## 前言

## 设计约束

## 设计概述



### 3.1系统功能设计

### 3.2功能点描述

|  |  |  |
| --- | --- | --- |
| **所属模块** |  | **功能清单** |
| 服务编排 | 应用查看修改 | 部署应用列表 |
| 查看应用 |
| 修改应用 |
| 停止应用 |
| 应用重启 |
| 应用服务部署 | 应用模板选择 |
| 创建应用部署 |
| 容器端口配置 |
| 模块环境变量配置 |
| 创建网络组件 |
| 配置网络端口 |
| 控制台信息输出 |
| 应用状态监控 | 平台状态总览 | 集群物理设备信息 |
| 微服务资源使用情况 |
| 平台实时CPU负载 |
| 平台实时内存负载 |
| 平台微服务负载分布 |
| 平台应用部署量趋势分析 |
| 平台资源总览 |
| 应用状态总览 |
| 应用服务状态检测 | 应用CPU实时使用 |
| 应用内存实时使用 |
| 微服务状态 |
| 运行状态预警 |
| 资源弹性伸缩状态 |
| 容器模板信息 |
| 应用模块信息 |
| 微服务信息 |
| 网络服务 |

### 3.3数据库结构设计

#### 3.3.1数据库表说明

#### 3.3.1.1 容器主表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | DB.CONTAINER | | | | | | |
| 表说明 | | 容器相关数据存储 | | | | | | |
| 序号 | 字段名 | 类型 | 是否主键 | 字段说明 | 长度 | 栏位格式 | 默认值 | 备注 |
| 1. | ID | VARCHAR | Y | 应用模块编号 | 40 | varchar |  |  |
| 2. | CONTAINERNAME | VARCHAR | N | 容器名称 | 50 | varchar |  |  |
| 3. | IMAGENAME | VARCHAR | N | 容器镜像名称 | 100 | varchar |  |  |
| 4. | IMAGEPOLICY | VARCHAR | N | 镜像拉取协议 | 1 | varchar |  |  |
| 5. | WORKCATALOG | VARCHAR | N | 默认值 | 300 | varchar | default |  |
| 6. | SHAREVOLUMENAME | VARCHAR | N | 共享卷名称 | 300 | varchar |  |  |
| 7. | VOLUMECONTAINERPATH | VARCHAR | N | 共享卷路径 | 1 | varchar |  |  |
| 8. | READONLY | VARCHAR | N | 只可读 | 1 | varchar | N |  |
| 9. | CPULIMIT | INTEGER | N | CPU限制 |  | int |  |  |
| 10. | MEMLIMIT | INTEGER | N | 内存限制 |  | int |  |  |
| 11. | CPUREQUEST | INTEGER | N | CPU请求 |  | int |  |  |
| 12. | MEMREQUEST | INTEGER | N | 内存请求 |  | int |  |  |
| 14. | RELATIVEID | VARCHAR | N | 所属的应用ID | 40 | varchar |  |  |
| 15. | CREATEUSERID | VARCHAR | N | 创建者ID | 40 | varchar |  |  |
| 16. | CREATEUSERNO | VARCHAR | N | 创建者账号 | 10 | varchar |  |  |
| 17. | CREATEDATETIME | VARCHAR | N | 创建时间 | 20 | varchar |  |  |
| 18. | UPDATEUSERID | VARCHAR | N | 修改者ID | 40 | varchar |  |  |
| 19. | UPDATEUSERNO | VARCHAR | N | 修改者账号 | 10 | varchar |  |  |
| 20. | UPDATEDATETIME | VARCHAR | N | 修改时间 | 20 | varchar |  |  |
| 21. | INTISDELETE | INTEGER | N | 是否（1/0）被删除 |  | int |  |  |
| 22. | INTISACTIVE | INTEGER | N | 状态是否（1/0）正常 |  | int |  |  |
| 23. | COMPANYID | VARCHAR | N | 所属公司ID | 40 | varchar |  |  |

