爱旅行-系统架构设计及部署策略分析

爱旅行项目是一个综合性的旅游电商平台，并具备跨平台性。那么基于PC、移动端（Andriod、Ios）的考虑，以及该项目的非功能性需求（详见《爱旅行系统需求分析说明书.doc》）:安全性、可靠性、互操作性、健壮性、易用性、可维护性、可移植性、可重用性、可扩充性等等都是基于性能、安全等架构目标的，因此爱旅行项目系统架构的设计目标有以下7个：

1. 可靠性（reliable）
2. 安全性（secure）
3. 可扩展性（scalable/extension）
4. 可定制化（customizable）
5. 可维护性（maintainable）
6. 客户体验（customer experience）
7. 市场时机（time to market）

## 系统架构设计

爱旅行系统的整体技术架构如图1所示：

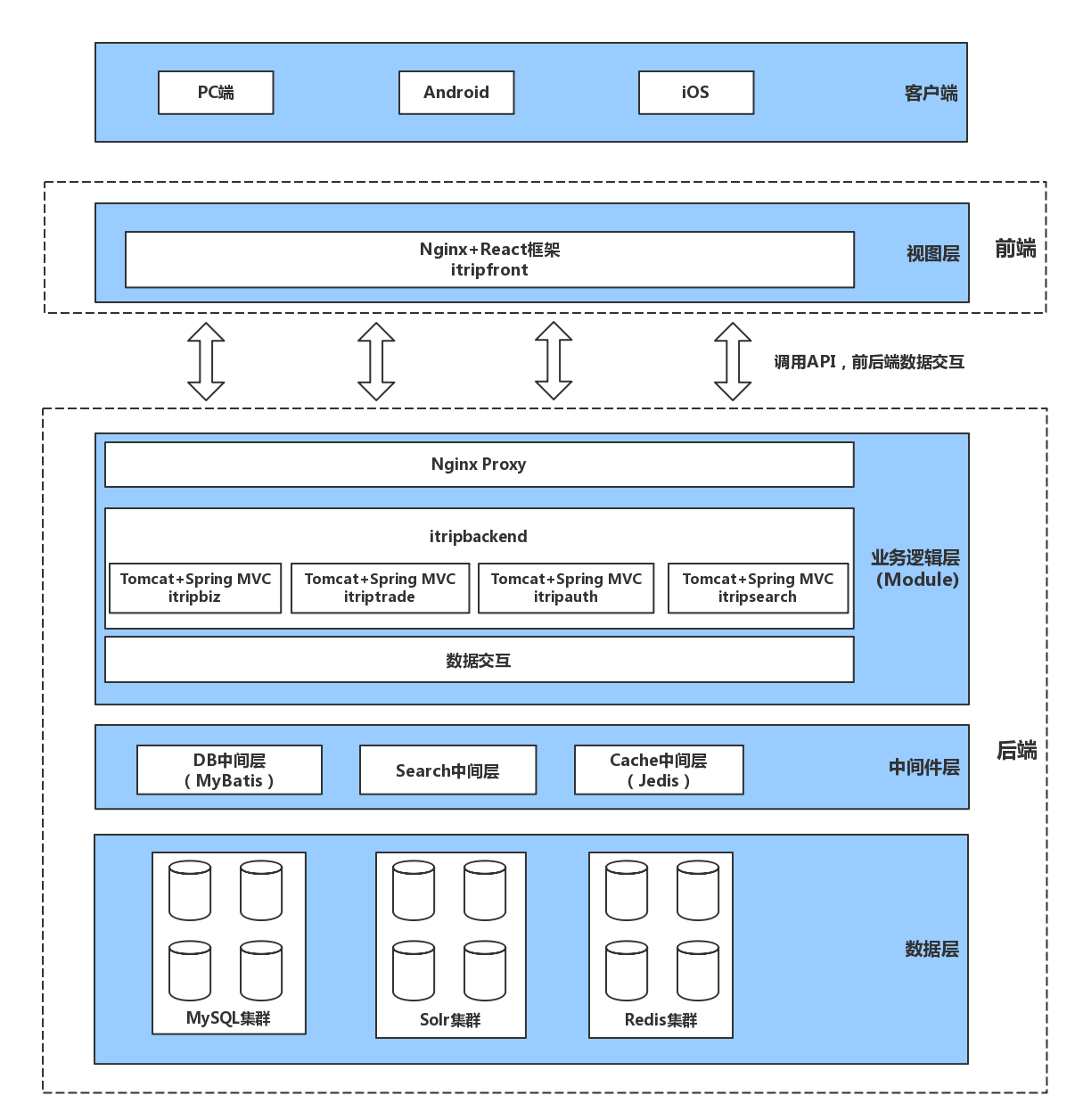


图1：爱旅行系统架构图

爱旅行架构从两个方向进行设计：前端架构和后端架构，将前端和后端完全分离，并采用分布式集群的架构设计，通过定义API，与前端进行数据交互，前端使用html进行数据展现，并可加入移动端（Andriod、Ios）实现，直接调用API即可。

前端架构采用React JavaScript 库实现组件化开发，并通过Nginx反向代理到后端项目，实现API的调用，前后端数据交互。

后端架构采用模块化开发，采用Maven进行多Module管理。根据业务进行模块划分，itripbackend分为以下四个Web模块：

1. 主业务模块（itripbiz）：包括酒店业务、旅游业务、机票业务、攻略业务
2. 搜索模块（itripsearch）：爱旅行项目中所有搜索功能（solr）
3. 认证模块（itripauth）：负责用户身份验证，生成并维护Token
4. 支付模块（itrptrade）：包括支付宝支付、微信支付

以及为Web模块提供服务的三个基础模块：itripdao、itriputils、itripbeans。

每个web模块均独立部署在单独的web服务器上，并通过Nginx反向代理到相应的tomcat服务器上：

upstream itripbiz\_server

{

server IP:port; # itripbiz模块部署的服务器IP及端口号

server IP:port backup; # itripbiz模块部署的服务器IP及端口号（备份服务器）

}

upstream itripsearch\_server

{

server IP:port; # itripsearch模块部署的服务器IP及端口号

