约名称1 - 合约文本

明确业务的权责关系：权利方的权利、责任方的责任以及履约时限等

甲方：任你行差旅平台

乙方：mjx

权责说明：

- 乙方要求甲方在乙方选择提交订单后，为乙方生成订单，否则下单失败。
- 乙方要求甲方在乙方选择取消订单后，取消乙方的订单。
- 乙方要求甲方在乙方选择改签订单后，让乙方能对订单进行改签。

业务分析和建模

业务建模

/thoughtworks

业务建模图 - 图例参考

时标对象 (Evidences)

<rfp>
**Request
For Proposal**
created_at
expired_at

方案征集书

<proposal>
Proposal
created_at

提案

<contract>
Contract
created_at

合约

<fulfillment>
**Fulfillment
Request**
created_at
expired_at

履约请求

<fulfillment>
**Fulfillment
Confirmation**
created_at

履约确认

<evidence>
Evidence
created_at

凭证

参与者 (Participants)

<party>
Party

参与方

<thing>
Thing

标的物

<place>
Place

场所

角色 (Roles)

<party>
Party

合约中的角色

<domain>
Domain

领域逻辑角色化

<3rd system>
3rd System

第三方系统角色化

<evidence>
**Evidence
As
Role**
created_at

凭证角色化

关系 (Relationships)

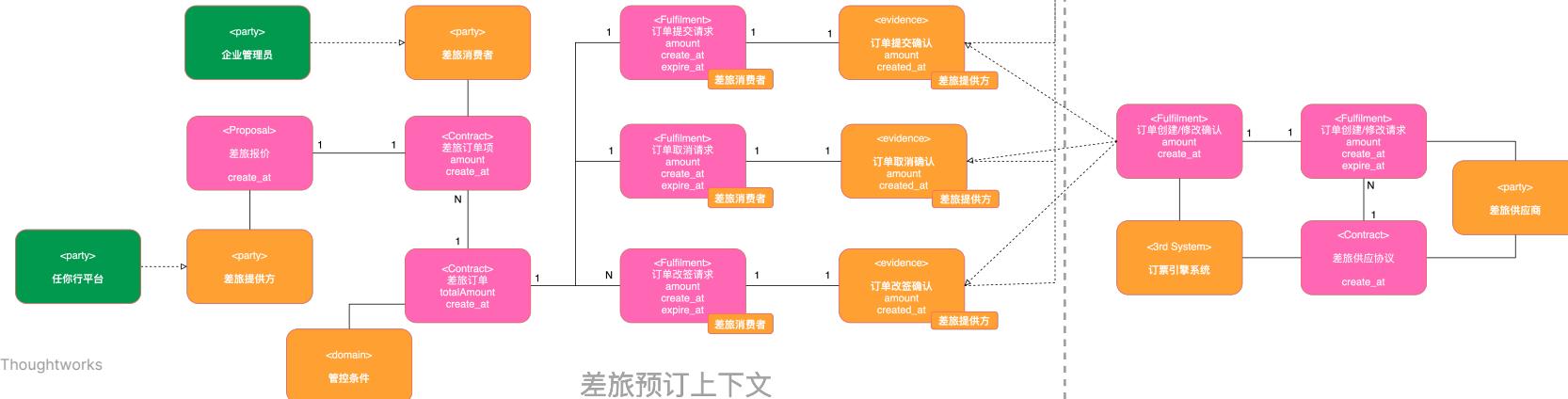
1 _____ N

联系 (有数量)

联系 (无数量)

扮演某角色

业务建模图



2 架构设计

/thoughtworks

架构设计
服务与业务能力

/thoughtworks

业务能力表 - 企业差旅服务上下文

业务建模将模型实现为 RESTful API

角色	Method	URI	业务能力	业务能力服务 (对应架构图中的业务能力服务)
企业管理员	GET	bills	查看所有账单	企业差旅服务系统
企业管理员	GET	bills/{:bid}	查看账单明细	企业差旅服务系统
企业管理员	POST	bills	生成账单请求	企业差旅服务系统
平台	POST	bills/confirmation	生成账单确认	企业差旅服务系统
企业管理员	GET	bills/{:bid}/invoice	查看账单发票	企业差旅服务系统
企业管理员 / 平台	POST	bills/{:bid}/invoice	生成发票请求	企业差旅服务系统
平台	POST	bills/{:bid}/invoice/confirmation	生成发票确认	企业差旅服务系统
企业管理员	POST	bills/{:bid}/payment	账单支付完成请求	企业差旅服务系统
平台	POST	bills/{:bid}/payment/confirmation	账单支付完成确认	企业差旅服务系统

业务能力表 - 企业差旅服务上下文

业务建模将模型实现为 RESTful API

角色	Method	URI	业务能力	业务能力服务 (对应架构图中的业务能力服务)
企业管理员	POST	account/payment	固定服务费支付完成请求	企业差旅服务系统
平台	POST	account/payment/confirmation	固定服务费支付完成请求	企业差旅服务系统
企业管理员	GET	account	查看企业服务状态	企业差旅服务系统
平台	POST	account/start	企业服务开通请求	企业差旅服务系统
企业管理员	POST	account/start/confirmation	企业服务开通确认	企业差旅服务系统
平台	POST	orders/{:oid}/charge	订单改签服务费创建请求	企业差旅服务系统
企业管理员	POST	orders/{:oid}/charge/confirmation	订单改签服务费创建确认	企业差旅服务系统
© 2021 Thoughtworks				

业务能力表 - 差旅预订服务上下文

业务建模将模型实现为 RESTful API

角色	Method	URI	业务能力	业务能力服务 (对应架构图中的业务能力服务)
企业员工	GET	orders	获取我的所有订单	差旅预订系统
企业管理员 / 企业员工	GET	orders/{:id}	查看订单	差旅预订系统
企业员工	POST	orders	创建订单请求	差旅预订系统
平台	POST	orders/confirmation	创建订单确认	差旅预订系统
企业员工	POST	orders/{id}/cancellation	取消订单请求	差旅预订系统
平台	POST	orders/{id}/cancellation/confirmation	取消订单确认	差旅预订系统
平台	POST	orders/{id}/change	改签订单请求	差旅预订系统
平台	POST	orders/{id}/change/confirmation	改签订单确认	差旅预订系统

