

AsialInfo

云梯开发框架（ODF）技术培训

Open Development Framework

英林海
亚信科技

目录

一

背景

二

总体架构

三

主体功能介绍

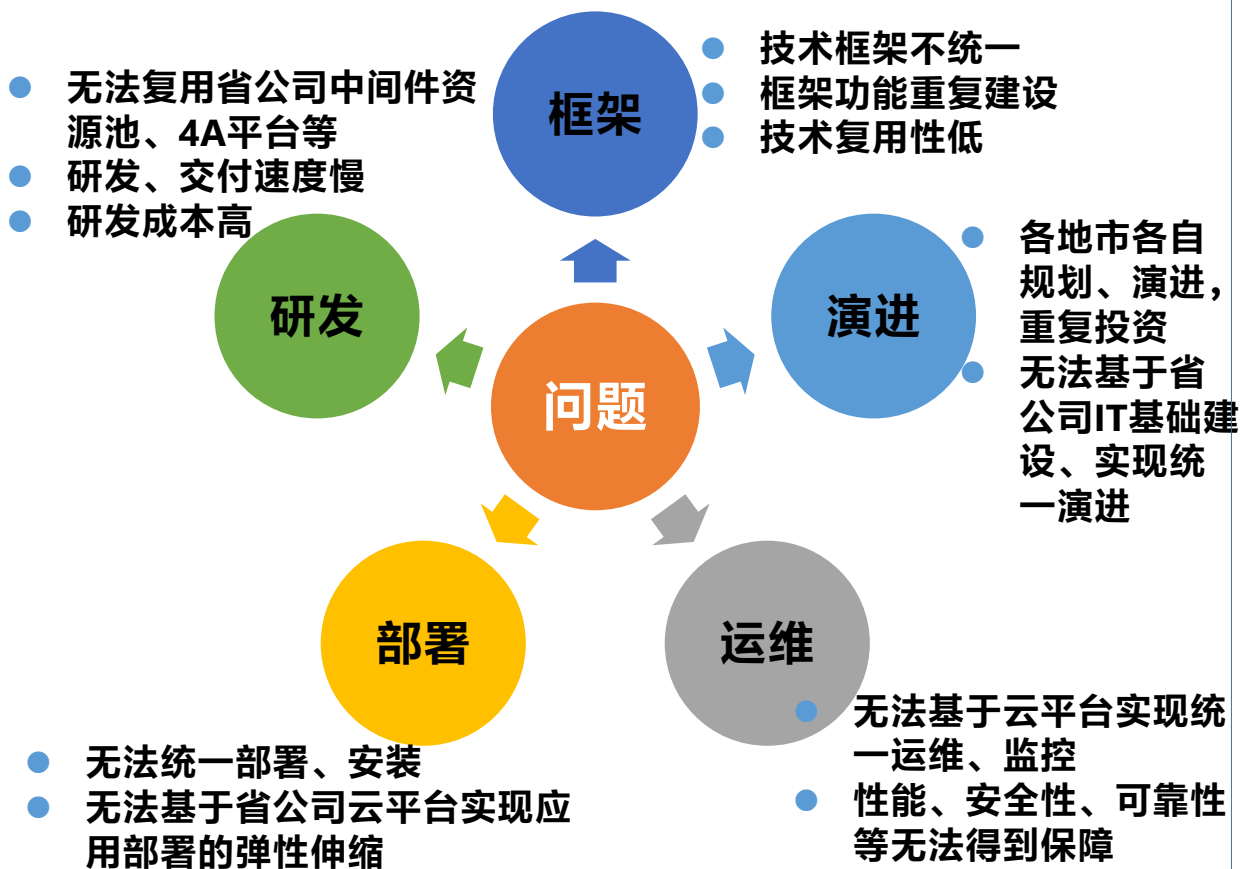
四

开发流程及demo

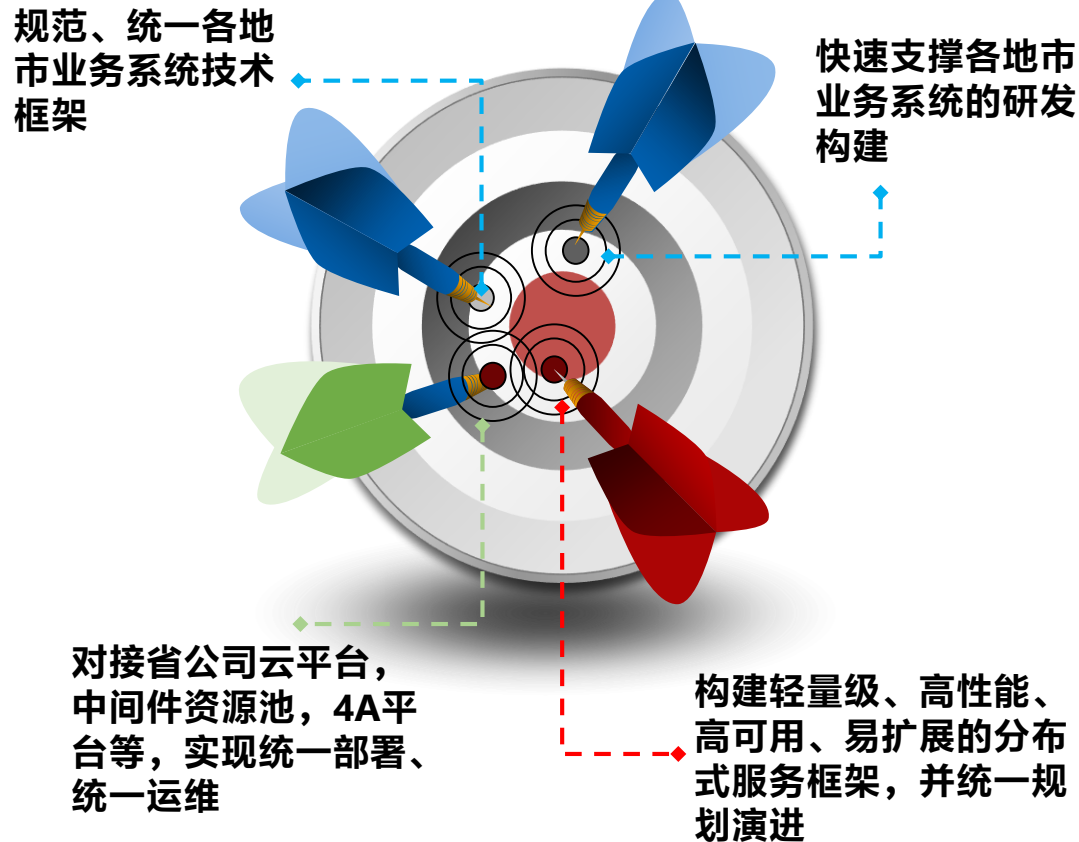
背景

浙江移动各地市公司业务系统建设研发框架当前存在技术不统一、无法复用省公司云平台、中间件资源池、4A平台等IT基础设施，地市公司期望省公司采用开源技术框架组织建设一套统一的分布式服务框架，解决地市业务建设遇到的问题。

存在问题



建设目标



云梯开发框架技术选型



Spring MVC
Spring

Spring Web MVC是一种基于Java的实现了Web MVC设计模式的请求驱动类型的轻量级Web框架。



Ribbon
Netflix

提供云端负载均衡，有多种负载均衡策略可供选择，可配合服务发现和断路器使用。



Eureka
Netflix

云端服务发现，一个基于 REST 的服务，用于定位服务，以实现云端中间层服务发现和故障转移。



Hystrix
Netflix

熔断器，容错管理工具，旨在通过熔断机制控制服务和第三方库的节点,从而对延迟和故障提供更强大的容错能力。



Spring Boot
Pivotal

Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。

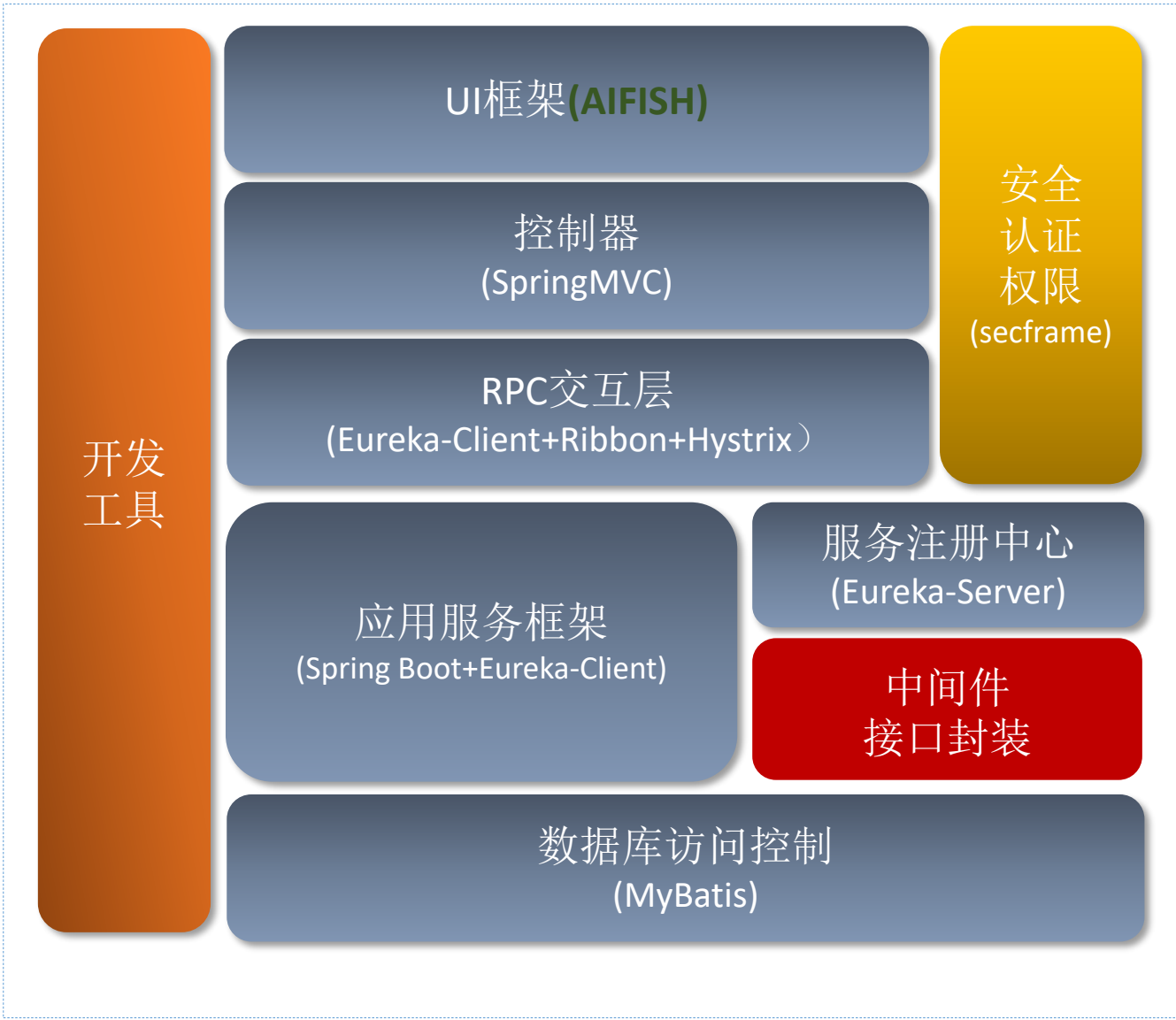


MyBatis 是支持定制化 SQL、存储过程以及高级映射的优秀的持久层框架。



AIFISH是由亚信前端技术团队基于开源框架及工具集整合的Web组件库+前端技术栈，整理出的一套轻量级、模块化的前端解决方案，用以更高效地进行前端开发。

云梯开发框架基础功能模块



■ UI视图:

亚信AIFISH由亚信前端技术团队基于开源框架及工具集整合的Web组件库，页面布局、典型模板，样式风格包，登录首页和页面框架等；

■ 控制器:

- 过滤器中增加针对性的安全访问控制、认证、权限过滤
- 会话管理等能力的扩展和增强；
- 提供请求URL、报文加解密、编解码的支持。

■ RPC交互层:

- 对请求报文做扩展，增加业务公共属性的定义；
- 增加对服务调用的鉴权和流量控制；
- 修改序列化方式，采用效率较高的方式如thrift等。

■ 服务层:

- 封装服务调用链路、异常处理、数据源访问、拦截器等接口；

■ 服务注册中心:

- 增加对资源监控、状态监听、服务目录、实例目录的支持；

■ 数据库访问层:

- 增加针对sql性能的监控、序列获取API封装、最大结果集控制；

■ 中间件接口封装:

- 封装对接4A、日志、消息、缓存、监控等中间件的相关API；

■ 安全认证权限组件:

- 引入secframe组件用于操作员账号、权限配置等能力的支持；

■ 开发工具:

- 补充业务开发辅助工具，辅助javabean、服务接口、mapper接口等基础文件及代码的生成和数据源配置、系统参数配置等。

云梯开发框架的核心价值



- 引入AIFISH的前端页面框架,基于当下最热门的前端UI框架Bootstrap,提供大量组件,完成组件视觉交互的标准化及规范化产出,同时提高了开发效率,并便于UI异常定位解决;
- 通过对控制器层、RPC交互层以及应用层的底层API的扩展和封装,提高了框架的安全性、可靠性和性能,同时屏蔽了底层技术变化对业务的影响,提高了业务应用开发效率;
- 引入了中间件接口封装层,便于地市业务方便快捷无缝对接4A、缓存、日志、消息、监控等中间件系统;
- 通过引入secframe安全、认证、权限管理组件,提供了统一安全的加解密算法,防SQL注入等安全保障;统一的认证、鉴权接口;统一的权限、组织管理支持;
- 通过提供基于Eclipse插件的辅助开发工具和web化的系统管理界面,将通用的文件、代码、配置一键生成,简化了开发,提高了开发效率。