**3.3.1.2 容器端口表**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | DB.CONTAINERPORT | | | | | | |
| 表说明 | | 容器端口相关配置 | | | | | | |
| 序号 | 字段名 | 类型 | 是否主键 | 字段说明 | 长度 | 栏位格式 | 默认值 | 备注 |
| 1. | ID | VARCHAR | Y | 应用模块编号 | 40 | varchar |  |  |
| 2. | CONTAINERNAME | VARCHAR | N | 端口所属容器名称 |  | varchar |  |  |
| 3. | PORTNUMBER | VARCHAR | N | 端口号 | 300 | varchar |  |  |
| 4. | PORTPPROTOCOL | VARCHAR | N | 端口协议 | 20 | varchar |  |  |
| 5. | DEPLOYMENTID | VARCHAR | N | 端口所属的应用模块ID | 40 | varchar |  |  |
| 6. | RELATIVEID | VARCHAR | N | 所属的应用ID | 40 | varchar |  |  |
| 7. | CREATEUSERID | VARCHAR | N | 创建者ID | 40 | varchar |  |  |
| 8. | CREATEUSERNO | VARCHAR | N | 创建者账号 | 10 | varchar |  |  |
| 9. | CREATEDATETIME | VARCHAR | N | 创建时间 | 20 | varchar |  |  |
| 10. | UPDATEUSERID | VARCHAR | N | 修改者ID | 40 | varchar |  |  |
| 11. | UPDATEUSERNO | VARCHAR | N | 修改者账号 | 10 | varchar |  |  |
| 12. | UPDATEDATETIME | VARCHAR | N | 修改时间 | 20 | varchar |  |  |
| 13. | INTISDELETE | INTEGER | N | 是否（1/0）被删除 |  | int |  |  |
| 14. | INTISACTIVE | INTEGER | N | 状态是否（1/0）正常 |  | int |  |  |
| 15. | COMPANYID | VARCHAR | N | 所属公司ID | 40 | varchar |  |  |

**3.3.1.3 容器环境变量表**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | DB.CONTAINERENV | | | | | | |
| 表说明 | | 容器环境变量相关配置 | | | | | | |
| 序号 | 字段名 | 类型 | 是否主键 | 字段说明 | 长度 | 栏位格式 | 默认值 | 备注 |
| 1. | ID | VARCHAR | Y | 应用模块编号 | 40 | varchar |  |  |
| 2. | CONTAINERNAME | VARCHAR | N | 端口所属容器名称 | 50 | varchar |  |  |
| 3. | ENVVALNAME | VARCHAR | N | 环境变量名称 | 100 | varchar |  |  |
| 4. | ENVVALVALUE | VARCHAR | N | 环境变量的值 | 300 | varchar |  |  |
| 5. | DEPLOYMENTID | VARCHAR | N | 所属的应用模块的ID | 40 | varchar |  |  |
| 6. | RELATIVEID | VARCHAR | N | 所属的应用ID | 40 | varchar |  |  |
| 7. | CREATEUSERID | VARCHAR | N | 创建者ID | 40 | varchar |  |  |
| 8. | CREATEUSERNO | VARCHAR | N | 创建者账号 | 10 | varchar |  |  |
| 9. | CREATEDATETIME | VARCHAR | N | 创建时间 | 20 | varchar |  |  |
| 10. | UPDATEUSERID | VARCHAR | N | 修改者ID | 40 | varchar |  |  |
| 11. | UPDATEUSERNO | VARCHAR | N | 修改者账号 | 10 | varchar |  |  |
| 12. | UPDATEDATETIME | VARCHAR | N | 修改时间 | 20 | varchar |  |  |
| 13. | INTISDELETE | INTEGER | N | 是否（1/0）被删除 |  | int |  |  |
| 14. | INTISACTIVE | INTEGER | N | 状态是否（1/0）正常 |  | int |  |  |
| 15. | COMPANYID | VARCHAR | N | 所属公司ID | 40 | varchar |  |  |