业务能力表 - 差旅预订服务上下文

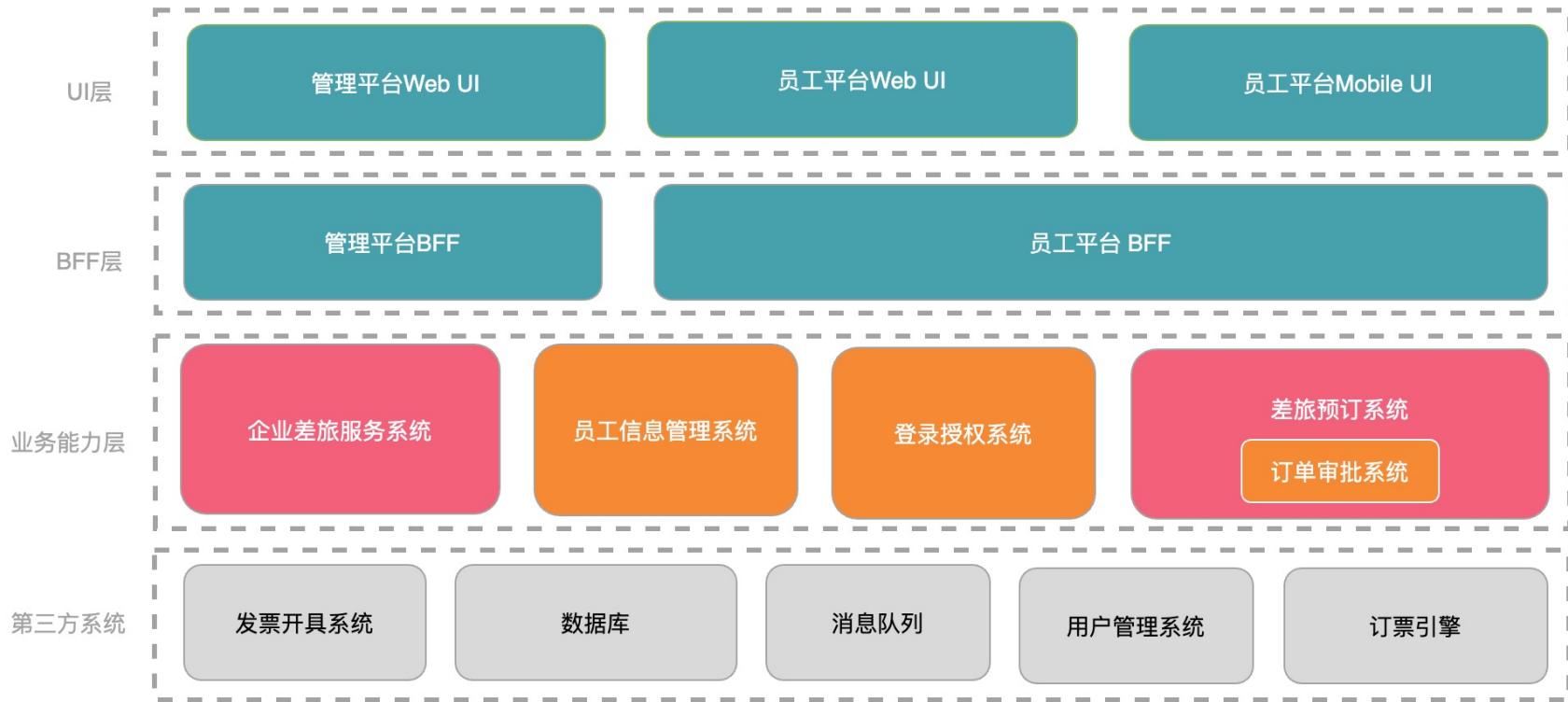
业务建模将模型实现为 RESTful API

角色	Method	URI	业务能力	业务能力服务 (对应架构图中的业务能力服务)
企业主管员工	GET	orders/approval	获取所有待我审批的订单	差旅预订系统 – 管控条件
企业主管员工	POST	orders/{:oid}/approval	审批通过订单	差旅预订系统 – 管控条件
企业主管员工	POST	orders/{:oid}/reject	驳回审批订单	差旅预订系统 – 管控条件
企业主管员工	GET	department/orders	查看部门所有订单	差旅预订系统 – 管控条件
企业管理员	GET	account/orders	查看公司所有账单	差旅预订系统