目录

一

背景

二

总体架构

三

主体功能介绍

四

开发流程及demo

云梯开发框架总体功能架构

原始功能

二次开发

新研发

Web层

UI视图层

页面布局

页面组件

页面样式

页面模板

登录页面

页面框架

同步交互

异步加载

控制器层

编码解码

加密解密

数据访问域

认证

鉴权

访问控制

会话管理

视图解析

RPC交互层

传输协议

数据协议

连接池管理

超时监控

序列化

心跳检测

负载均衡

服务发现

服务路由

认证权限管理

账号管理

认证鉴权

权限管理

菜单管理

组织管理

岗位管理

角色管理

系统管理

服务层

应用服务层

服务监听

反序列化

数据解析

服务拦截器

服务调用

服务注册

服务发布

线程数据访问域

基础数据缓存

异常处理

中间件接口封装层

4A SDK

日志SDK

监控SDK

缓存SDK

消息SDK

其它

服务注册中心

服务注册

服务发现

状态监听

资源监控

服务目录

实例目录

数据访问层

持久层框架

数据库连接池

JDBC驱动

Oracle

MySQL

其它

辅助开发工具

Bean自动生成

服务接口自动生成

Mybatis 接口生成

Mybatis配置修改

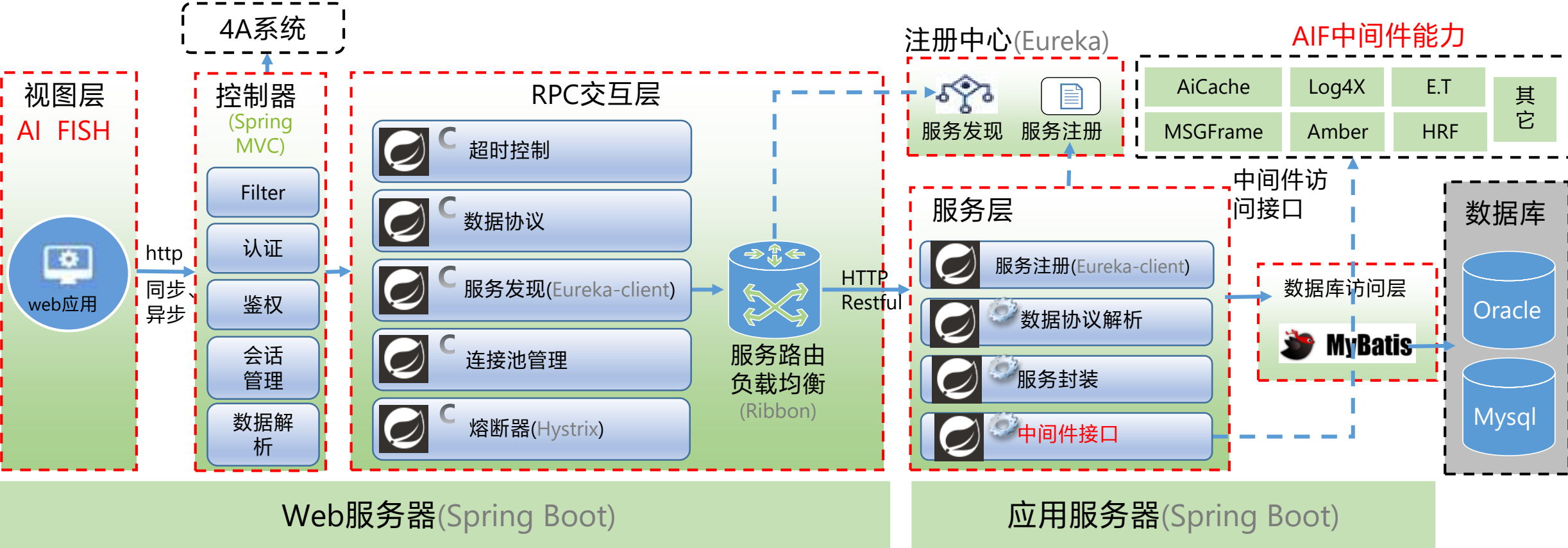
简单SQL自动生成

服务自动注册

Javadoc注解生成

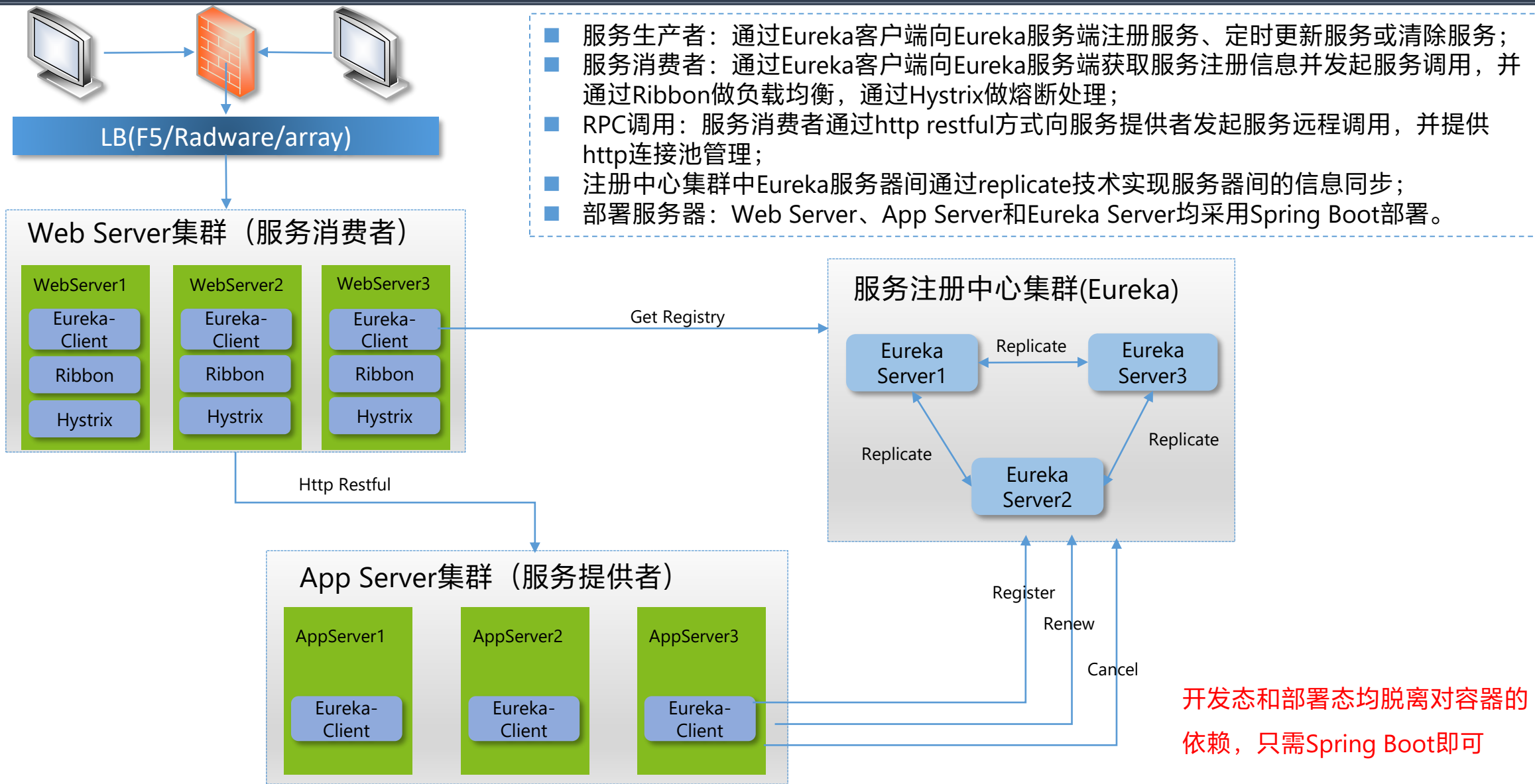
其它

云梯开发框架技术架构

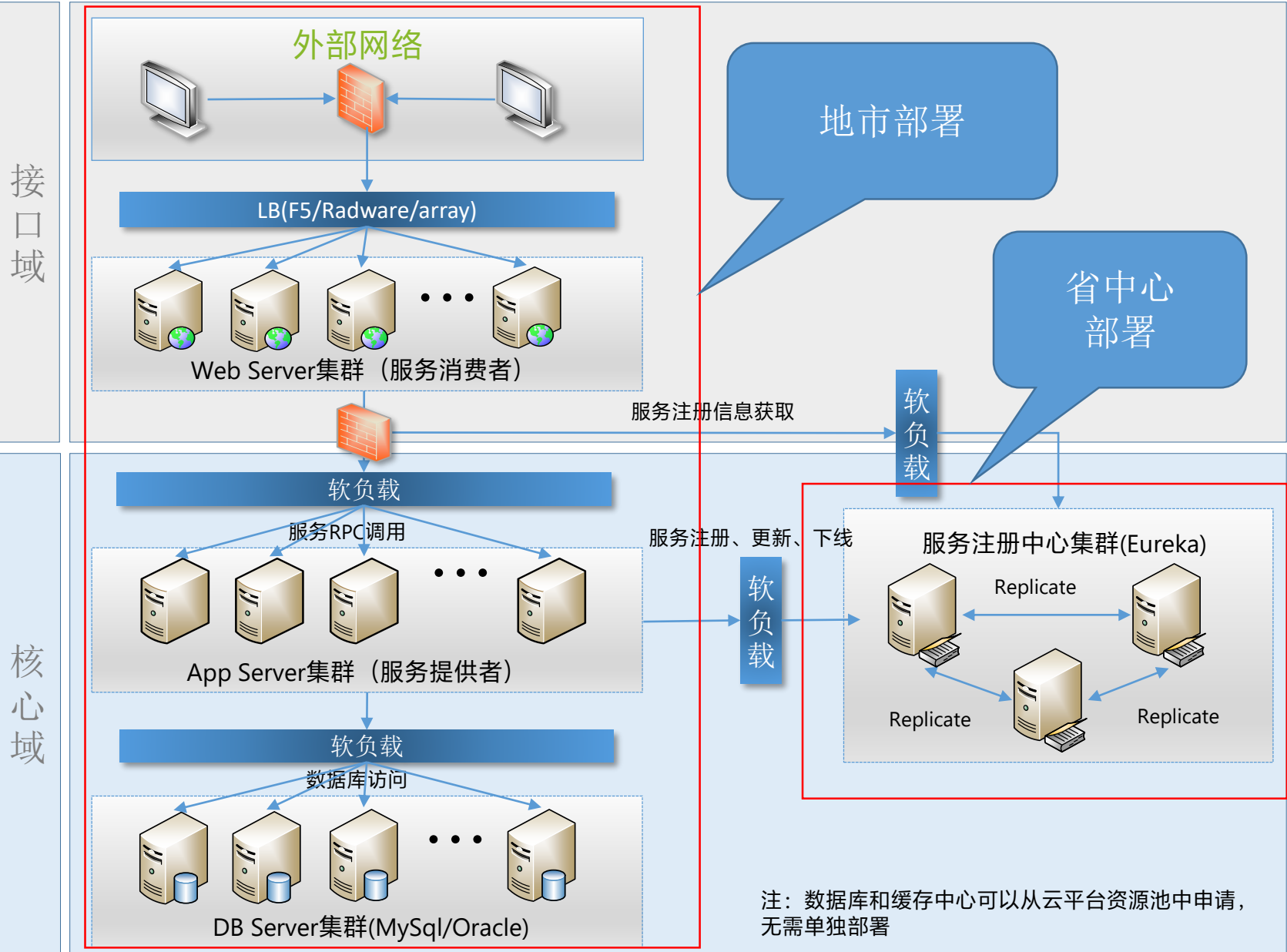


- 视图层AIFISH：采用亚信AIFISH框架，实现页面布局、控件、表单、页面样式、模板化等功能封装、并支持http请求的异步，同步交互
- 控制器：采用开源Spring MVC 实现控制器层的请求响应，并集成亚信USPA框架实现账号认证、鉴权，会话管理等功能
- RPC交互层：采用SpringCloud的Ribbon, Eureka, HyStrix，实现传输协议、连接池管理，超时控制、负载均衡、服务发现、路由、服务熔断等
- 服务层：采用SpringBoot，实现应用服务器功能、服务注册、数据协议解析、服务封装、底层开源组件接口封装等功能
- 数据访问层：采用开源Mybatis，基于数据库连接池DBCP实现持久层访问功能，同时提供MyBatis访问接口生成和配置修改工具
- 中间件接口访问层：集成AIF中间件访问的客户端SDK，提供统一的API访问接口，实现对相应中间件的访问