}

upstream itripauth \_server

{

server IP:port; # itripauth模块部署的服务器IP及端口号

}

upstream itriptrade\_server

{

server IP:port; # itriptrade模块部署的服务器IP及端口号

}

### 1.1.1 后端架构

后端架构（iTripBackend）采用： ORM+cache+Spring Core+SpringMVC，如下图1所示：

为了提高代码复用性及可维护性，采用Maven进行多Module管理，进行模块化架构设计

### 1.1.4 模块化开发

iTripBackend分为以下四个Web模块：

1. 主业务模块（itripbiz）：包括酒店业务、旅游业务、机票业务、攻略业务
2. 搜索模块（itripsearch）：爱旅行项目中所有搜索功能（solr）
3. 认证模块（itripauth）：负责用户身份验证，生成并维护Token
4. 支付模块（itrptrade）：包括支付宝支付、微信支付

对于上述Web模块，都存在一些公用的部分（比如：bean、dao、utils），为提高代码复用性及可维护性，故采用Maven进行多Module管理，一共7个Module(itripbiz、itripsearch、itripauth、itriptrade、itripdao、itriputils、itripbeans)，具体Module结构及atifactId如图1、2所示：

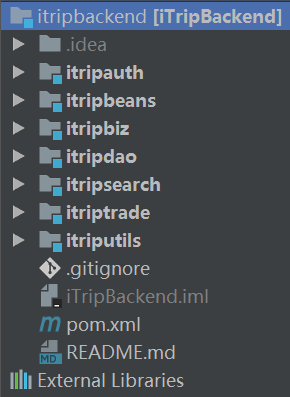
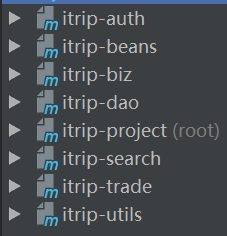
 

图1 iTripBackend-模块结构 图2 artifactId

注意：各模块之间存在依赖，切不可存在相互依赖。依赖关系如下：

1. itripdao模块depedency：itrip-beans、itrip-utils
2. itriputils模块depedency：itrip-beans
3. itripbiz模块depedency：itrip-dao、itrip-beans、itrip-utils
4. itripsearch模块depedency：itrip-dao、itrip-beans、itrip-utils
5. itriptrade模块depedency：itrip-dao、itrip-utils
6. itripauth模块depedency：itrip-dao、itrip-utils、itrip-beans