架构设计
进程间架构设计

/thoughtworks

进程间架构设计 - 架构图



进程间架构设计 - 架构图说明

管理平台Web UI：对企事业管理员用户提供电脑浏览器端的数据和操作显示

管理平台BFF：提供管理平台Web UI 数据汇总和请求验证转发

员工平台Web UI：对企业员工用户提供电脑浏览器端的数据和操作显示

员工平台Mobile UI：对企业员工用户提供手机端的数据和操作显示

员工平台BFF：对员工平台Web UI 和员工平台Mobile UI提供数据汇总和请求验证转发

登录授权系统：提供用户登录及其授权功能

员工信息管理系统：提供企业员工信息的管理功能

企业差旅服务系统：提供企业管理与平台对接处理的功能，包括开户，结算，明细查看，发票查看等功能。

差旅预订系统：提供企业员工预订差旅服务功能，包括下单，取消订单和改签等。

订单审批系统：提供订单按照管控条件的审批功能，包含自动审批通过，同意和驳回等功能。

发票开具系统：提供与第三方开发票集成服务。

订票引擎：提供所有差旅服务与其供应商之间的预订和退改等服务。

数据库：提供数据保存功能。

消息队列：提供事件存储和推送，通知等功能。

架构设计

进程中架构设计

/thoughtworks

进程内架构设计 -企业差旅服务系统

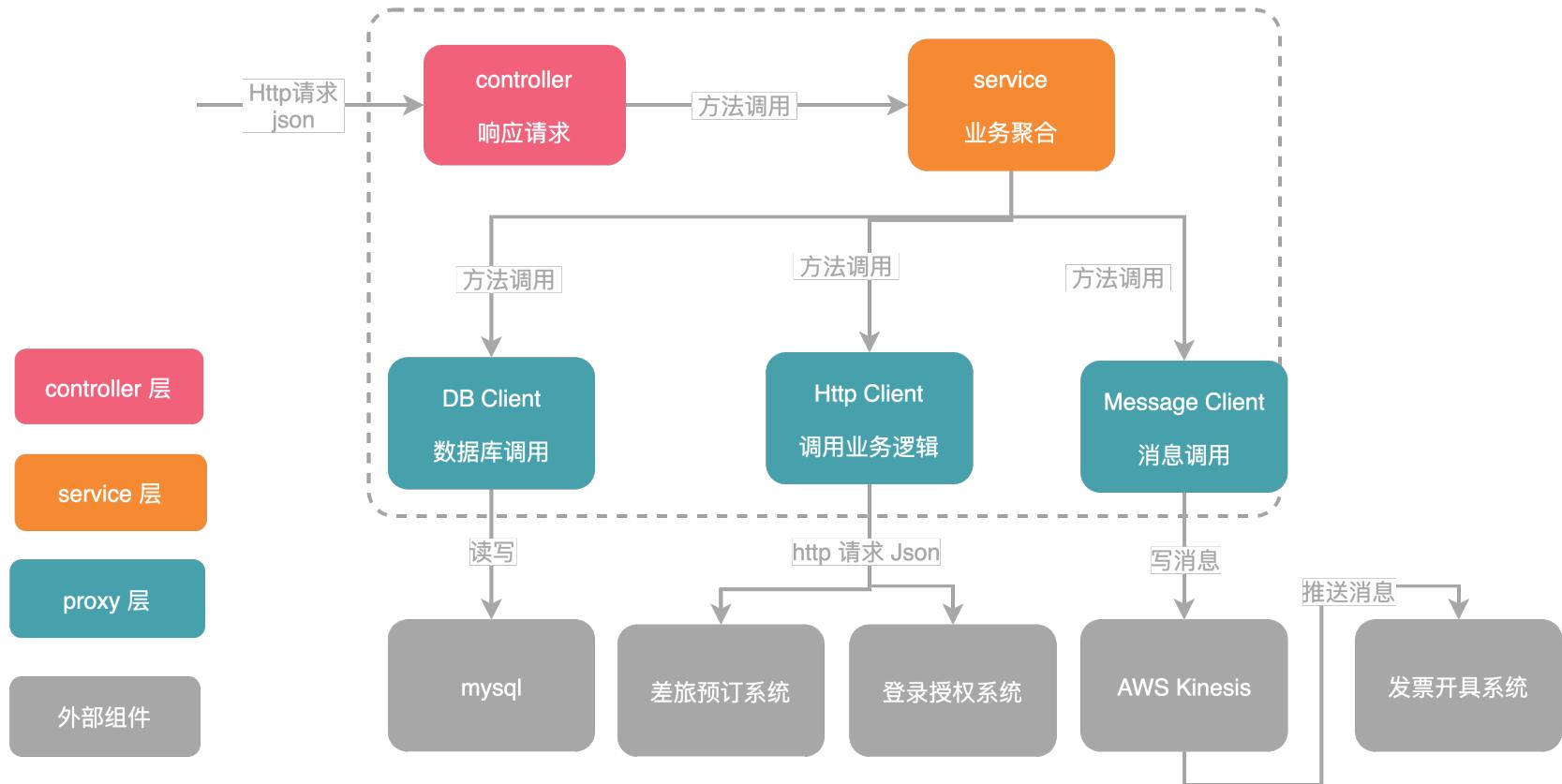
该进程为企业差旅服务系统

其职责是提供企业管理与平台对接处理的功能，包括开户，结算，明细查看，发票查看等功能。

它以 RESTful API 的形式对外提供服务

采用的技术选型为 Nodejs + express + typescript + jest

进程内架构 - 架构图



进程内架构 - 架构图

controller 层，负责响应外部请求，验证请求参数和通过调用service层生成响应数据

Service 层，负责请求的具体业务处理，或数据生成，通过Client 层跟外部系统通信。

Client 层，是对外请求接口，负责service层与外部服务的交流

组件 HttpClient：提供通过HTTP调用其他服务的数据

组件 DBClient：提供数据访问接口，提供与数据库交互能力

组件 MessageClient：提供消息队列访问接口

应对进程间分区 - 保证业务准确性

举例说明分区场景和采用CP的应对方案

正常情况下，该进程调用用户登录授权系统完成用户登录
当分区异常发生时，调用登录授权系统失败，会产生

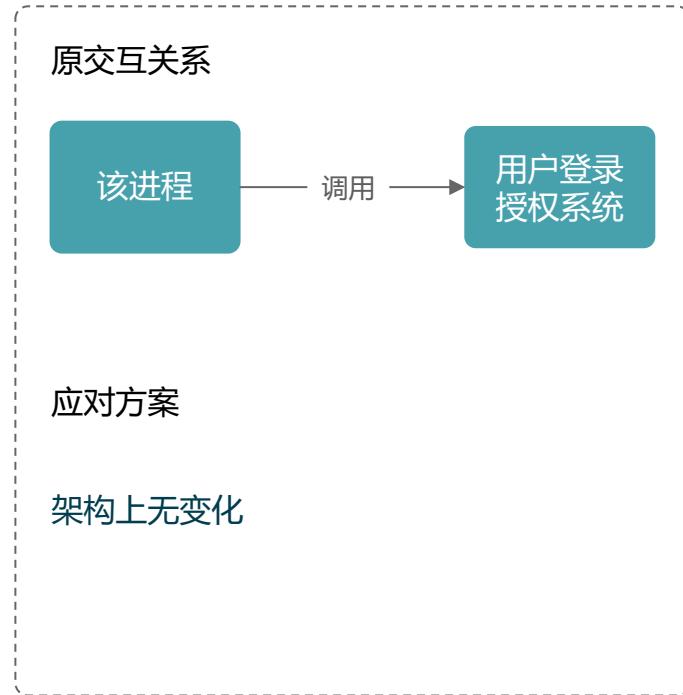
企业管理员无法进入系统进行操作

业务方想要严格控制用户及数据的安全性

为此需要保证业务的准确性。

为了保证业务准确性，在架构上采用了以下应对方案，如右图：

使用直接调用的关系，如果调用失败，则及时反馈用户服务不可用，不会有不一致性的状态产生，从而保证业务的准确性。



应对进程间分区 - 保证业务可用性

举例说明分区场景和采用AP的应对方案

正常情况下，该进程调用发票开具系统完成发票申请

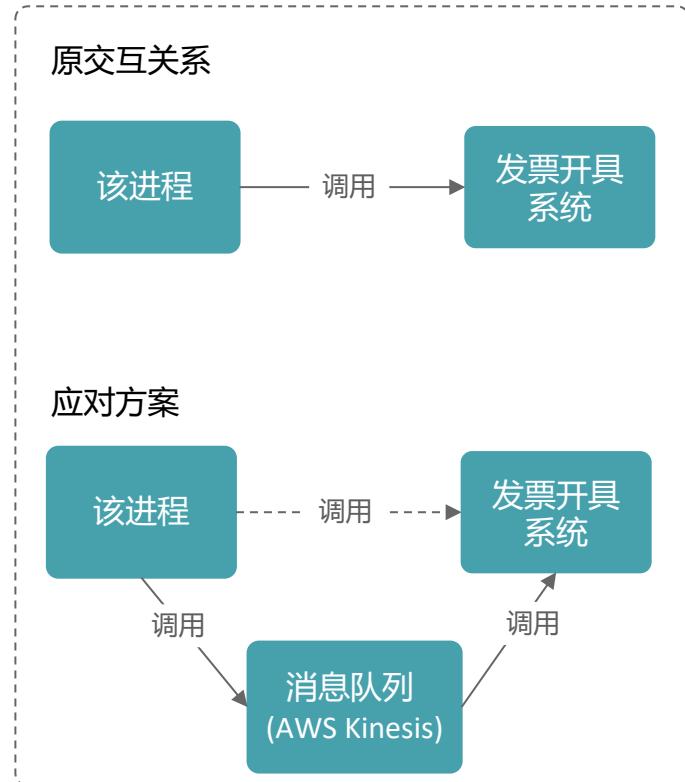
当分区异常发生时，调用发票开具系统失败，会产生

无法申请发票

业务方想要用户永远能够发票申请成功

为此需要保证业务的可用性。

为了保证业务可用性，在架构上采用了以下应对方案，如右图：在调用过程中添加一个消息队列，来帮助缓存请求。当分区发生时，请求暂时被换存在消息队列中，一旦分区恢复，则可以继续执行，但是用户对此并不感知，没有阻塞用户行为。消息队列相对发票开具系统的可用性是特别高的，所以这样设计一定程度降低了系统的一致性，提升了可用性。