基于云梯开发框架的应用部署逻辑架构

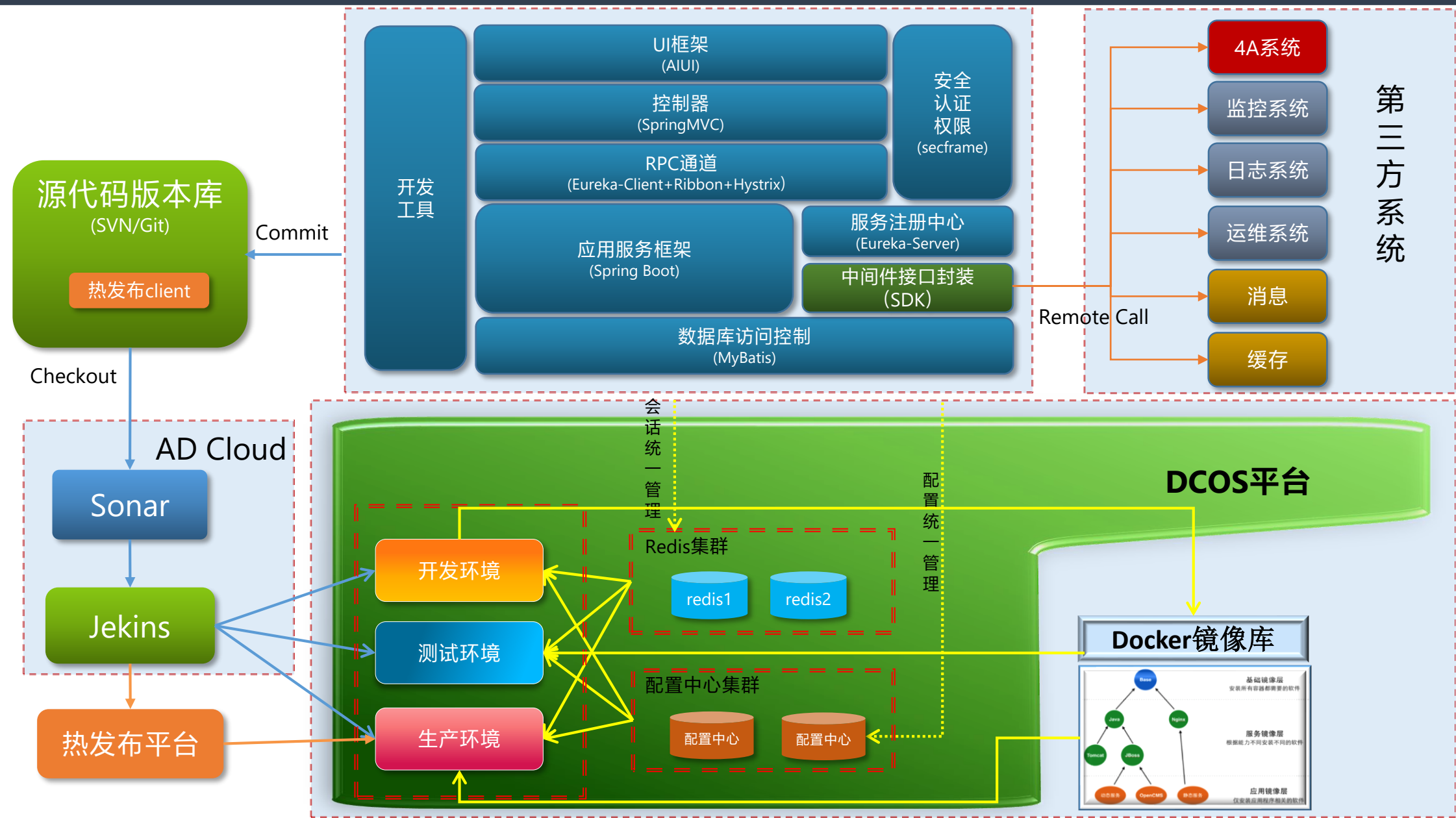


基于云梯开发框架的应用部署物理架构



- AppServer: 部署后端src源码、业务配置文件、数据库访问相关配置文件，通过Eureka客户端向Eureka服务端注册服务、定时更新服务或清除服务；
- WebServer: 部署UI前端html、js、css、图标文件和控制器层相关src文件和配置文件，通过Eureka客户端向Eureka服务端获取服务注册信息并发起服务调用，并通过Ribbon做负载均衡，通过Hystrix做熔断处理；
- RPC调用: 服务消费者通过http restful方式向服务提供者发起服务远程调用，并提供http连接池管理；
- 注册中心: 集群中Eureka服务器间通过replicate技术实现服务器间的信息同步，由省中心集中部署，提供访问url给地市使用；
- 部署服务器: Web Server、App Server和注册中心server均采用Spring Boot部署。
- 数据库、缓存、日志、消息等中间件系统，可由中间件资源池提供资源，无需单独部署。

云梯开发框架与其他系统边界



目录

一

背景

二

总体架构

三

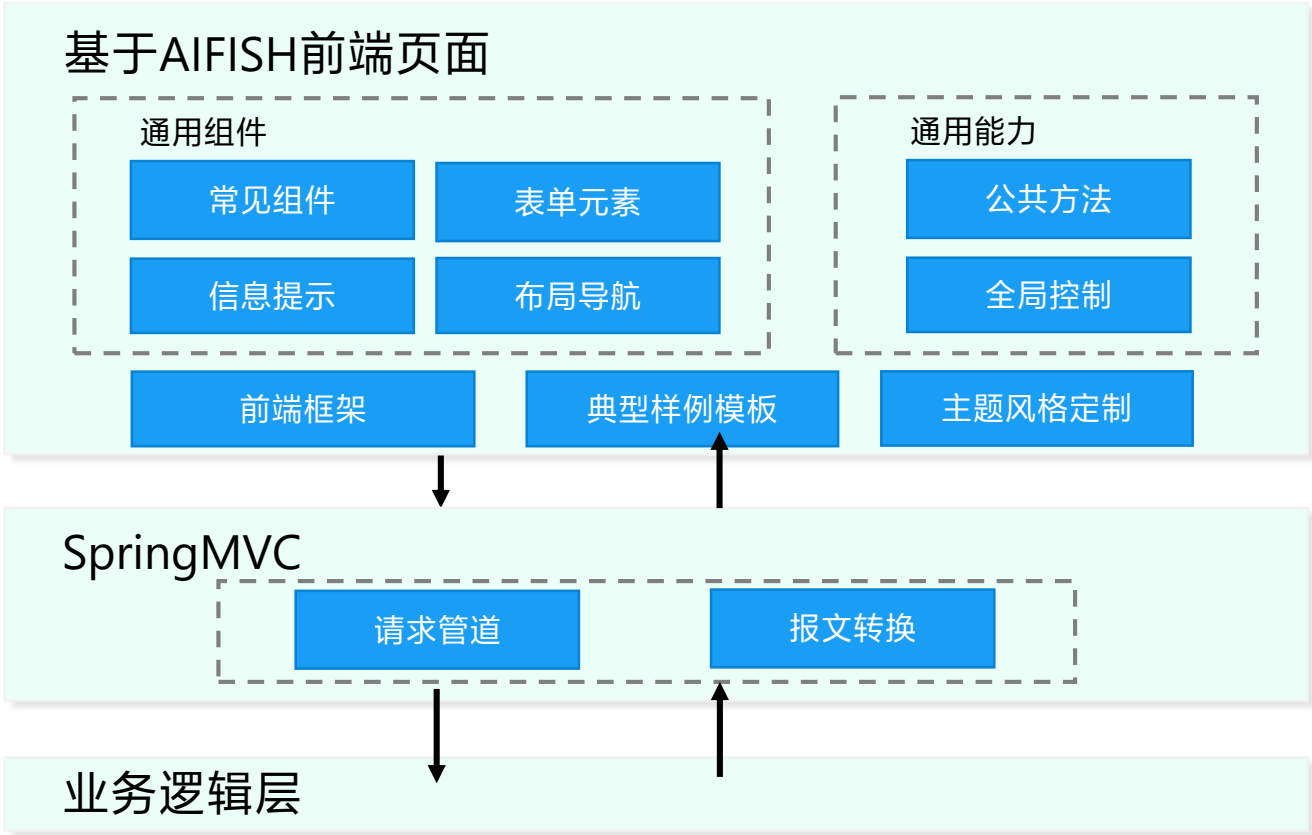
主体功能介绍

四

开发流程及demo

主体功能1—AIFISH框架

- AIFISH基于当下最热门的前端UI框架Bootstrap，并优化整合AdminLTE，提供至少50+组件，完成组件视觉交互的标准化及规范化产出。
- AIFISH基于 jQuery + Bootstrap + Handlebars + Seajs 基础技术栈，利用模板引擎，实现业务组件，提高开发效率。
- AIFISH采用前后端分离的模式，通过层次解耦实现前后端并行开发，页面和数据请求分离，开发人员不需关注后端服务逻辑，提高开发效率。
- AIFISH与SpringMVC无缝对接，最大化适配SpringMVC的Controller能力。



主体功能1— AIFISH通用组件

常见组件

- 上传 Upload
- 树 Tree
- 分页 Pagination
- 菜单 Menu
- 表单 Form
- 表格 Table
- 下拉菜单
Dropdown
- 按钮 Button
- 图标 Icon
- 栅格 Grid
- ...

表单元素

- 下拉框 Combobox
- 单选框 Radio
- 备注框 Textarea
- 复选框 Checkbox
- 弹出框 Textpopup
- 数值框 Number
- 日历框 Datepicker
- 开关框 Switch
- 输入框 Input
- 滑动框 Slider
- ...

信息提示

- 对话框 Dialog
- 提示框 Tips
- 文字提示 Tooltip
- 气泡确认
Popconfirm
- 通知提醒
Notification
- 加载条 Loading
- 进度条Nprogress
- 按钮加载ladda
- 增强弹框popover
- ...

布局导航

- 布局 Layout
- 图表 Chart
- 面板 Panel
- 页签 Tab
- 步骤条 Steps
- 面包屑 Breadcrumb
- 可伸缩布局flexible
- 单页滚屏 fullpage
- 书本翻页 turn
- 页面引导 intro
- ...

50+ 丰富页面组件，覆盖业务常用场景

主体功能1—页面组件模板快速构建页面

文本

单选下拉

日期

时间

单选框

复选框

树选择框

描述

查看源码

```
require(["form"], function() {
```

属性

方法

属性

方法

提供在线帮助文档及模板代码，方便页面快速设计实现并在demo中会有导航：
<http://alsiso.gitee.io/aifish/views/introduce.html>

| 方法 | 参数 | 说明 |
|--------|-------|--|
| width | width | 设置组件整体宽度，width可以是百分比,可以是数字 |
| layout | type | 设置表单布局方式，支持格式“col-n” |
| titleV | width | 设置title宽度，itleWidth可以是百分比,可以是数字 |
| size | param | 设置输入框尺寸，定义了三种尺寸（“L”、“M”（默认）、“S”），高度分别为 40px、32px 和 24px |
| data | data | 刷新form数据 |
| loadF | fn | 数据转换 |
| colun | ds | 设置表单元素 |
| onRei | flag | 获取表单数据，当flag为true时，直接取所有的表单元素的数据，否则取\$.extend({}, this._D, data) |
| afterF | name | 获取指定表单元素的实例 |

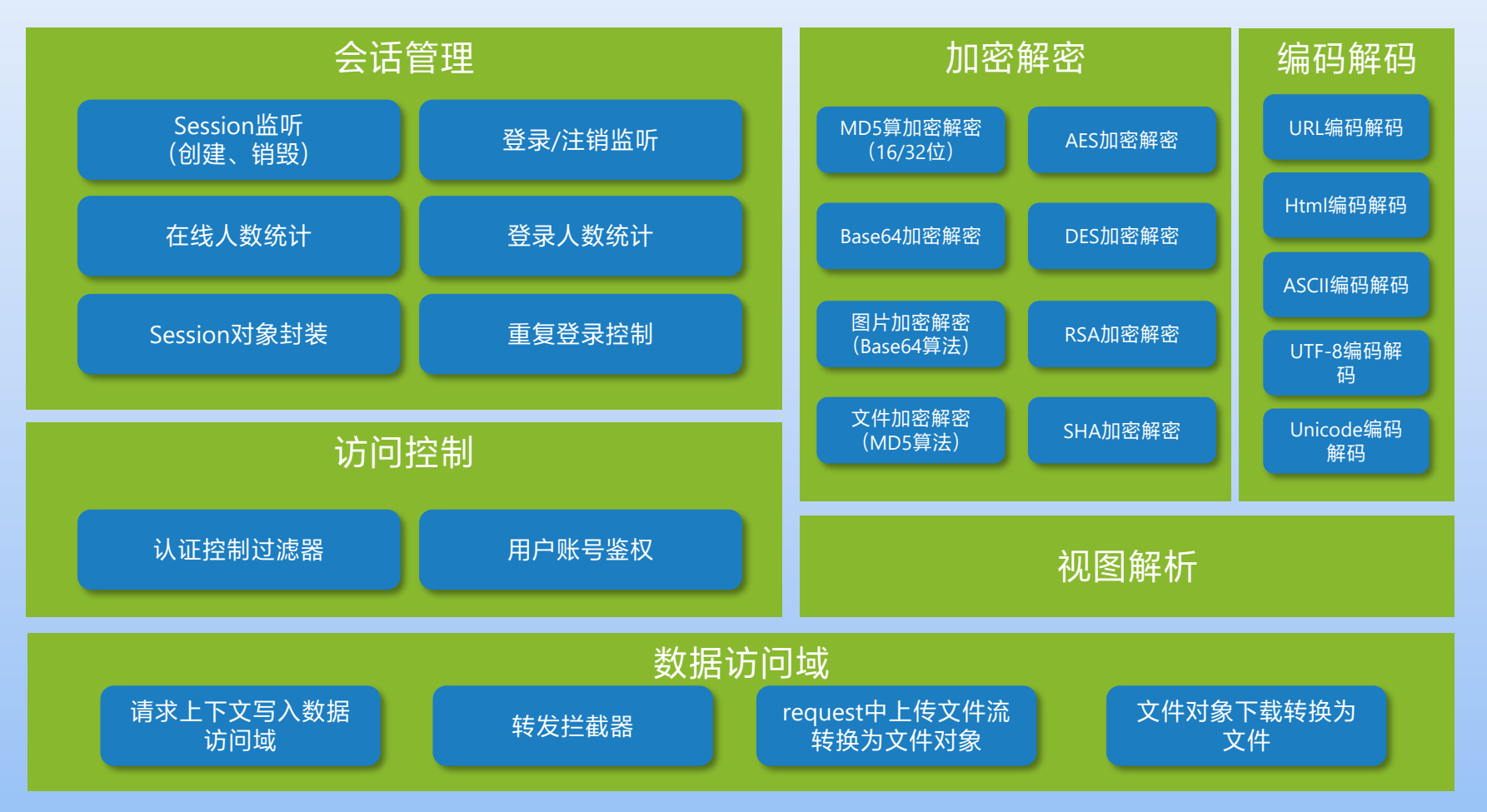
主体功能1—页面框架快速实现页面集成

- AIFISH框架提供默认的登录页面和系统页面框架，包含左侧的菜单栏和上侧的功能区域；登录首页有使用说明的介绍，可根据引导替换形成自身风格的登录页和首页。
- 业务开发只需使用页面组件组装设计页面后，通过权限管理将菜单权限配置上去，通过登录即可呈现页面开发的效果。