**3.3.1.4 容器组（微服务）表**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | DB.POD | | | | | | |
| 表说明 | | 容器组相关的数据 | | | | | | |
| 序号 | 字段名 | 类型 | 是否主键 | 字段说明 | 长度 | 栏位格式 | 默认值 | 备注 |
| 1. | ID | VARCHAR | Y | 应用模块编号 | 40 | varchar |  |  |
| 2. | PODNAME | VARCHAR | N | 容器组名称 | 50 | varchar |  |  |
| 3. | PODNAMESPACE | VARCHAR | N | 容器组命名空间 | 100 | varchar | default |  |
| 4. | RESTARTSTRATEGY | VARCHAR | N | 重启协议 | 1 | varchar | 空 | 无效 |
| 5. | LABELNAME | VARCHAR | N | 标签名称 | 100 | varchar |  |  |
| 6. | LABELVALUE | VARCHAR | N | 标签值 | 300 | varchar |  |  |
| 7. | SECRETNAME | VARCHAR | N | 密码名称 | 200 | varchar | 空 | 无效 |
| 8. | REALNAME | VARCHAR |  | 真实值 | 50 | varchar | default | 无效 |
| 9. | RELATIVEID | VARCHAR | Y | 所属的应用ID | 40 | varchar |  |  |
| 10. | CREATEUSERID | VARCHAR | N | 创建者ID | 40 | varchar |  |  |
| 11. | CREATEUSERNO | VARCHAR | N | 创建者账号 | 10 | varchar |  |  |
| 12. | CREATEDATETIME | VARCHAR | N | 创建时间 | 20 | varchar |  |  |
| 13. | UPDATEUSERID | VARCHAR | N | 修改者ID | 40 | varchar |  |  |
| 14. | UPDATEUSERNO | VARCHAR | N | 修改者账号 | 10 | varchar |  |  |
| 15. | UPDATEDATETIME | VARCHAR | N | 修改时间 | 20 | varchar |  |  |
| 16. | INTISDELETE | INTEGER | N | 是否（1/0）被删除 |  | int |  |  |
| 17. | INTISACTIVE | INTEGER | N | 状态是否（1/0）正常 |  | int |  |  |
| 18. | COMPANYID | VARCHAR | N | 所属公司ID | 40 | varchar |  |  |

**3.3.1.5 应用模块主表**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | DB.DEPLOYMENT | | | | | | |
| 表说明 | | 应用模块属性配置数据 | | | | | | |
| 序号 | 字段名 | 类型 | 是否主键 | 字段说明 | 长度 | 栏位格式 | 默认值 | 备注 |
| 1. | ID | VARCHAR | Y | 应用模块编号 | 40 | varchar |  |  |
| 2. | DEPLOYMENTNAME | VARCHAR | N | 应用模块名称 | 50 | varchar |  |  |
| 3. | DEPLOYMENTNAMESPACE | VARCHAR | N | 应用模块命名空间 | 100 | varchar | default |  |
| 4. | REPLICAS | VARCHAR | N | 副本数量 | 50 | varchar | 1 |  |
| 5. | LABELNAME | VARCHAR | N | 标签名称 | 100 | varchar |  |  |
| 6. | LABELVALUE | VARCHAR | N | 标签值 | 300 | varchar |  |  |
| 7. | INITVALUE | VARCHAR | N | 对应的模块元素的初始值 | 20 | varchar |  |  |
| 8. | REALNAME | VARCHAR | N | 真实值 | 50 | varchar | default | 无效 |
| 9. | RELATIVEID | VARCHAR | Y | 所属的应用ID | 40 | varchar |  |  |
| 10. | CREATEUSERID | VARCHAR | N | 创建者ID | 40 | varchar |  |  |
| 11. | CREATEUSERNO | VARCHAR | N | 创建者账号 | 10 | varchar |  |  |
| 12. | CREATEDATETIME | VARCHAR | N | 创建时间 | 20 | varchar |  |  |
| 13. | UPDATEUSERID | VARCHAR | N | 修改者ID | 40 | varchar |  |  |
| 14. | UPDATEUSERNO | VARCHAR | N | 修改者账号 | 10 | varchar |  |  |
| 15. | UPDATEDATETIME | VARCHAR | N | 修改时间 | 20 | varchar |  |  |
| 16. | INTISDELETE | INTEGER | N | 是否（1/0）被删除 |  | int |  |  |
| 17. | INTISACTIVE | INTEGER | N | 状态是否（1/0）正常 |  | int |  |  |
| 18. | COMPANYID | VARCHAR | N | 所属公司ID | 40 | varchar |  |  |