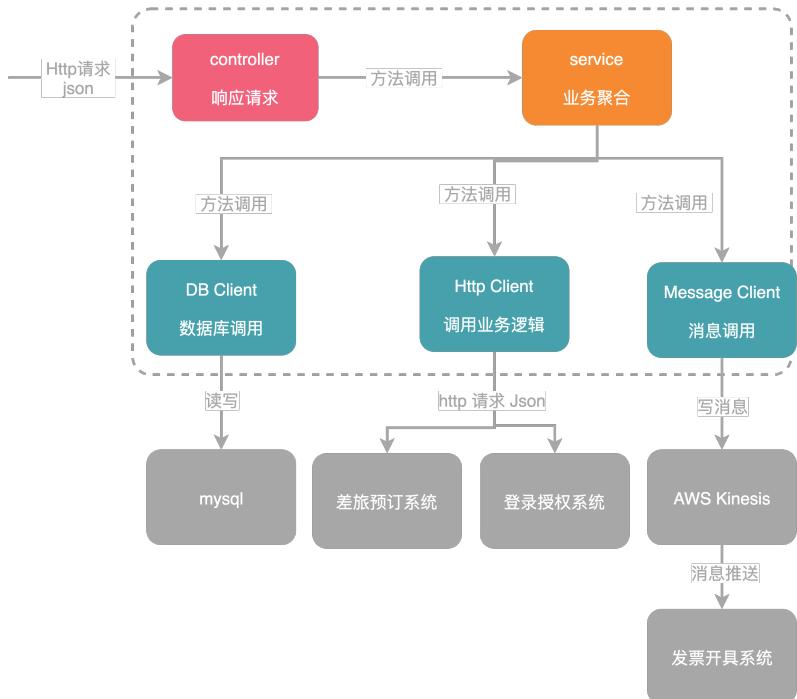
架构设计

进程中架构的测试策略

/thoughtworks

测试策略 - 全景

进程内架构图

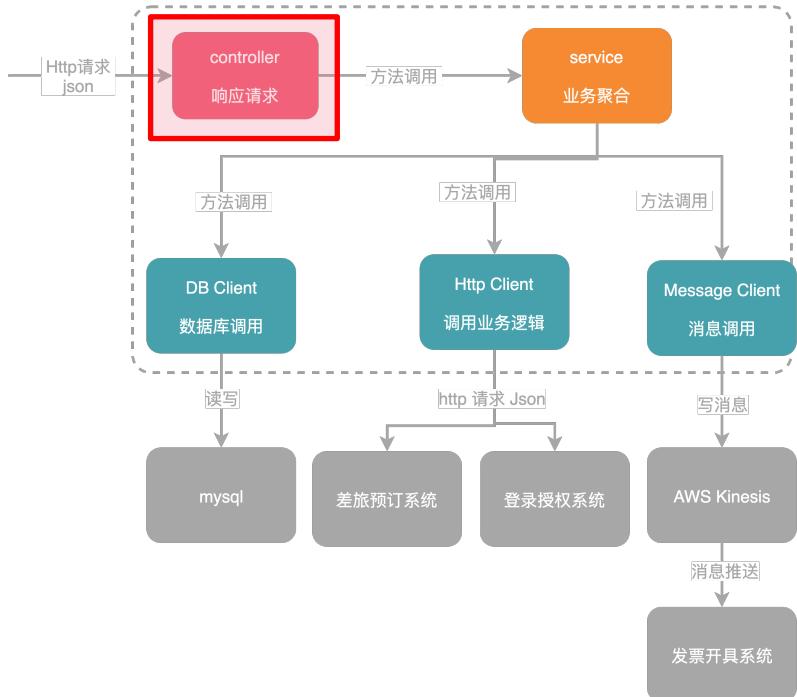


针对当前架构，采用以下测试策略覆盖进程中各个组件并保证进程边界处的测试有效性，来帮助发现问题和定位问题：

序号	被测组件 (一个或多个组件的组件名称)	测试类型 组件测试/单元测试/功能测试等	测试象限 (Q1 - Q4)
1	Controller	功能测试	Q2
2	Controller + Service	组件测试	Q1
3	Service + DB Client	组件测试	Q1
4	Service + Http Client	组件测试	Q1
5	Service + Message Client	组件测试	Q1

测试策略 - 1 - Controller

进程内架构图



被测组件：

Controller

采用 功能测试属于第2象限的测试，

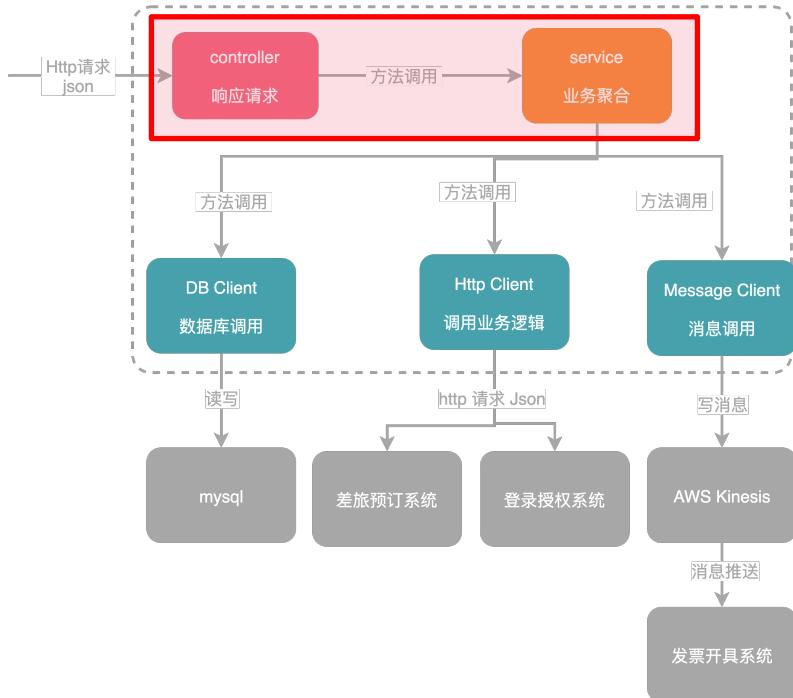
因为Controller是面向请求，它能否正确的返回对应值的以为整个服务提供的功能是否完整和正确。