主体功能2—控制层

- 控制器是前端界面与后端服务交互时，解析用户输入，并将其转换为合理的模型数据，并通过视图（view）展示给用户；
- 同时在前后端交互的过程中做安全访问控制、会话管理、数据的加解密、编解码处理等；



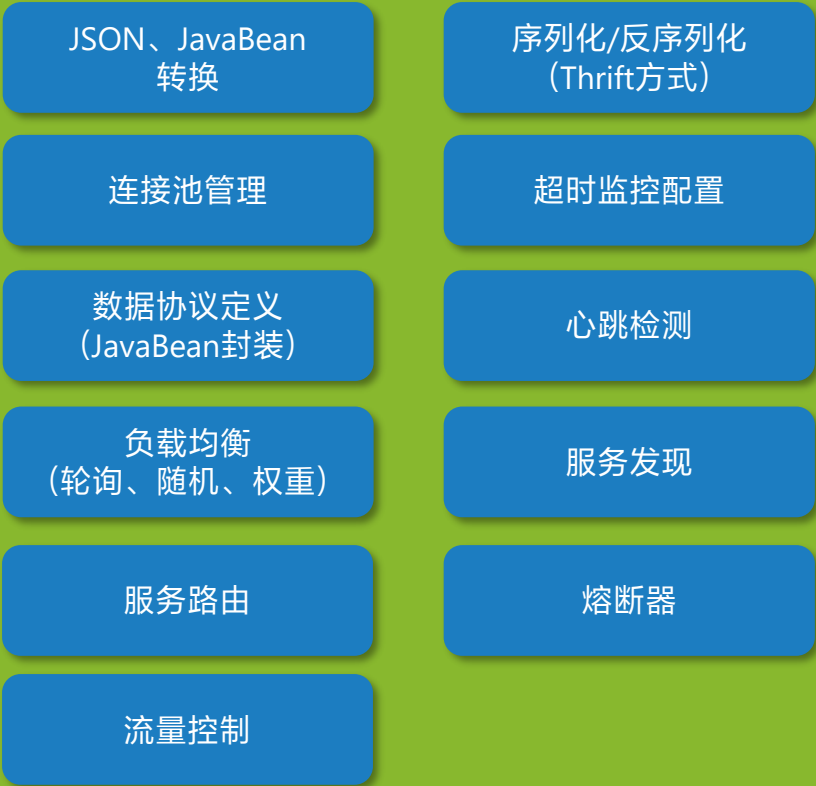
- 会话管理：提供会话相关的管理功能和封装；
- 访问控制：提供过认证过滤、账号鉴权等功能；
- 数据访问域：提供请求上下文数据的读写、上传下载文件对象转换等功能；
- 视图解析：采用SpringMVC开源框架功能用于页面视图渲染、跳转；
- 编码解码：提供常用的编码解码工具类供业务开发使用；
- 加密解密：提供常用的加密解密工具类供业务开发使用。

通过对控制器层的封装，业务开发无需关注会话、安全、访问控制等基础功能，框架并提供了常规的工具类，方便业务快速开发。

主体功能3—RPC交付层

■ RPC交互层在云梯开发框架中起到前后台数据传递的关键桥梁作用。

RPC客户端



- JSON、JavaBean转换：将前端页面提交的JSON报文转换成公共的JavaBean对象，将服务调用返回的JavaBean转换成前端页面识别的JSON报文；
- 序列化/反序列化：对于前端页面提交的数据转换成的JavaBean需要序列化后通过网络传输到服务端，对于服务调用返回的JavaBean二进制流也要反序列化，目前采用Thrift方式；
- 连接池管理：RPC请求采用Http协议，需要频繁建立连接和释放连接，采用池化处理来提高数据访问性能，并支持连接池数量的可配置；
- 超时监控配置：配置超时时间，配合熔断器做http请求超时和响应超时控制；
- 数据协议定义：定义一个公共的javaBean对象，用于前后台数据传递；
- 心跳监测：提供一个常驻进程，RPC客户端定时监测服务端服务器状态；
- 负载均衡：采用开源Ribbon组件提供的能力，可采用轮询、随机、权重方式，支持可配置；
- 服务发现和服务路由：采用开源的Ribbon和Eureka提供的能力；
- 熔断器：采用开源的HyStrix提供的能力。
- 流量控制：对服务请求进行流量控制，避免过载。

通过对PRC层的封装，将开源技术组件做了相应的整合，通过连接池管理和序列化算法的扩展，提高了PRC通信效率及数据的转换和传输效率；集成超时控制、心跳监测、负载均衡、熔断器、流量控制，大大提高了系统的可靠性。

主体功能4— Server层

■ Server层是云梯开发框架的核心，主要用于处理服务调用相关的逻辑。

服务端

服务监听

服务注册

服务发布

序列化/反序列化
(Thrift方式)

数据解析

服务拦截器

访问线程域

服务调用

基础数据缓存

异常处理

服务日志查看

- 服务监听：提供监控管理页面，用于服务状态监控，服务器状态监控；
- 服务注册、服务发布：采用开源Eureka提供的能力；
- 序列化/反序列化：对于前端请求通过RPC传递过来的数据需要反序列化转换成javaBean，对于服务返回的数据要通过反序列化后返回前端，目前采用Thrift方式；
- 数据解析：将公共的javaBean映射转换成业务定义的javaBean；
- 服务拦截器：拦截服务调用，用于业务来实现拦截服务请求的相关业务处理；
- 访问线程域：为方便参数在服务间的传递，对ThreadLocal做封装；
- 服务调用：采用开源SpringBoot的能力做服务调用的具体执行；
- 基础数据缓存：提供缓存类将基础静态数据在应用启动时加载到JVM缓存中，并提供静态数据获取API；
- 异常处理：封装异常类和定义异常编码；
- 服务访问日志查看：对接log4X系统实现服务日志的查看

提供服务监听，便于运维监控服务状态以及服务器状态；扩展序列化算法，提高数据转换和传输效率；
提供基础数据缓存解决方案，方便系统数据及业务数据的缓存，提高数据加载效率；提供异常处理解决方案，规范业务异常的处理流程；
提供服务访问拦截器的接口封装，便于业务开发扩展实现具体的业务服务请求拦截及相应的处理。

主体功能5— 数据库访问层

- 数据库访问层用于业务服务访问数据库的相关管控，包括数据库连接的建立，数据库访问，事务控制，数据库访问路由，数据库监控，负载均衡等。



- MyBatis集成：将开源MyBatis框架集成到分布式服务框架中；
- 数据库连接池配置：基于开源MyBatis对数据库连接池的配置能力；
- 事务控制：采用开源MyBatis提供的能力，对数据库访问事务一致性做控制；
- 数据库方言扩展：实现MyBatis提供的接口，适配MySQL和Oracle的sql拼写差异；
- 配置信息优化：将数据库访问层相关配置信息集中化配置；
- Druid SQL监控：集成开源Druid组件，做数据库访问性能监控、连接池资源使用情况监控和SQL执行日志记录。

通过对数据库方言的扩展，适配了MySQL和Oracle的sql拼写差异；

通过集成DruidSQL监控，实现数据库访问层的相关监控，便于维护和提高系统的可靠性。

主体功能6—注册中心

■ 注册中心用于服务注册和服务发现，在开源SpringCloud的Eureka组件的基础上做了一些扩展，增加服务治理的相关功能。



- 服务注册和服务发现：采用开源组件Eureka提供的能力；
- 服务器节点状态监听：提供可视化界面，用于展示注册到注册中心的服务器节点的状态；
- 服务器资源监控：提供可视化界面，用于展示注册到注册中心的服务器节点的资源（包括内存、cpu等）使用情况；
- 服务实例目录展示：提供可视化界面展示注册到注册中心的服务器节点列表信息；
- 服务目录展示：提供可视化界面展示注册到注册中心的服务器节点上部署的服务列表信息；
- 服务状态监控：提供可视化界面，展示服务的使用情况，包括调用响应时间、调用成功率等信息；
- 服务调用链跟踪：结合log4x日志系统，展示服务调用链信息；
- 服务访问授权：支持服务访问按app授权，每个app接入只可查看和访问自己注册的服务信息；
- 服务升降级：

通过对服务治理的扩展，便于更加直观监控服务器资源使用情况，服务运行状态等信息，提高了系统的可靠性和可维护性。

主体功能7—接口封装层

- 封装现有的一些中间件资源接口API，以SDK包的方式暴露给业务开发使用。

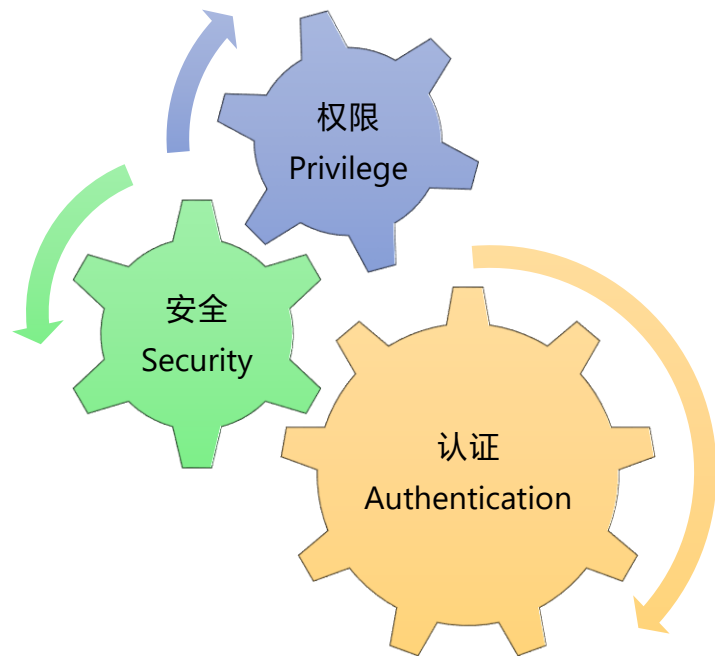


通过sdk的方式复用现有的资源，资源的使用直接从中间件资源池中申请，避免重复建设导致的浪费资源。

主体功能8—认证权限管理

Secframe是一套统一的、安全快捷的、可独立部署的认证和权限管理中心，同时提供安全管控、日志等安全保障。

- Security组件：提供安全访问控制、安全规范校验等支持；
- Privilege模块：负责对系统功能的权限管理及授权管理，含组织模型管理；
- Authentication模块：负责对访问系统的操作员或应用程序做身份的合法性认证，并提供账号注册、单点登录、密码修改等API供业务系统调