各模块具体说明如下：

1. itripbeans：包括dto、pojo、vo

注：该模块需要打成jar包，供其他Web模块使用

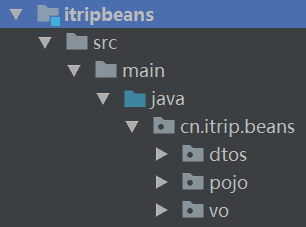


图3 itripbeans

1. itripdao：包括dao层所有代码

注：该模块需要打成jar包，供其他Web模块使用

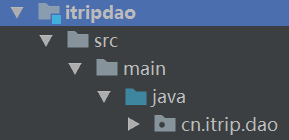


图4 itripdao

1. itriputils：包括common包下所有工具类

注：该模块需要打成jar包，供其他Web模块使用

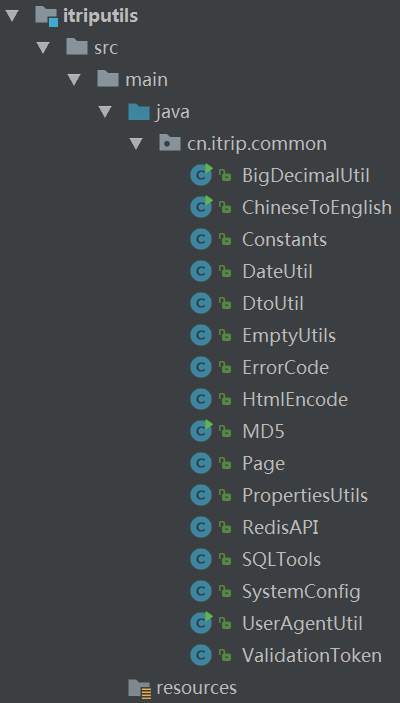


图5 itriputils

1. itripbiz：包括controller、service层以及resources、webapp

注：该web模块需要打成war包，部署路径：/biz

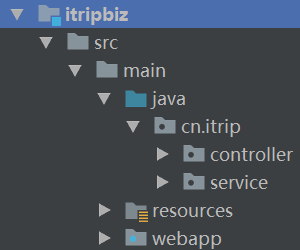


图6 itripbiz

1. itripsearch：包括controller、service、dao层以及resources、webapp

注：该web模块需要打成war包，部署路径：/ search

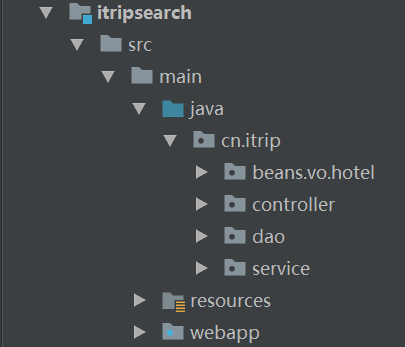


图7 itripsearch

1. itripauth：包括controller、service层以及resources、webapp

注：该web模块需要打成war包，部署路径：/ auth

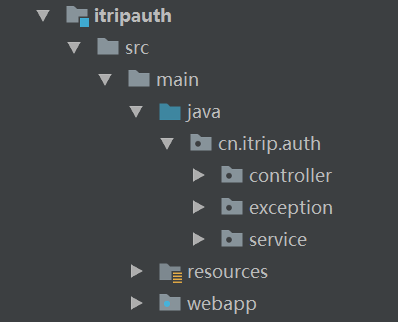


图8 itripauth

1. itriptrade：包括controller、service层以及resources、webapp

注：该web模块需要打成war包，部署路径：/ itrade

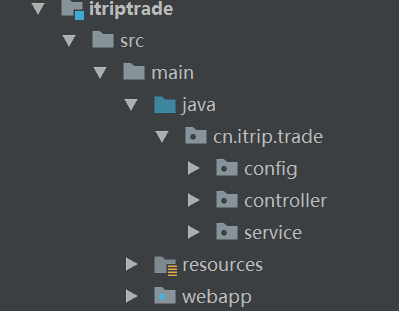


图9 itriptrade

基于以上多模块开发，需要遵守以下的代码约定：

#### 1.1.4.1 itripbeans模块

1. pojo：代码自动生成
2. dto：数据传输对象，包括输出前端Dto以及接收前端传入的InputDto，数据结构如下：

1> Dto .java

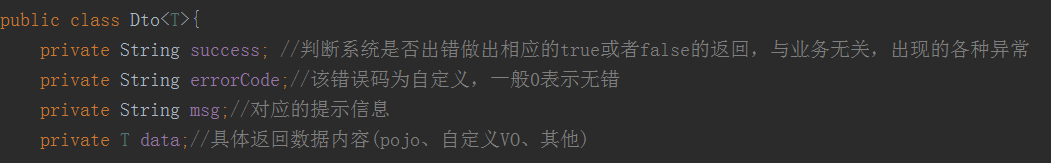