测试时，会依赖：

- 进程内组件 **service**，采用Fake Object来替代该组件，因为该测试无需测试对**service**的调用情况和内部状态确认，对**service**的替代仅仅是为了让**controller**运行起来。
- 进程外组件 **httpClient** 来发起http请求，采用一个 Postman来辅助,因为只在乎能不能发起http请求。

测试策略 - 2 - Controller + service

进程内架构图



被测组件：

Controller + service

采用组件测试，属于第1象限的测试，

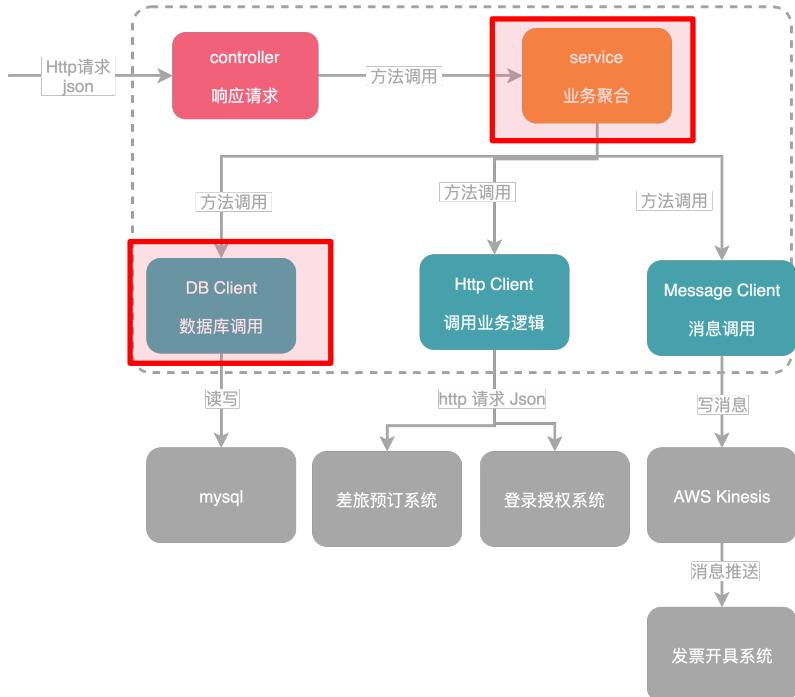
从业务的角度出发，帮助团队定位 controller 和 service 可能存在的问题；

测试时，会依赖：

- 进程内组件 DB Client，采用 Test Stub 来替代该组件，因为偏向黑盒测试，不关注具体实现
- 进程内组件 Http Client，采用 Test Stub 来替代该组件，因为偏向黑盒测试，不关注具体实现
- 进程内组件 Message Client，采用 Test Stub 来替代该组件，因为偏向黑盒测试，不关注具体实现

测试策略 - 3 - Service + DB Client

进程内架构图



被测组件：

Service + DB Client

采用组件测试，属于第1象限的测试，

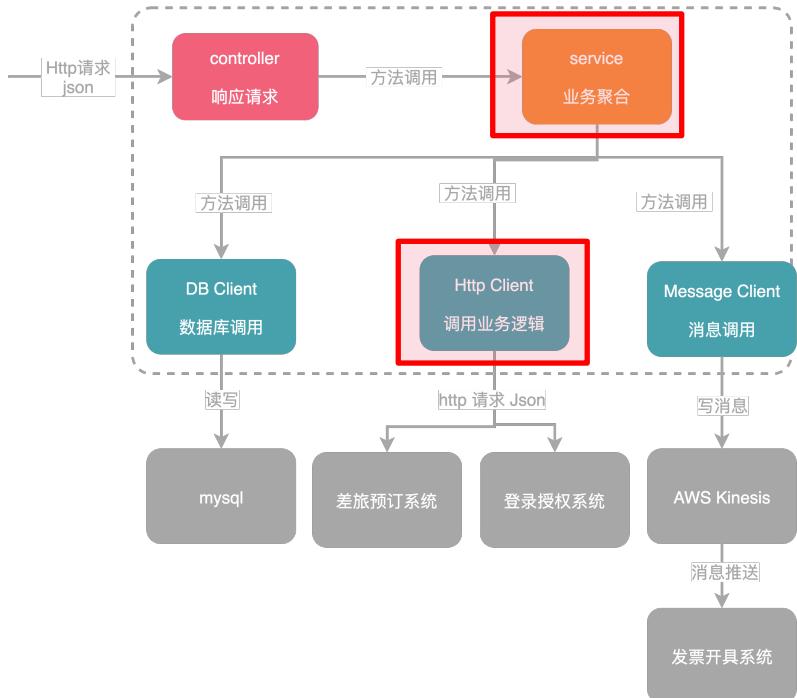
这样保证Service正确的调用DBClient，也可以验证DBClient的逻辑。能够快速的定位问题。

测试时，会依赖：

- 进程内组件 **Message Client**、**Http Client**，采用 **Test Stub** 来替代该组件，因为偏向黑盒测试，不关注具体实现
- 进程外组件 **mysql**，采用 **fake object** 来替代该组件，因为不仅关注被调用，还要关注到用参数，是个偏白盒测试。

测试策略 - 4 - Service + Http Client

进程内架构图



被测组件：

Service + Http Client

采用组件测试，属于第1象限的测试，

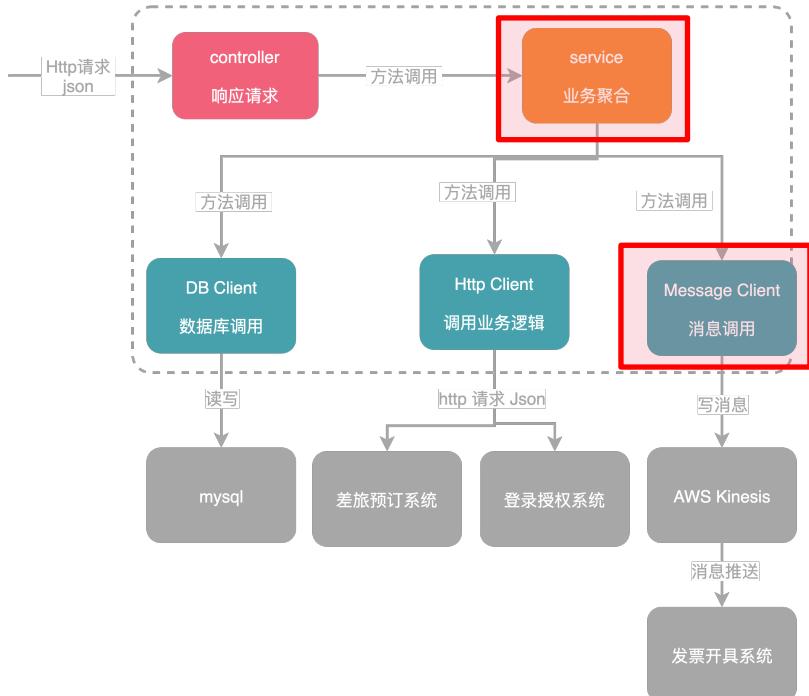
这样保证Service正确的调用HttpClient，也可以验证HttpClient的逻辑。能够快速的定位问题。