通过安全权限认证中心的接入使用，可保证业务系统本身具备独立的账号权限管理系统，并可独立使用；同时支持对4A系统的对接。

培训内容

一

背景

二

总体架构

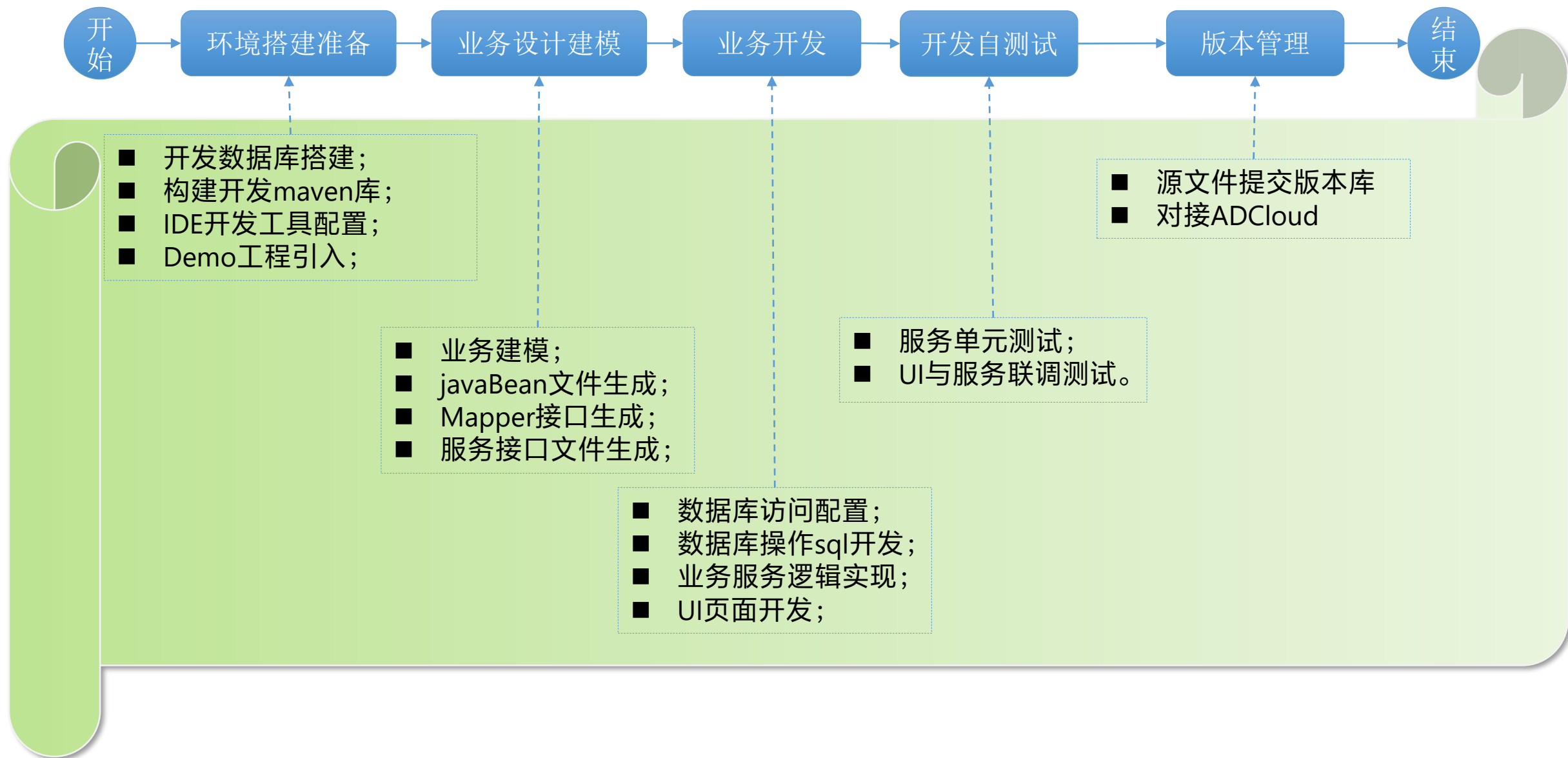
三

主体功能介绍

四

开发流程及demo

开发流程



构建Maven库

- Maven库存放云梯开发框架依赖的相关jar包，仅需统一搭建一套共享的Maven库即可，无需单独搭建；
- 业务系统在新建maven工程后，在pom.xml文件配置公共Maven库（可使用demo工程中的pom.xml文件替换）。

The screenshot displays the Sonatype Nexus web interface on the left and a corresponding pom.xml file configuration on the right.

Sonatype Nexus Interface:

- Left Sidebar:** Contains navigation links for Nexus, Artifact Search, Advanced Search, Views/Repositories, Repositories, Administration, and Help.
- Main Content Area:**
 - Repositories Tab:** Shows a table of repositories. The 'fast-osd' repository is highlighted.
 - fast-osd Repository Details:** Shows the repository structure under 'fast-osd' with a tree view of artifacts like 'com', 'ai', 'osd', 'cache-api', etc.

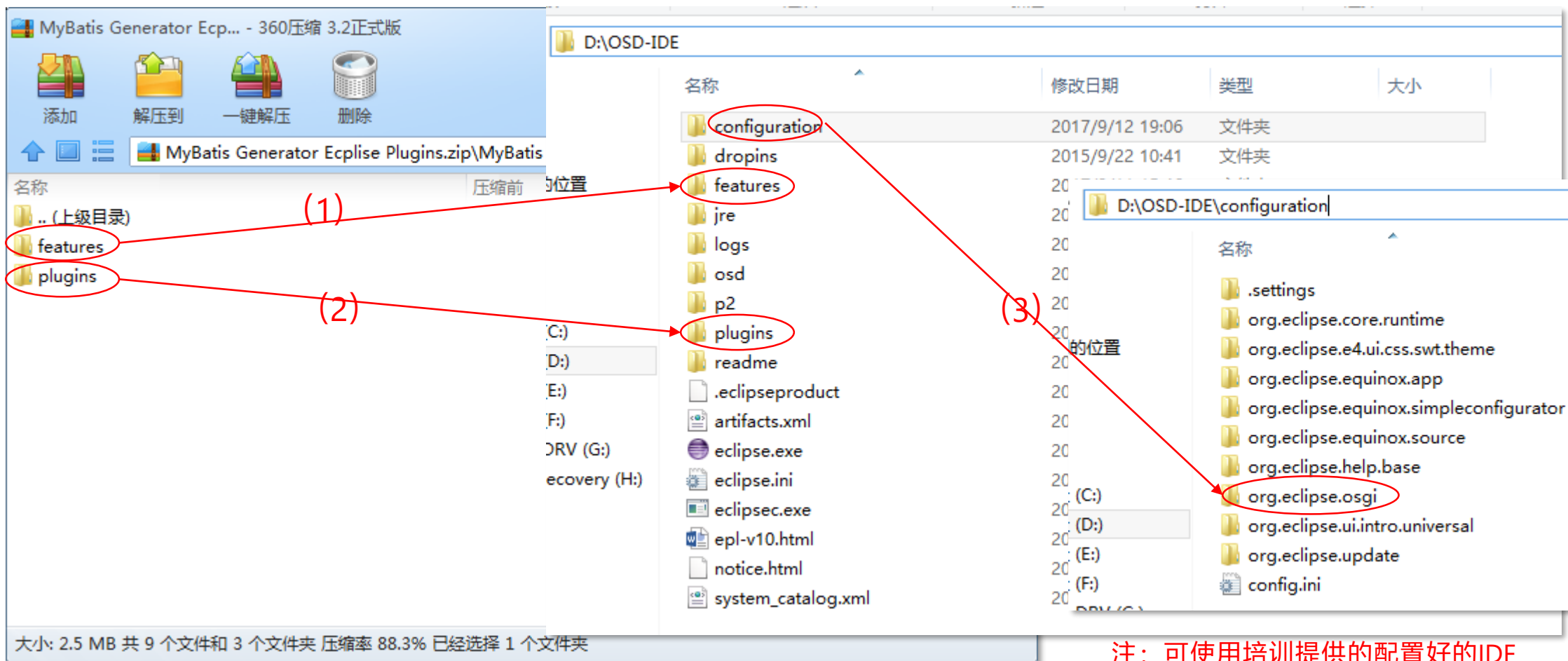
fast-osd/pom.xml Configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.ai.osd</groupId>
  <artifactId>cm-osd-server</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>
  <name>cm-osd-server</name>
  <description>cm-osd-server</description>
  <parent>
    <groupId>com.ai.osd</groupId>
    <artifactId>osd</artifactId>
    <version>1.0-SNAPSHOT</version>
  </parent>
  <repositories>
    <repository>
      <id>osd-proxy</id>
      <name>osd proxy</name>
      <url>http://10.70.181.10:8081/nexus/content/repositories/fast-osd-proxy</url>
    </repository>
    <repository>
      <id>fast-osd-release</id>
      <name>osd-release</name>
      <url>http://10.70.181.10:8081/nexus/content/repositories/fast-osd/</url>
    </repository>
    <repository>
      <id>fast-osd-snapshot</id>
      <name>osd snapshot</name>
      <url>http://10.70.181.10:8081/nexus/content/repositories/fast-release/</url>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <id>osd-proxy</id>
      <name>osd proxy</name>
      <url>http://10.70.181.10:8081/nexus/content/repositories/fast-osd-proxy</url>
    </pluginRepository>
  </pluginRepositories>
  <dependencies>
    <dependency>
      <groupId>com.ai.osd</groupId>
      <artifactId>osd</artifactId>
      <version>1.0-SNAPSHOT</version>
      <scope>compile</scope>
    </dependency>
  </dependencies>
</project>
```

配置IDE开发工具

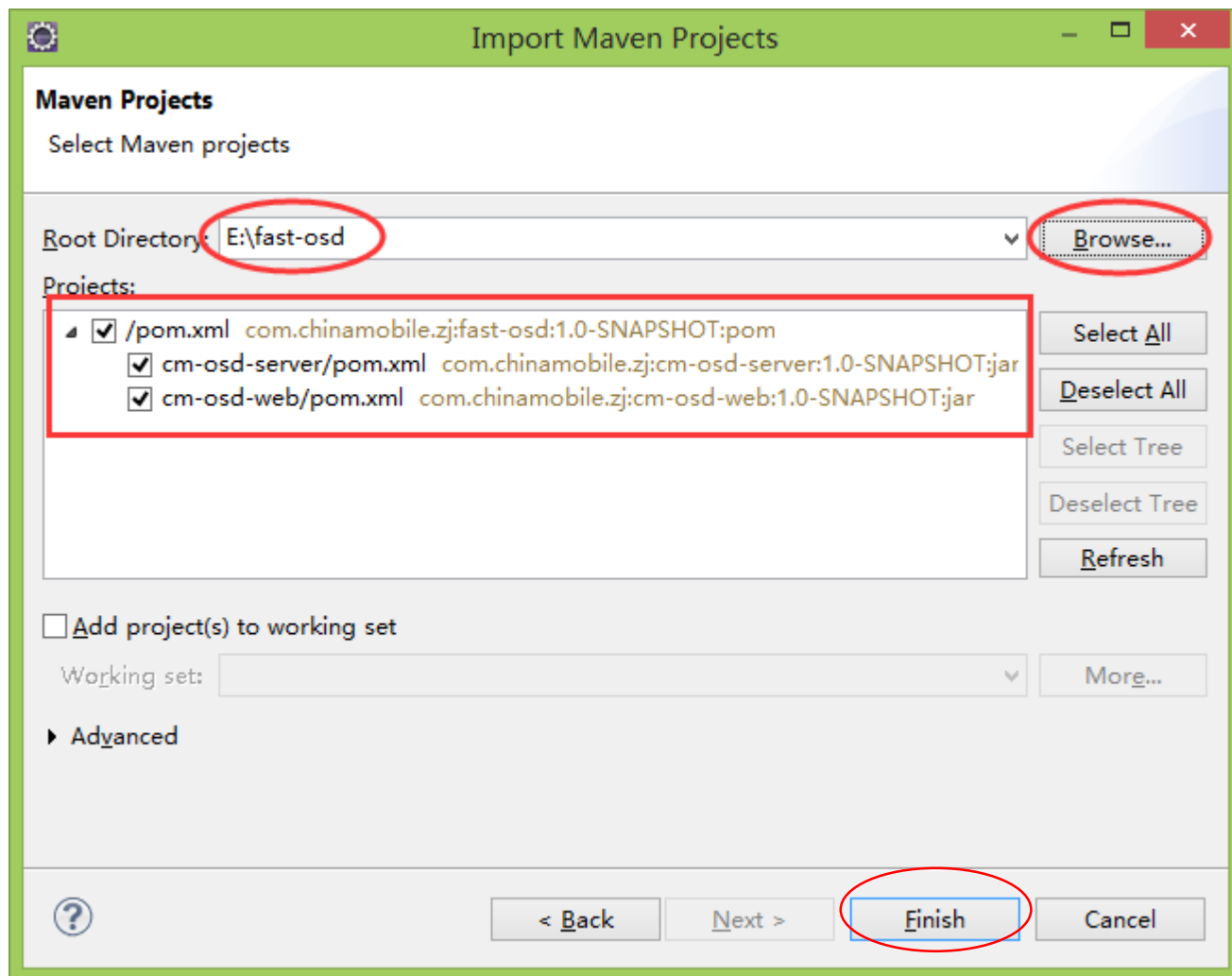
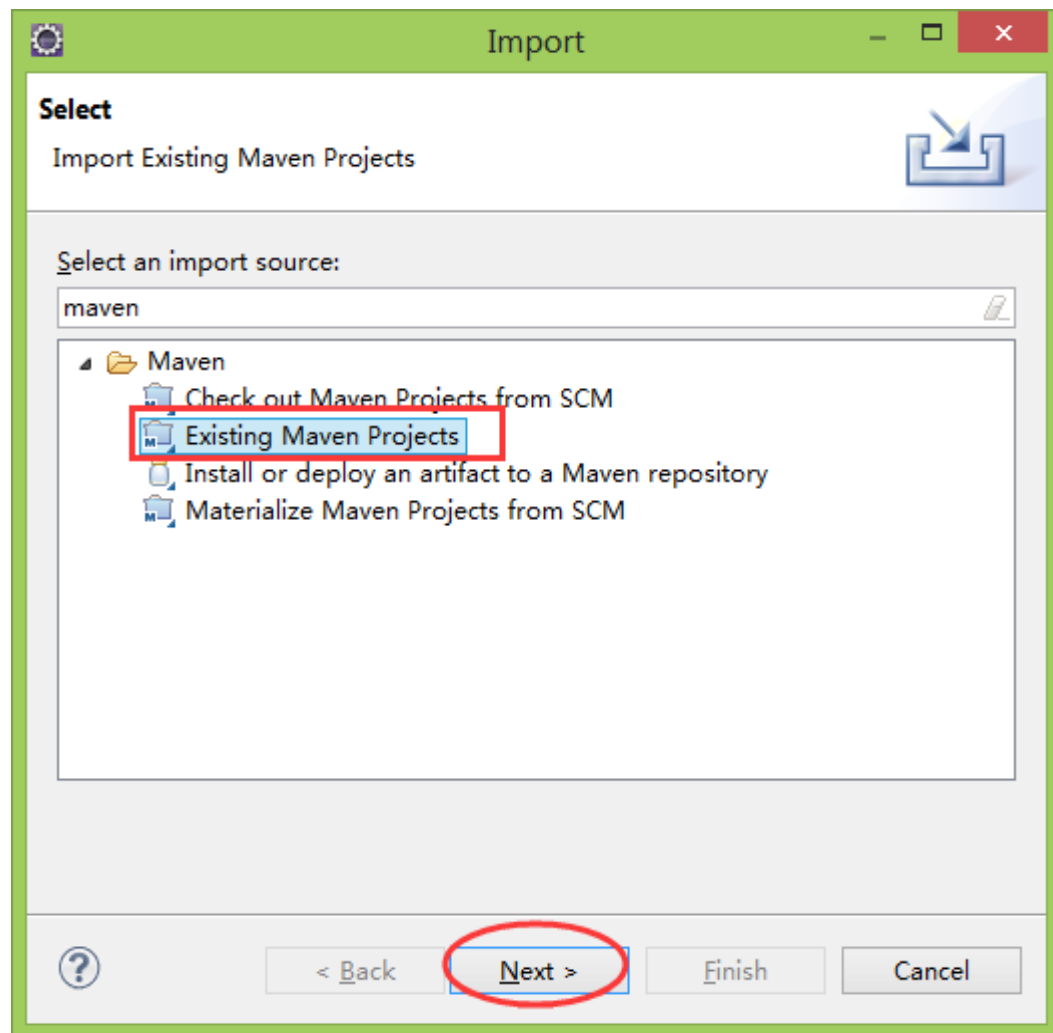
■ 云梯开发框架提供基于Eclipse的插件工具，用于javabean、mapper、criteria、handler等基础文件的生成，具体配置步骤如下：

- (1) 将插件包中的plugin目录下的文件放置在eclipse的plugin目录下；
- (2) 将插件包中的feature目录下的文件放置在eclipse的feature目录下；
- (3) 删除eclipse的configuration目录下的org.eclipse.osgi缓存文件夹。



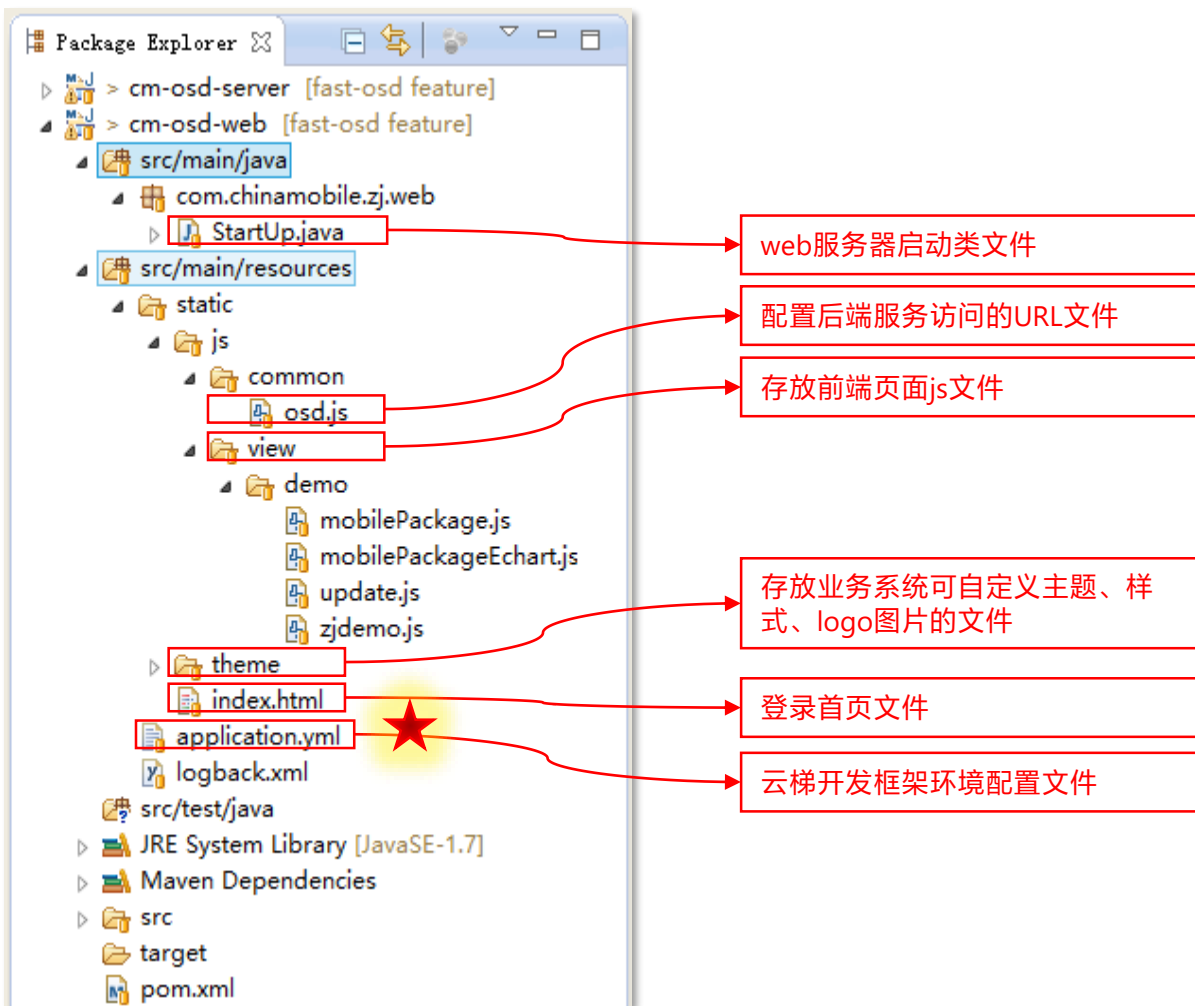
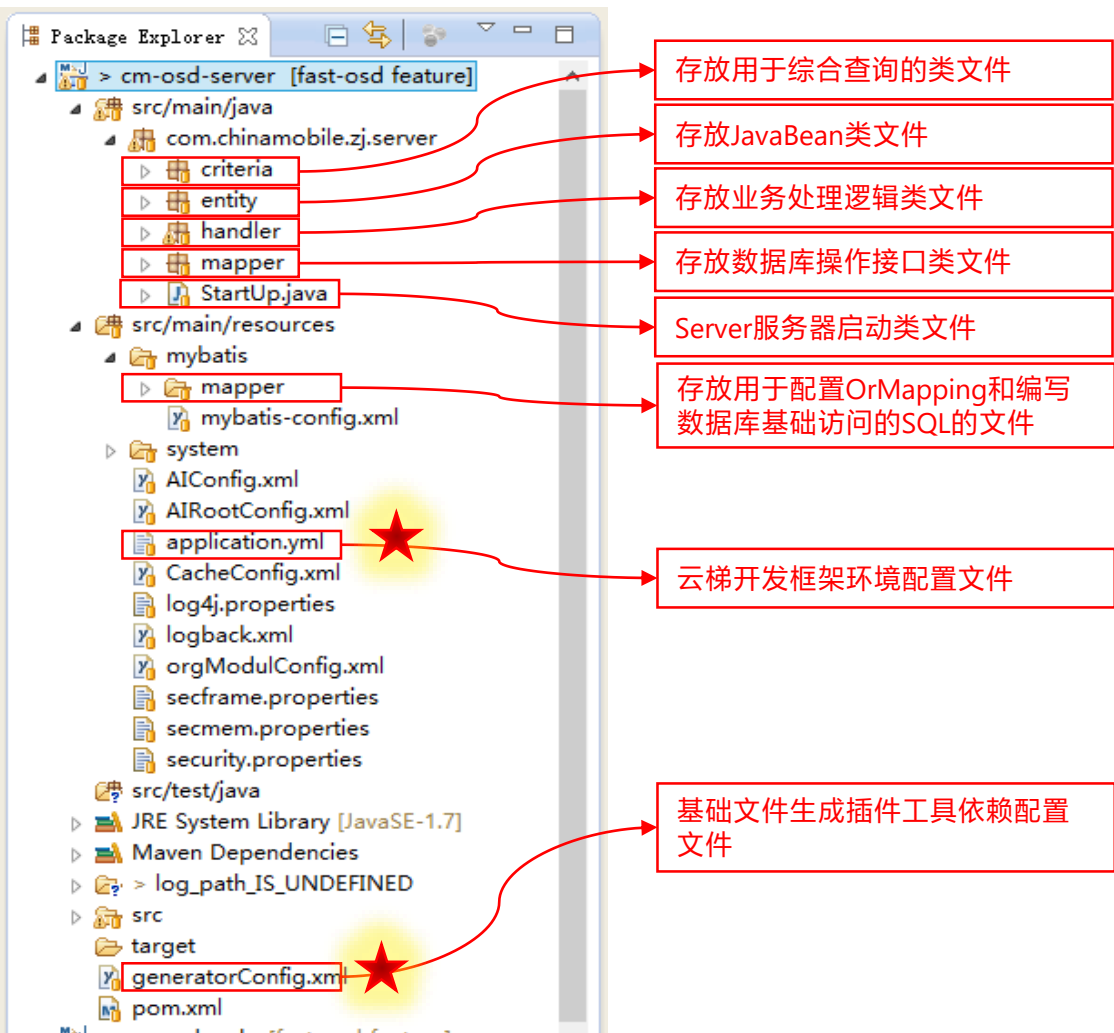
Demo工程的引入

- 启动eclipse，选择File→Import→选择Existing Maven Projects；
- 点击Next按钮，通过Browser选择demo工程目录，在Projects框中勾选所有的工程，点击Finish按钮即可完成demo工程的导入。



Demo工程结构

- Demo工程包含一个主工程fast-osd及其两个子工程cm-osd-server和cm-osd-web;
 - ◆ cm-osd-server用于开发业务模型及后端服务的相关代码;
 - ◆ cm-osd-web用于开发前端页面的相关代码。



配置文件application.yml(server)