图10 dto

根据service返回的结果，Controller层的API定义往前端输出的数据内容（json格式），e.g：

图11正确返回 图12 错误返回

注意：错误码errorCode根据业务定义

|  |  |
| --- | --- |
| **业务模块** | **模块规则** |
| itripbiz | 1开头（100000） |
| itripsearch | 2开头（200000） |
| itripauth | 3开头（300000） |
| itrptrade | 4开头（400000） |

2> InputDto.java

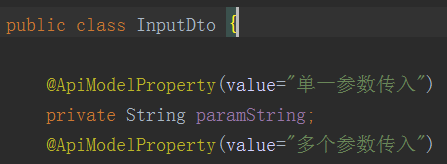


图11 Inputdto

1. vo：根据业务功能创建各自VO，并根据业务分包放置，如图12所示：

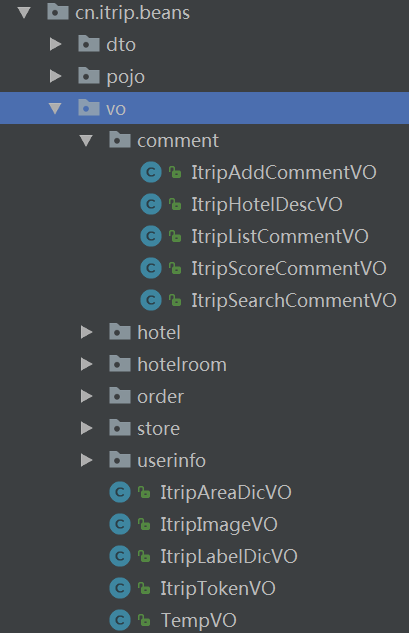


图12 vo

注意：VO是前后端参数传递的接收对象，需要加入Swagger的@ApiModel，进行属性的详细说明及输入要求。

#### 1.1.4.2 itripdao模块

该模块的初始化代码（普通的增、删、改、查）均是自动生成，若因业务需求，则需新增接口方法。

#### 1.1.4.3 itriputils模块

所有的工具类都放置在此模块内，注意：不能使用注解，以降低耦合度。

#### 1.1.4.4 itripbiz模块

1. service层

1> 均需throws Exception，以便事务控制。

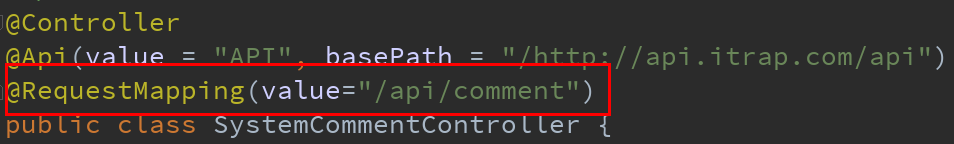
2> 若业务需进行事务处理，则需在接口方法名加上前缀itriptx，比如：

public Integer itriptxModifyItripHotelOrder(ItripHotelOrder itripHotelOrder)throws Exception;

1. controller层

1> 提供API接口，各web工程都需要加入Swagger，以便给前端生成API文档

2> Controller必须加统一前缀：/api/\*\*\*，如图13所示：



注：其他模块（itripsearch、itriptrade、itripauth）与该模块规范一致，此处不再赘述。

## 1.2 系统部署架构设计

1、【强制】前端的健壮性开发，需要达到：在无API的情况下，前端一样能够完美奔跑

2、【推荐】前端采用mock开发