测试时，会依赖：

- 进程内组件 DB Client、Message Client，采用 Test Stub 来替代该组件，因为偏向黑盒测试，不关注具体实现
- 进程外组件差旅预订系统，采用 fake object 来替代该组件，因为不仅关注被调用，还要关注到用参数，是个偏白盒测试。

测试策略 - 5 - Service + Message Client

进程内架构图



被测组件：

Service + Message Client

采用组件测试，属于第1象限的测试，
这样保证Service正确的调用MessageClient，也可以验证
MessageClient的逻辑。能够快速的定位问题。

测试时，会依赖：

- 进程内组件 DBClient、Http Client，采用 Test Stub 来替代该组件，因为偏向黑盒测试，不关注具体实现
- 进程外组件AWS Kinesis，采用 fake object 来替代该组件，
因为不仅关注被调用，还要关注到用参数，是个偏白盒
测试。

3 效能管理

/thoughtworks

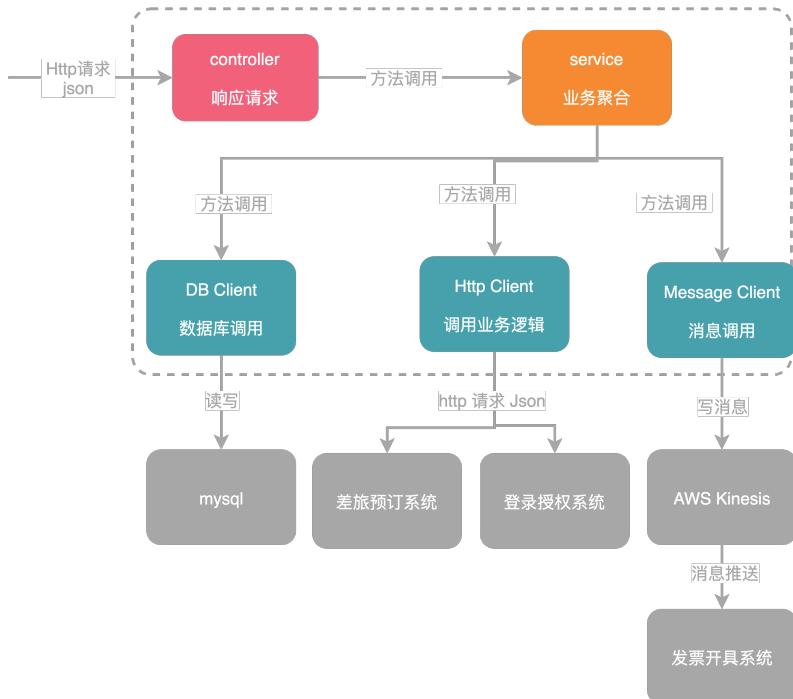
效能管理

工序设计与任务拆解

/thoughtworks

工序设计

进程内架构图



针对当前架构，设计以下工序来实现架构，覆盖所有组件并应用此前设计的测试策略：

Controller

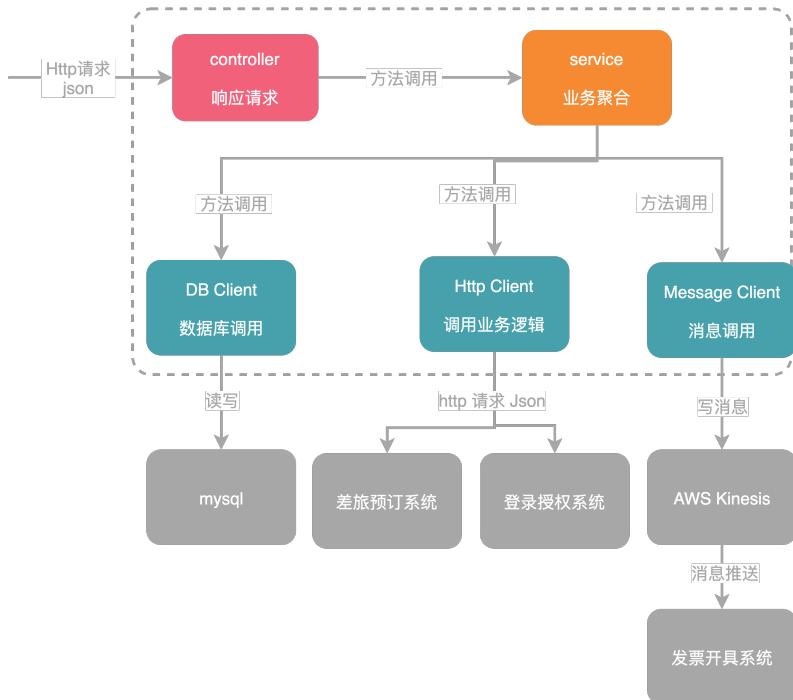
- 工序1：Fake service，实现Controller，使其能够响应http请求，并对输入验证，返回对应的结果。

Controller + Service

- 工序2：stub DBClient，Http Client，Message Client，实现controller在能够正确的调用service，并正确的返回service的逻辑结果，实现service对应的业务逻辑。

工序设计

进程内架构图



Service + DB Client

- 工序3：stub Http Client、Message Client , fake mysql , 实现service在能够正确的调用 DBClient , 将业务数据持久化在数据库中。

Service + Http Client

- 工序4：stub DB Client、Message Client , 差旅预订系统 , spy 登陆授权系统 , service 能够通过Http Client调用登录授权系统，能够获取用户详细信息。
- 工序5：stub DB Client、Message Client、登陆授权系统 , spy 差旅预订系统 , service 能够通过Http Client调用差旅预订系统，能够获取订单详细信息。

Service + Message Client

- 工序6：stub DB Client、Http Client , spy AWS Kinesis , 实现 service 通过AWS Kinesis 事件完成发票开具的功能。

Story 1 生成账单描述及AC

Story:

作为平台管理员，我想要在生成上个月账单，以便于我司能够对账付款。

计算逻辑：

账单金额 = 上月所有成功的订单金额 + 上月的退改服务费

上月所有订单从差旅预订服务中获取。

上月退改服务费从数据库中获取

ACs:

AC1:

Given : 企业管理员请求账单

When : 上个月的账单还未曾生成

Then : 返回200

AC2:

Given : 企业管理员请求账单

When : 上个月的账单已经生成

Then : 返回 403 禁止重复生成账单

AC3:

Given : 非企业管理员请求账单

When : 上个月的账单还未曾生成

Then : 返回 401 未授权

Story 1 生成账单描述及AC

相关 API 定义 : API 设计

Method: POST

Endpoint: /bills

Request-header:

x-access-token: string. User token

Success:

* 成功 code 200

Code: 200

Response Body:

{

“id”: 100012,

“number”: “ZD202112”

“amount”: 120000

“fromDate”: “2021-12-01”,

“endDate”: “2021-12-31”,

Failure:

- token 丢失 code 404
- 不是管理员权限 code 401
- 账单已生成, code: 403
 - { code: 403,
message: ‘订单已存在’}
- 数据库连接失败 503
 - { code: 503,
message: ‘数据库连接失败’}
- 调用用户登录授权系统失败 503
 - { code: 503,
message: ‘用户登录授权系统访问失败’}
- 调用差旅预订系统失败 503
 - { code: 503,
message: ‘用户登录授权系统访问失败’}

Story 1 生成账单描述及AC

相关 API 定义：外部API

获取企业上月所有订单API

Method: POST
Endpoint: /account/orders
Request-header:
x-access-token: string. User token
Request-body:
from-date: 1628035200000 开始时间戳
end-date: 1630713600000 结束时间戳
Success:
* 成功 code 200
Code: 200
Response Body: [
{ id: 1000010, amount: 1000},
.....
{ id: 1000220, amount: 45.0},
]
* 失败 code 500

账单数据库创建脚本

```
CREATE TABLE `travel`.`bill` (`id` int NOT  
NULL AUTO_INCREMENT, `number`  
varchar(20) NOT NULL, `create_date`  
TIMESTAMP(0) DEFAULT  
CURRENT_TIMESTAMP, `amount`  
DECIMAL(10,2), `invoice_status` boolean  
NOT NULL DEFAULT '0', `payment_status`  
boolean NOT NULL DEFAULT  
'0', `comments` VARCHAR(400), PRIMARY  
KEY (id);
```

```
CREATE TABLE `travel`.`bill_item` (`id` int  
NOT NULL AUTO_INCREMENT, `bill_id` int  
NOT NULL, `amount`  
DECIMAL(10,2), `transaction_type`  
VARCHAR(60), `refer_id` int, `comments`  
VARCHAR(400), PRIMARY KEY (id);
```

获取用户信息api:

Method: GET
Endpoint: /auth/user
Request-header:
x-access-token: string. User token
Success:
* 成功 code 200
Code: 200
Response Body:
{ id: 1000010,
role: 'admin'}

* 失败 code 404 token 不合法

工序列表及效能基线

工序1：实现controller的http的响应，返回对应的response（20分钟）

工序2：通过 controller 对 service 的调用，在service 层实现业务逻辑（30分钟）

工序3：实现DBClient，通过service对 DBClient的调用，实现与数据库的交互（45分钟）

工序4：实现HttpClient，通过service对 HttpClient的调用，实现与差旅预订系统的交互（45分钟）

工序5：通过service对 HttpClient的调用，实现与登录授权系统的交互（40分钟）

工序6：实现MessageClient，通过service对 MessageClient调用，实现与消息队列的交互（35分钟）

Story 1 任务拆解 – AC1

企业管理员请求账单(given) , 当上个月的账单已经生成时(when) , 会返回200(then)

- EXAMPLE 1 : 上个月没有已成功的订单 , 只有5个退改服务费时 , 账单金额应该只有100.
 - 工序1 , fake billService , 当调用 billService 的 generateBillForLastMonth 时返回{id: 1002, amount: 100} , 而 controller 返回200 , 实现被测对象 billcontroller 的对 http 请求的处理 , 得到对 billcontroller 功能的验证
 - 工序2 , stub dbClient , 当调用 dbClient 的 get 时返回 [{ id: 1000010,amount: 20}, { id: 1000011,amount: 20}, { id: 1000012,amount: 20}, { id: 1000013,amount: 20}, { id: 1000014,amount: 20}] , 实现被测对象 billService 能够正确的生成账单的能力。
 - 工序4 , 通过 spy mysql lib 来完成对 mysql 的 fake object , 当 mysql 的 query 请求时 返回 [{ id: 1000010,amount: 500}] , 实现 db client 对 mysql 的读取调用。
 - 工序2 , stub dbClient , 当调用 dbClient 的 save 时返回 {} , 实现被测对象 billService 能够正确的保存账单的能力。
 - 工序4 , 通过 spy mysql lib 来完成对 mysql 的 fake object , 当 mysql 的 save 请求时 返回 {} , 实现 db client 对 mysql 的存储调用。

Story 1 任务拆解 – AC1

企业管理员请求账单(given) , 当上个月的账单还未曾生成时(when) , 会返回200(then)

- EXAMPLE 2 : 上个月只有已成功的一个金额500的订单 , 没有退改服务费时 , 账单金额应该只有500。
 - 工序2 , stub httpClient , 当调用 httpClient 的 get 时返回 [{ id: 1000010,amount: 500}] , 实现被测对象 billService能够争取的生成账单的能力。
 - 工序4 , 通过spy axios 来完成对差旅预订系统的fake object , 当调用axios的 get请求时 返回 [{ id: 1000010,amount: 500}] , 实现http client对差旅预订系统访问调用。
- EXAMPLE 3 : 上个月即有已成功的一个金额500的订单 , 同时也有2个退改服务费时 , 账单金额应该只有540.