```
server:
  port: ${odf.port:8990} #应用服务器端口
  tomcat:
    maxKeepAliveRequest: -1 #默认每100次请求
    重新建立连接, -1表示不限制
  # keepAliveTimeOut: 60000 #默认60s服务端断
  开连接
  #-----
  #boot admin
  info:
    app:
      name: "server" #从pom.xml中获取
      description: "@project.description@"
      version: "@project.version@"
  #-----
  spring:
    jackson:
      serialization: true
    application:
      name: cm-osd-server #应用服务器名称
    mail:
      from: ai@asiainfo.com
      host: mail.asiainfo.com
      username: user
      password: pwd
      properties:
        mail:
          smtp:
            auth: true
            starttls:
              enable: true
              required: true

#启时是否自检Email服务
management:
  security:
    enabled: false
  health:
    mail:
      enabled: false #启动时根据该参数判断是否装载health
  #-----
  eureka:
    client:
      registry-fetch-interval-seconds: 5 #eureka client间隔多
      久去拉取服务注册信息, 默认为30秒
      serviceUrl:
        defaultZone:
          ${eureka.serviceUrl:http://20.26.20.16:20036/eureka/} #注册中心服务器
      instance:
        metadataMap:
          instanceId: ${spring.application.name}:${random.value}
          preferIpAddress: true
          instanceId: ${HOST}:${PORT0} # 这里使用DCOS的环境变
          量
        #instanceId:
          ${spring.cloud.client.ipAddress}:${server.port}
        lease-expiration-duration-in-seconds: 5 #续约更新时间
        间隔 (默认30秒)
        lease-renewal-interval-in-seconds: 10 #续约到期时间
        (默认90秒)
        statusPageUrlPath: /info
        healthCheckUrlPath: /health
        ip-address: ${HOST} # 在DCOS环境, 需要指定IP
        non-secure-port: ${PORT0:80}
```