**3.3.1.6 网络组件主表**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | DB.SERVICE | | | | | | |
| 表说明 | | 网络组件属性配置数据 | | | | | | |
| 序号 | 字段名 | 类型 | 是否主键 | 字段说明 | 长度 | 栏位格式 | 默认值 | 备注 |
| 1. | ID | VARCHAR | N | 网络组件编号 | 40 | varchar |  |  |
| 2. | SERVICENAME | VARCHAR | Y | 网络组件名称 | 50 | varchar |  |  |
| 3. | SERVICENAMESPACE | VARCHAR | N | 网络组件命名空间 | 100 | varchar | default |  |
| 4. | REPLICAS | INTEGER | N | 副本数量 | 4 | int | 1 |  |
| 5. | LABELNAME | VARCHAR | N | 标签名称 | 100 | varchar |  |  |
| 6. | LABELVALUE | VARCHAR | N | 标签值 | 300 | varchar |  |  |
| 7. | INITVALUE | VARCHAR | N | 对应的组件元素的初始值 | 20 | varchar |  |  |
| 8. | REALNAME | VAARCHAR | N | 真实值 | 50 | varchar | default | 无效 |
| 9. | RELATIVEID | VARCHAR | Y | 所属的应用ID | 40 | varchar |  |  |
| 10. | CREATEUSERID | VARCHAR | N | 创建者ID | 40 | varchar |  |  |
| 11. | CREATEUSERNO | VARCHAR | N | 创建者账号 | 10 | varchar |  |  |
| 12. | CREATEDATETIME | VARCHAR | N | 创建时间 | 20 | varchar |  |  |
| 13. | UPDATEUSERID | VARCHAR | N | 修改者ID | 40 | varchar |  |  |
| 14. | UPDATEUSERNO | VARCHAR | N | 修改者账号 | 10 | varchar |  |  |
| 15. | UPDATEDATETIME | VARCHAR | N | 修改时间 | 20 | varchar |  |  |
| 16. | INTISDELETE | INTEGER | N | 是否（1/0）被删除 |  | int |  |  |
| 17. | INTISACTIVE | INTEGER | N | 状态是否（1/0）正常 |  | int |  |  |
| 18. | COMPANYID | VARCHAR | N | 所属公司ID | 40 | varchar |  |  |

**3.3.1.7 网络端口表说明**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | DB.SERVICEPORT | | | | | | |
| 表说明 | | 网络组件端口配置数据 | | | | | | |
| 序号 | 字段名 | 类型 | 是否主键 | 字段说明 | 长度 | 栏位格式 | 默认值 | 备注 |
| 1. | ID | VARCHAR | Y | 网络端口编号 | 40 | varchar |  |  |
| 2. | SERVICENAME | VARCHAR | N | 所属网络组件名称 | 50 | varchar |  |  |
| 3. | TYPE | VARCHAR | N | 端口类型 | 20 | varchar |  |  |
| 4. | INTERPORT | VARCHAR | N | 内部端口 | 20 | varchar |  |  |
| 5. | TARGETPORT | VARCHAR | N | 目标端口 | 30 | varchar |  |  |
| 6. | NODEPORT | VARCHAR | N | 节点端口 | 20 | varchar |  |  |
| 7. | PORTPOTOCOL | VARCHAR | N | 端口协议 | 10 | varchar |  |  |
| 8. | SERVICEID | VARCHAR | N | 创建后视图对应的组件元素的值 | 40 | varchar |  |  |
| 9. | RELATIVEID | VARCHAR | N | 所属的应用ID | 40 | varchar |  |  |
| 10. | CREATEUSERID | VARCHAR | N | 创建者ID | 40 | varchar |  |  |
| 11. | CREATEUSERNO | VARCHAR | N | 创建者账号 | 10 | varchar |  |  |
| 12. | CREATEDATETIME | VARCHAR | N | 创建时间 | 20 | varchar |  |  |
| 13. | UPDATEUSERID | VARCHAR | N | 修改者ID | 40 | varchar |  |  |
| 14. | UPDATEUSERNO | VARCHAR | N | 修改者账号 | 10 | varchar |  |  |
| 15. | UPDATEDATETIME | VARCHAR | N | 修改时间 | 20 | varchar |  |  |
| 16. | INTISDELETE | INTEGER | N | 是否（1/0）被删除 |  | int |  |  |
| 17. | INTISACTIVE | INTEGER | N | 状态是否（1/0）正常 |  | int |  |  |
| 18. | COMPANYID | VARCHAR | N | 所属公司ID | 40 | varchar |  |  |

**3.3.1.8 应用数据库表说明**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | DB.APPLICATION | | | | | | |
| 表说明 | | 平台部署应用的信息 | | | | | | |
| 序号 | 字段名 | 类型 | 是否主键 | 字段说明 | 长度 | 栏位格式 | 默认值 | 备注 |
| 1. | ID | VARCHAR | Y | 应用编号 | 40 | varchar |  | 唯一标识应用 |
| 2. | APPNAME | VARCHAR | N | 应用名称 | 40 | varchar |  |  |
| 3. | CREATEUSERID | VARCHAR | N | 创建者ID | 50 | varchar |  |  |
| 4. | CREATEUSERNO | VARCHAR | N | 创建者账号 | 10 | varchar |  |  |
| 5. | CREATEDATETIME | VARCHAR | N | 创建时间 | 20 | varchar |  |  |
| 6. | UPDATEUSERID | VARCHAR | N | 修改者ID | 40 | varchar |  |  |
| 7. | UPDATEUSERNO | VARCHAR | N | 修改者账号 | 10 | varchar |  |  |
| 8. | UPDATEDATETIME | VARCHAR | N | 修改时间 | 20 | varchar |  |  |
| 9. | INTISDELETE | INTEGER | N | 是否（1/0）被删除 |  | int |  |  |
| 10. | INTISACTIVE | INTEGER | N | 状态是否（1/0）正常 |  | int |  |  |
| 11. | COMPANYID | VARCHAR | N | 所属公司ID | 40 | varchar |  |  |