Story 1 任务拆解 – AC2

企业管理员请求账单(given) , 当上个月的账单已经生成时(when) , 会返回403(then)

- EXAMPLE 1 : 上个月账单已经生成 , 再次申请时 , 返回403 , 账单已存在.
 - 工序1 , fake billService , 当调用 billService 的 getBillForLastMonth 时返回{id: 1000010, amount: 200} , 而controlle返回403 , 实现被测对象 billcontroller 的对 http 请求 的处理 , 得到对 billcontroller功能的验证。
 - 工序2 , stub dbClient , 当调用 dbClient 的 get 时返回 { id: 1000010,amount: 200} , , 实现被测对象 billService能够正确的查看账单的能力。

Story 1 任务拆解 – AC3

企业管理员请求账单(given) , 当企业管理员已不是管理员时(when) , 会返回401(then)

- EXAMPLE 1 : 当一个不是管理员身份的用户申请账单时返回403.
 - 工序1 , fake userService , 当调用 userService 的 getUserRole 时返回{id: 1002, role: "employee"} , 而 controller 返回401 , 实现被测对象 billcontroller 的对 http 请求的处理 , 得到对 billcontroller 功能的验证
 - 工序2 , stub httpClient , 当调用 httpClient 的 get 时返回 [{ id: 1000010,amount: 500}] , 实现被测对象 userService 能够争取的生成账单的能力。
 - 工序4 , 通过 spy axios 来完成对登录授权系统的fake object , 当调用 axios 的 get 请求时 返回 {id: 1002, role: "employee"} , 实现 http client 对登录授权系统访问调用。

Story 2 - Story 生成发票及AC

Story:

作为企业管理员，我想要在账单结算成功之后立即申请发票，以便于企业及时完成自己的税务申报。

ACs:

- AC1: 当账单已经支付并且还未申请发票(given) , 用户申请了发票(when) , 会返回200 , 申请成功(then)
- AC2: 当账单还未支付(given) , 用户申请了发票(when) , 会返回403 , 申请失败(then)
- AC3: 当账单已经支付并且已经申请过发票(given) , 用户申请了发票(when) , 会返回403 , 申请失败(then)

Story 1 生成账单描述及AC

相关 API 定义：API 设计

Method: POST

Endpoint: /bills/{:bid}/invoice

Success:

* 成功 code 200

Code: 200

Response Body:

{

“id”: 100012,

“number”: “FP202112”

“amount”: 120000

”link”: “<https://invoice.travel.com/100012>”

}

Failure:

- 发票已生成, code: 403
{ code: 403,
 message: ‘发票已存在’}
- 数据库连接失败 503
{ code: 503,
 message: ‘数据库连接失败’}
- 调用发票开具系统失败 503
{ code: 503,
 message: ‘发票开具系统访问失败’}
- 消息队列连接失败 503
{ code: 503,
 message: ‘消息队列连接失败’}

Story 2 - Story 生成发票及AC

相关 API 定义：外部API

发票申请MQ event:

```
{  
    bill_id: 10001001,      账单id  
    amount: "230000.00"    账单金额  
    client_id: 100020     企业编号  
}
```

发票数据库创建脚本：

```
CREATE TABLE `travel`.`invoice` (`id` int NOT NULL  
AUTO_INCREMENT, `bill_id` int NOT NULL, `amount` DECIMAL(10,2), `link`  
VARCHAR(600), `comments` VARCHAR(400), PRIMARY KEY (id));
```

Story 2 - 任务拆解 – AC1

**当账单已经支付并且还未申请发票(given) , 用户申请了发票(when) , 会返回200 , 申请成功
(then)**

- EXAMPLE 1 : 对一个未曾开具过发票的已支付账单 , 进行发票申请 , 则返回 200
 - 工序1 , fake invoiceService , 当调用 invoiceService 的 generateInvoice 时返回{} , 而 invoiceController 返回200 , 实现被测对象 invoiceController 的对 http 请求 的处理 , 得到对 invoiceController 功能的验证
 - 工序2 , stub dbClient , 当调用 dbClient 的 get 时返回 { id: 1000010,amount: 20, link: "https://asdfasd.com/invoice/111"} , 实现被测对象 invoiceService 能够正确的申请发票的能力。

Story 2 - 任务拆解 – AC2

当账单还未支付(given) , 用户申请了发票(when) , 会返回403 , 申请失败(then)

- EXAMPLE 1 : 对一个未支付的账单 , 进行发票申请 , 则返回 403 , 申请失败。
 - 工序1 , fake billService , 当调用 billService 的 getBill 时返回{..., paymentStatue: 'false'} , 而 invoiceController 返回403 , 实现被测对象 invoiceController 的对 http 请求 的处理 , 得到对 invoiceController 功能的验证

Story 2 - 任务拆解 – AC2

当账单已经支付并且已经申请过发票(given) , 用户申请了发票(when) , 会返回403 , 申请失败 (then)

- EXAMPLE 1 : 对一个未曾开具过发票的已支付账单 , 进行发票申请 , 则返回 200
 - 工序1 , fake invoiceService , 当调用 invoiceService 的 getInvoice 时返回{id:10011} , 而 invoiceController 返回403。实现被测对象 invoiceController 的对 http 请求 的处理 , 得到对 invoiceController 功能的验证

效能管理
带领团队落实

/thoughtworks

代码结构展示

代码结构/目录截图

主要业务代码在 src 文件夹中，其中：

Clients 表示client层，负责跟外部系统交互

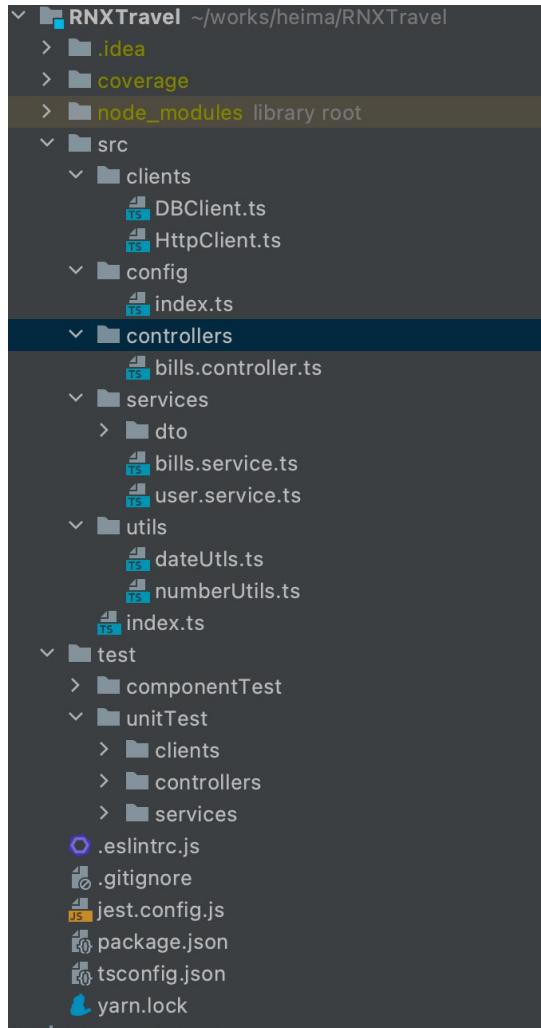
Service表示service层，包含DTO，负责主要的业务逻辑实现。

Controller表示controller层，负责对http的请求和返回值的生成。

主要测试代码在 test 文件夹中，其中：

UnitTest 表示单元测试，负责每个业务/系统元素的测试

ComponentTest 表示部分组件测试，负责元素之间的集成测试。



Q & A

马建勋
AU

/thoughtworks