```
#数据库连接池
osd:
  druid:
    datasource:
      url: jdbc:oracle:thin:@20.26.11.5:1521/CSHP02
      username: osdf_kf
      password: osdf_kf
      driver-class-name: oracle.jdbc.driver.OracleDriver
    filters: stat
    maxActive: 100
    initialSize: 1
    maxWait: 3000
    minIdle: 1
    timeBetweenEvictionRunsMillis: 60000
    minEvictableIdleTimeMillis: 300000
    validationQuery: select 'x' from dual
    testWhileIdle: true
    testOnBorrow: true
    testOnReturn: false
    poolPreparedStatements: true
    maxOpenPreparedStatements: 50
  cache:
    enable: true
    cacheType: LOCAL
    overflowToDisk: false #当内存中对象数量达到最大时, 是否写磁盘.
    diskPersistent: false #是否缓存虚拟机重启期数据
    maxElementsInMemory: 0 #缓存最大个数,配为0表示不做限制
    timeToLiveSeconds: 0
    timeToIdleSeconds: 0
    diskExpiryThreadIntervalSeconds: 120 #磁盘失效线程运行时间间隔
    eternal: false #对象是否永久有效, 一旦设置了, timeout将不起作用
  #-----
  logging:
    level:
      ROOT: info
      com.ai.osd: debug
```

配置文件application.yml(web-1)

server:

port: \${odf.port:8986}

服务器端口

#context-path: /osd-web/

#-----

#boot admin 管理配置

management:

security:

enabled: false

health:

redis:

enabled: false

info:

app:

name: "web" #从pom.xml中获取

description: "@project.description@"

version: "@project.version@"

#-----

spring:

application:

name: cm-osd-web

服务器名称

session:

store-type: none #只有redis和none两个可选值

resources:

static-locations: classpath:/META-

INF/resources/,classpath:/resources/,classpath:/static/,classpath:/public/,f

ile:\${view.path}

#redis:

host:

port:

cluster:

nodes: 10.21.20.107:7000,10.21.20.107:7001,10.21.20.107:7002

eureka:

client:

registry-fetch-interval-seconds: 5 #eureka client间隔多久去拉取服务注册信息，默认为30秒

serviceUrl:

defaultZone: \${odf.eureka.serviceUrl:http://20.26.20.16:20036/eureka/}

注册中心服务器

instance:

preferIpAddress: true

instanceId: \${HOST}:\${PORT0} # 这里使用DCOS的环境变量

#instanceId: \${spring.cloud.client.ipAddress}:\${server.port}

leaseExpirationDurationInSeconds: 10 #续约到期时间（默认90秒）

lease-renewal-interval-in-seconds: 5 #续约更新时间间隔（默认30秒）

statusPageUrlPath: \${server.context-path:/}info

healthCheckUrlPath: \${server.context-path:/}health

ip-address: \${HOST} # 在DCOS环境，需要指定IP

non-secure-port: \${PORT0:80}

#metadata-map:

management.context-path:

配置文件application.yml(web-2)

#Fast osd 相关配置

osd:

server:

serviceld:

order: cm-osd-server

httpClient:

maxConnectionCount: 2000

defaultMaxPerRoute: 500

requestSentRetryEnabled: true

retryCount: 2

connectTimeout: 3000

readTimeout: 5000

connectionRequestTimeout: 200

bufferRequestBody: true

maxIdleTime: 60000

hystrix:

coreSize: 20

maxQueueSize: 500

queueSizeRejectionThreshold: 20

circuitBreakerRequestVolumeThreshold: 10

login:

filterPatterns: /* #多个过滤串以分号隔开

excludePatterns: services/*;four/a/* #不过滤的请求pattern，多个过滤串以分号隔开

excludePrefix:

health;info;configprops;logfile;metrics;jolokia;dump;trace;env;archaius;user;aifui;/data;/js;/theme;/service;webjars;/swagger-resources;swagger-

ui.html;v2;public/LOGIN/checkFirstLogin;public/LOGIN/checkChangePassword;public/@uspa_operatorAction/changePassword;login;i

nitSSO;index.html;ssolIndex.html; #不过滤的请求前缀，多个过滤串以分号隔开

excludeSuffix: .js;.jpg;.png;.css;.ico;.ttf;.woff2 #不过滤的请求后缀，多个过滤串以分号隔开

maxClientNum: 1 #0表示不限制客户端个数

maxExceeded: true #true表示客户端达到个数之后禁止登陆，false表示把之前登录的客户端踢掉

use4a: false #是否使用4a登录，默认不使用

ssoLimit: false #是否对4a系统登录的用户进行客户端个数限制，默认不限制

allowPath: login;index.html;ssolIndex.html;initSSO #4a过滤器配置

isLog: false #true或false

domain: zj.chinamobile.com #默认为浙江移动

cache:

enable: true

cacheType: LOCAL

overflowToDisk: false

diskPersistent: false

4a:

role: ZJROLE

organize: ZJORGANIZE

district: ZJDISTRICT

default-password: 123456

#-----

logging:

level:

ROOT: info

com.ai.osd: debug

配置文件generatorConfig.xml

- 该文件初始化工程需通过File→New→Other，选择MyBatis Generator Configuration File生成，初始化生成的文件参数可参考如下模板配置（可直接使用demo中提供的文件）；

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE generatorConfiguration PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN" "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd" >
<generatorConfiguration >
<classPathEntry location= "D:\OSD-IDE\osd\ojdbc5-11.2.0.1.0.jar" />
<context id="context1" >
  <plugin type="org.mybatis.generator.aicloud.codegen.mybatis3.plugin.CriteriaPackageRenamePlugin" >
    <property name="targetPackage" value="com.chinamobile.zj.server.criteria" />
  </plugin>
  <plugin type="org.mybatis.generator.aicloud.codegen.mybatis3.plugin.HandlerGenerator" >
    <property name="targetProject" value="cm-osd-server" />
    <property name="targetPackage" value="com.chinamobile.zj.server.handler" />
    <property name="isGenerateMethod" value="true" />
  </plugin>
  <jdbcConnection driverClass="com.mysql.jdbc.Driver" connectionURL="jdbc:mysql://10.11.20.101:3666/CSHP02" userId="osdf_kf" password="osdf_kf" />
<commentGenerator >
  <property name="suppressDate" value="true" />
  <property name="suppressAllComments" value="true" />
</commentGenerator>
<jdbcConnection driverClass="oracle.jdbc.driver.OracleDriver" connectionURL="jdbc:oracle:thin:@20.26.11.5:1521/CSHP02" userId="osdf_kf" password="osdf_kf" />
<javaModelGenerator targetPackage="com.chinamobile.zj.server.entity" targetProject="cm-osd-server" />
<sqlMapGenerator targetPackage="mybatis.mapper" targetProject="cm-osd-server/src/main/resources" />
<javaClientGenerator targetPackage="com.chinamobile.zj.server.mapper" targetProject="cm-osd-server" type="XMLMAPPER" />
<table schema="osdf_kf" tableName="mobile" domainObjectName="Mobile">
<!-- <columnOverride column="???" property="???" /> -->
</table>
<table tableName="package" domainObjectName="Package"> </table>
<table tableName="mobile_package" domainObjectName="MobilePackage"> </table>
<table schema="osdf_kf" tableName="cm_customer" domainObjectName="Customer">
</context>
</generatorConfiguration>
```

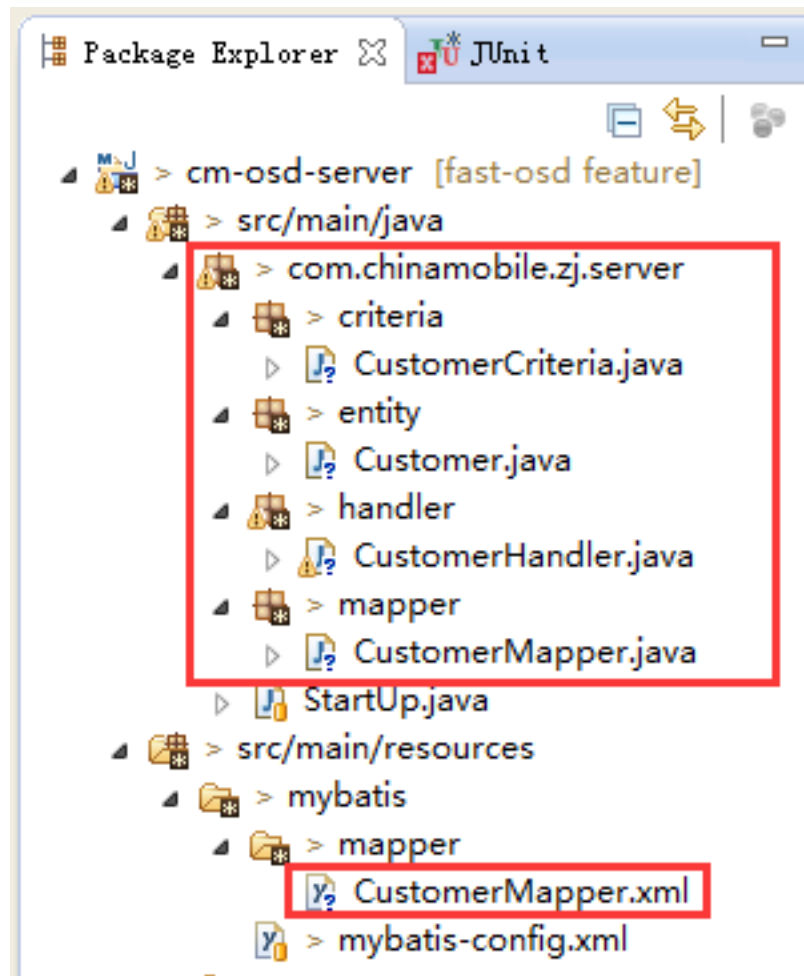
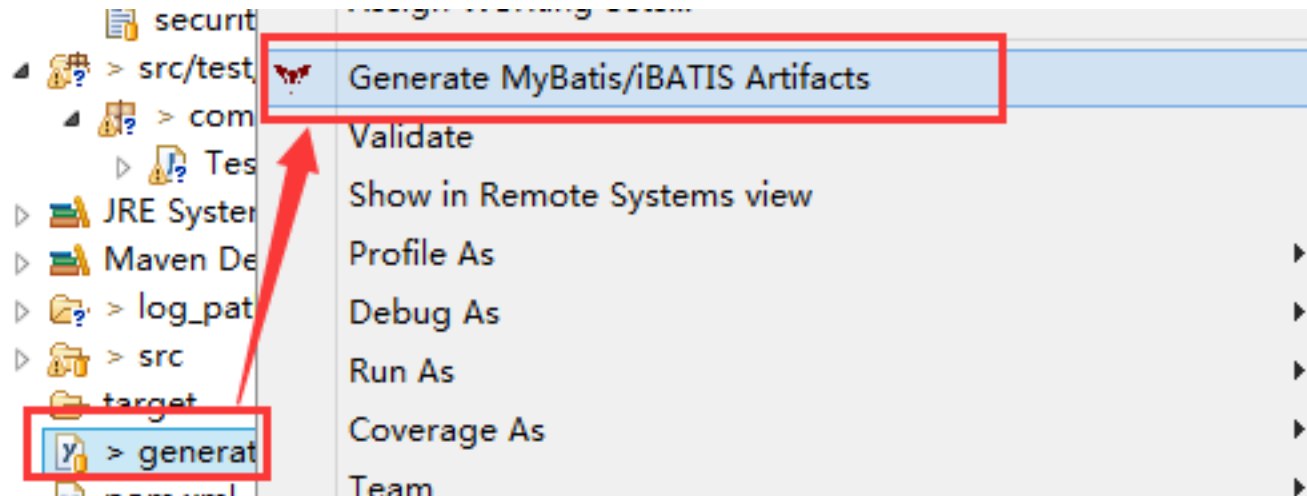
数据库访问驱动类文件配置

数据源配置

需要生成javaBean、mapper、handler等基础文件的表配置

基础文件生成

- 参考上页配置完generatorConfig.xml文件中的基础参数，并将需要生成基础文件的表配置在<table>中；
- 选择generatorConfig.xml文件，右键点击Generate MyBatis/iBATIS Artifacts菜单，即可生成相关基础文件。



业务服务代码逻辑实现编写

- 针对单个对象及其对用表的基础访问操作，在生成的***Handler.java文件中，已经生成空的服务，需要业务开发人员实现具体的业务逻辑；
- 复杂的后端业务服务，由业务开发人员在***Handler.java文件中自行新增和实现。
- 下面以Customer对象为例展示对该对象的新增、查询、修改、删除基础操作的代码实现：

```
CustomerHandler.java
1 package com.chinamobile.zj.server.handler;
2
3 import java.text.ParseException;
4
5 @EnableEventHandler("/CustomerHandler")
6 public class CustomerHandler {
7     @Autowired
8     private CustomerMapper mapper;
9
10    private CustomerHandler() {
11    }
12
13    @HandlerMapping("/addCustomer")
14    public Response addCustomer(AiRequest request) throws ParseException {
15        BaseTypeResponse response = new BaseTypeResponse(request.getHandlerType());
16        // 获取业务参数Map
17        Map<String, String> busParam = request.getBusParam();
18        // TODO业务处理逻辑
19        Customer customer = JSON.parseObject(busParam.get("customer"), Customer.class);
20        String custId = UUID.randomUUID().toString();
21        customer.setCustId(custId);
22        mapper.insert(customer);
23        response.setRetCode(MessageCode.SUCCESS);
24        response.setRetMessage("success");
25        return response;
26    }
27
28    @HandlerMapping("/updateCustomer")
29    public Response updateCustomer(AiRequest request) {
30        BaseTypeResponse response = new BaseTypeResponse(request.getHandlerType());
31        // 获取业务参数Map
32        Map<String, String> busParam = request.getBusParam();
33        // 业务处理逻辑
34        Customer customer = JSON.parseObject(busParam.get("customer"), Customer.class);
35        CustomerCriteria cond = new CustomerCriteria();
36        // 构造待更新记录对象
37        String custId = customer.getCustId();
38        cond.createCriteria().andCustIdEqualTo(custId);
39        mapper.updateByCriteriaSelective(customer, cond);
40        response.setRetCode(MessageCode.SUCCESS);
41        response.setRetMessage("success");
42        return response;
43    }
44 }
```

```
@HandlerMapping("/queryCustomer")
public Response queryCustomer(AiRequest request) {
    ListResponse response = new ListResponse(request.getHandlerType());
    // 获取业务参数Map
    Map<String, String> busParam = request.getBusParam();
    // TODO业务处理逻辑
    String custName = busParam.get("custName");
    CustomerCriteria cond = new CustomerCriteria();
    // TODO填充查询条件
    cond.createCriteria().andCustNameEqualTo(custName);
    List<Customer> list = mapper.selectByCriteria(cond);
    if (list != null && !list.isEmpty()) {
        List<Map<String, String>> resultList = new ArrayList<Map<String, String>>();
        for (Object obj : list) {
            Map<String, String> map = SystemHelper.objectToMap(obj);
            resultList.add(map);
        }
        response.setDataList(resultList);
    }
    response.setRetCode(MessageCode.SUCCESS);
    response.setRetMessage("success");
    return response;
}

@HandlerMapping("/deleteCustomer")
public Response deleteCustomer(AiRequest request) {
    BaseTypeResponse response = new BaseTypeResponse(request.getHandlerType());
    // 获取业务参数Map
    Map<String, String> busParam = request.getBusParam();
    // 业务处理逻辑
    String custId = busParam.get("custId");
    CustomerCriteria cond = new CustomerCriteria();
    // TODO填充查询条件
    cond.createCriteria().andCustIdEqualTo(custId);
    mapper.deleteByCriteria(cond);
    response.setRetCode(MessageCode.SUCCESS);
    response.setRetMessage("success");
    return response;
}
```


业务服务服务JUnit测试编写

- 业务服务的JUnit单元测试类按规范建在test目录下，子目录与对应的*Handler.java文件一致，类名为Test**Handler.java，方法名为test**()。
- 下面以CustomerHandler.java中addCustomer方法的JUnit单元测试编码为例展示单元测试代码编写：

The screenshot displays an IDE with two main panels. The left panel, titled 'Package Explorer', shows a project structure for 'cm-osd-server'. The 'src/test/java' directory is expanded, and the file 'TestCustomerHandler.java' is selected and highlighted with a red circle. The right panel shows the code for 'TestCustomerHandler.java'. The code includes package declarations, imports, and annotations. The annotations `@RunWith(SpringJUnit4ClassRunner.class)` and `@SpringBootTest(classes = StartUp.class)` are highlighted with a red box. The test method `testAddCustomer()` is also highlighted with a red circle. The method body contains logic to create an `AiRequest` object, set customer details, and call the `addCustomer` method on the `CustomerHandler`.

```
1 package com.chinamobile.zj.server;
2
3 import java.text.ParseException;
4
5 @RunWith(SpringJUnit4ClassRunner.class)
6 @SpringBootTest(classes = StartUp.class)
7
8 public class TestCustomerHandler {
9     @Autowired
10     private CustomerHandler handler;
11
12     @Test
13     public void testAddCustomer() throws ParseException {
14         AiRequest request = new AiRequest();
15         Customer customer = new Customer();
16         customer.setCustName("ylh");
17         SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
18         Date birthDate = sdf.parse("1983-11-14");
19         customer.setBirthDate(birthDate);
20         String jsonString = JSON.toJSONString(customer);
21         System.out.println(jsonString);
22         Map busParam = new HashMap();
23         busParam.put("customer", jsonString);
24         request.setBusParam(busParam);
25         handler.addCustomer(request);
26     }
27 }
```

前端页面代码模板及后端服务URL配置

js-template.js

```
1 define(["core", "table", "form"], function() {
2     var $table, $form, $searchBtn, $addBtn;
3     /*
4      * 初始化文档流
5      */
6     function initDom($page) {
7         //页面元素代码
8     }
9     /*
10    * 初始化事件
11    */
12    function initEvent($page) {
13        //页面元素绑定的事件的js方法
14    }
15    /*
16    * 渲染
17    * $page为一个静态的空div,该div已存在于文
18    */
19    function create($page) {
20        initDom($page);
21        initEvent($page);
22        return $page;
23    }
24    /*销毁,关闭页签前执行回收内存*/
25    function destroy() {
26        UI.alert("回收该模块相应的内存");
27    }
28    /*刷新*/
29    function refresh() {
30    }
31    return {
32        create: create
33    };
34 });
```

components

- btncheckbox
- checkbox
- combo
- combobox
- comboclock
- combodate
- combotree
- datepicker
- dialog
- dropdown
- file
- form
- pagination
- radio
- scrollbox
- sidewin
- switch
- table
- textarea
- textinput
- textnumber
- timepicker
- transfer
- tree
- treetable
- upload

osd.js

```
1 /*
2  * 所有的请求都在这里咯
3  */
4 define(["request"], function() {
5
6     /*
7     * 服务请求路径
8     */
9     return {
10         init: function () {
11             Request.set("add", "", "/public/CustomerHandler/addCustomer");
12             Request.set("search", "", "/public/CustomerHandler/queryCustomer");
13             Request.set("delete", "", "/public/CustomerHandler/updateCustomer");
14             Request.set("update", "", "/public/CustomerHandler/deleteCustomer");
15         }
16     };
17
18 });
```

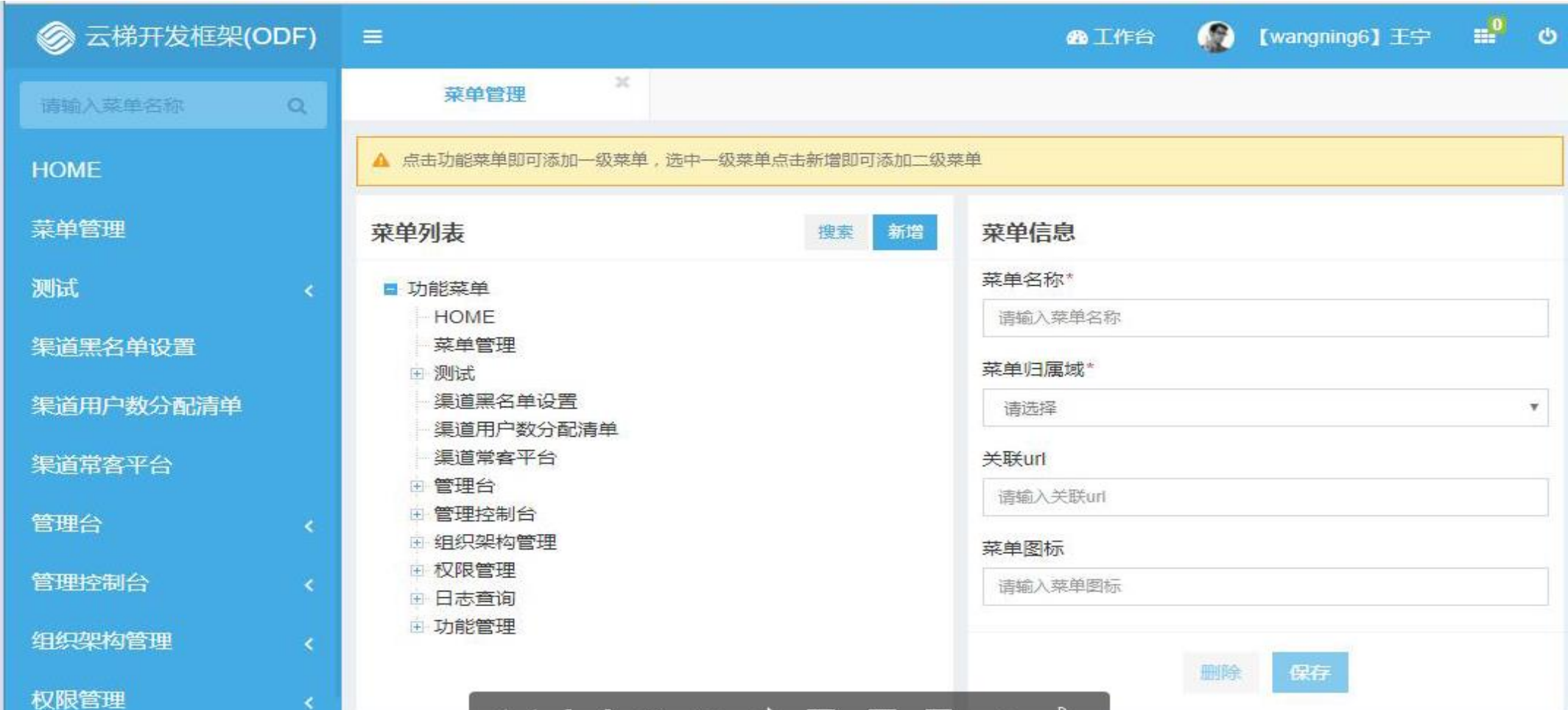
启动工程

- 分别以java application方式运行server工程和web工程中的StartUp.java中的main方法启动server服务器和web服务器；
- 启动完成后可访问前端页面框架登录首页：<http://127.0.0.1:port>（port来源于web工程application.yml中配置的port）
- 开发态调试登录的用于名密码为：admin/123456



页面菜单配置及页面集成调试

- 登录系统后，可在菜单管理页面选择已有的一个菜单，点击新增则在下级新增菜单，在弹出的页面上输入菜单名、归属域和菜单URL并选择菜单图标，点击“确定”按钮，即可新增新开发的菜单，新增完成后在右侧菜单栏右键选择“重新加载”即可看到新增的菜单。
- 菜单关联的URL配置为相对路径，从js目录的上级路径开始，如：../js/view/demo/customerManager



对接ADCloud

■ 开发自测完成后提交源码到GIT库，ADCloud配置GIT库获取代码进行安全扫描和持续集成。

Eureka

ODF

ADCloud首页/项目管理

devops.yw.zj.chinamobile.com/dist/

ADCloud 敏捷交付云平台

首页

集成管理

报表管理

环境管理

?

youwei

fast-osd...

fast-osd/eureka集成流水

fast-osd/fast-osd集成流水

应用

eureka

fast-osd

构建报表

构建报表

构建耗时

发布

发布计划

发布记录

代码

环境配置

系统设置

测试环境-sonar

构建 设置 订阅 日志

下载

开始时间: 09-11 17:56:41

耗时: 5秒

commitId : 0da7c9a

更新文件数 : 10

代码扫描

开始时间: 09-11 17:56:46

耗时: 37秒

扫描详情 : 查看详情

开发环境-fast

构建 设置 订阅 日志

下载

开始时间: 09-14 11:14:28

耗时: 6秒

commitId : 9ca5fc7

更新文件数 : 5

编译、打包

开始时间: 09-14 11:14:34

耗时: 1分0秒

编译文件数 : 14

部署

开始时间: 09-14 11:15:34

耗时: 33秒

部署包1 :

部署IP : server,odfweb

开发环境-安全

构建 设置 订阅 日志

Durid 数据库访问监控

■ Durid监控提供数据源配置查看、sql监控等功能，便于运维人员进行数据库访问的相关监控。

Eureka

ODF

ADCloud首页/项目管理

Sonatype Nexus

Druid Stat Index

20.26.20.16:20035/druid/index.html

Druid Monitor

首页

数据源

SQL监控

SQL防火墙

Web应用

URI监控

Session监控

spring监控

JSON API

重置

记录日志并重置

English | 中文

Stat Index 查看JSON API

| | |
|--------------|---|
| 版本 | 1.0.18 |
| 驱动 | sun.jdbc.odbc.JdbcOdbcDriver com.alibaba.druid.mock.MockDriver com.mysql.jdbc.Driver com.mysql.fabric.jdbc.FabricMySQLDriver oracle.jdbc.OracleDriver com.alibaba.druid.proxy.DruidDriver |
| 是否允许重置 | true |
| 重置次数 | 0 |
| java版本 | 1.7.0_79 |
| jvm名称 | Java HotSpot(TM) 64-Bit Server VM |
| classpath 路径 | /app/app/server/config /app/app/server/lib/HdrHistogram-2.1.9.jar /app/app/server/lib/activation-1.1.jar /app/app/server/lib/aicache-2.4.4.jar /app/app/server/lib/antlr-2.7.7.jar /app/app/server/lib/antlr-runtime-3.4.jar /app/app/server/lib/aopalliance-1.0.jar /app/app/server/lib/appframe-6.0.2.jar /app/app/server/lib/appframe_exeframe_cache-1.0.jar |

注册中心监控

■ 注册中心监控用于查看注册到注册中心的服务器列表及其当前运行状态，注册中心主机资源使用情况等，由省中心进行运维管控。

Eureka

20.26.20.16:20036

spring Eureka

HOME LAST 1000 SINCE STARTUP

System Status

| | | | |
|-------------|---------|--------------------------|---------------------------|
| Environment | test | Current time | 2017-09-14T16:45:57 +0800 |
| Data center | default | Uptime | 1 day 23:55 |
| | | Lease expiration enabled | true |
| | | Renews threshold | 10 |
| | | Renews (last min) | 72 |

THE SELF PRESERVATION MODE IS TURNED OFF.THIS MAY NOT PROTECT INSTANCE EXPIRY IN CASE OF NETWORK/OTHER PROBLEMS.

DS Replicas

localhost

Instances currently registered with Eureka

| Application | AMIs | Availability Zones | Status |
|-------------------|---------|--------------------|----------------------------|
| CM-OSD-SERVER | n/a (1) | (1) | UP (1) - 20.26.28.30:31287 |
| CM-OSD-WEB | n/a (1) | (1) | UP (1) - 20.26.28.29:31662 |
| MY-CM-OSD-SERVER | n/a (1) | (1) | UP (1) - 10.21.86.12:8990 |
| MY-CM-OSD-WEB | n/a (1) | (1) | UP (1) - 10.21.86.12:8986 |
| YLH-CM-OSD-SERVER | n/a (1) | (1) | UP (1) - 10.70.165.19:8990 |
| YLH-CM-OSD-WEB | n/a (1) | (1) | UP (1) - 10.70.165.19:8986 |



General Info

| Name | Value |
|----------------------|-------------------------------|
| current-memory-usage | 446mb (43%) |
| num-of-cpus | 12 |
| environment | test |
| total-avail-memory | 1017mb |
| server-uptime | 1 day 23:55 |
| available-replicas | |
| registered-replicas | http://localhost:8988/eureka/ |
| unavailable-replicas | http://localhost:8988/eureka/ |

Instance Info

| Name | Value |
|--------|-------------|
| ipAddr | 20.26.28.30 |
| status | UP |

SpringBoot服务器监控



APPLICATIONS










JOURNAL

ABOUT

Spring-Boot applications

Here you'll find all Spring-Boot applications that registered themselves at this admin application.

Filter

| Application ▲ / URL | Version | Info | Status | |
|--|---------|--|--------|---|
| OSD-EUREKA http://20.26.28.29:31327 | | app: name: eureka description: '@project.description@'  | UP | <div>Details </div> <div></div> |
| OSD-SERVER http://20.26.28.30:31850 | | app: name: server description: '@project.description@'  | UP | <div>Details </div> <div></div> |
| OSD-WEB http://20.26.28.29:31272 | | app: name: web description: '@project.description@'  | UP | <div>Details </div> <div></div> |



THANK YOU!