#### 3.3.1.4其他表文件见数据库设计文档

## 销售电子商务系统平台详细设计说明

### 4.1服务编排

#### 4.1.1模块架构

**4.1.1.1前后端文件**

|  |  |
| --- | --- |
| Jsp Name | Description |
| ServiceArrange/createContainerPort.jsp | 添加容器端口弹窗 |
| ServiceArrange/createDevelopment.jsp | 创建应用模块节点弹窗 |
| ServiceArrange/createEnv.jsp | 添加环境变量弹窗 |
| ServiceArrange/createService.jsp | 创建网络组件节点弹窗 |
| ServiceArrange/createServicePort.jsp | 添加网络端口弹窗 |
| ServiceArrange/main.jsp | 服务编排主页面 |
| ServiceArrange/sureArrange.jsp | 确认微服务部署设置弹窗 |
| ServiceArrange/updateArrange.jsp | 更新微服务部署设置弹窗 |
| ServiceArrange/updateDeployment.jsp | 更新应用模块设置弹窗 |
| ServiceArrange/updateService.jsp | 更新网络组件配置弹窗 |
| ServiceArrange/updateServicePort.jsp | 更新网路端口配置弹窗 |

**实体类**

|  |  |
| --- | --- |
| Class Name | Description |
| K8sService.java | 网络组件的实体类 |
| ServicePort.java | 网络端口的实体类 |
| Deployment.java | 部署模块的实体类 |
| Pod.java | pod的实体类 |
| Container.java | 容器的实体类 |
| ContainerPort.java | 容器端口的实体类 |
| Environment.java | 容器环境变量的实体类 |
| Application.java | 部署应用的实体类 |

**数据库读写**

|  |  |
| --- | --- |
| Class Name | Description |
| k8sServiceMapper.xml | 网络组件的配置信息的查询/添加/修改等SQL语句 |
| ServicePortsMapper.xml | 网络组件端口的配置信息的查询/添加/修改等SQL语句 |
| DeploymentMapper.xml | 部署模块的配置信息的查询/添加/修改等SQL语句 |
| PodMapper.xml | pod的配置信息的查询/添加/修改等SQL语句 |
| ContainerMapper.xml | 容器的配置信息的查询/添加/修改等SQL语句 |
| ContainerPortMapper.xml | 容器端口的查询/添加/修改等SQL语句 |
| EnvironmentMapper.xml | 容器环境变量的查询/添加/修改等SQL语句 |
| ApplicationMapper.xml | 部署应用信息的查询/添加/修改等SQL语句 |

**存储层**

|  |  |
| --- | --- |
| Class Name | Description |
| k8sServiceDao.java | 网络组件配置存储层 |
| ServicePortsDao.java | 网络组件端口配置存储层 |
| DeploymentDao.java | 部署模块存储层 |
| PodDao.java | Pod存储层 |
| ContainerDao.java | 容器存储层 |
| ContainerPortDao.java | 容器端口存储层 |
| EnvironmentDao.java | 容器环境变量存储层 |
| ApplicationDao.java | 应用存储层 |

**服务层**

|  |  |
| --- | --- |
| Class Name | Description |
| k8sServiceService.java | 网络组件配置服务层 |
| ServicePortsService.java | 网络组件端口配置服务层 |
| DeploymentService.java | 部署模块服务层 |
| PodService.java | pod服务层 |
| ContainerService.java | 容器服务层 |
| ContainerPortService.java | 容器端口服务层 |
| EnvironmentService.java | 容器环境变量服务层 |
| ApplicationService.java | 应用服务层 |

**控制层**

|  |  |
| --- | --- |
| Class Name | Description |
| ServiceArrangeController.java | 控制层 |

**k8s控制器部署配置模板文件**

|  |  |
| --- | --- |
| File Name | Description |
| static/k8s/deploymentTemplate.ftl | 应用模块配置文件模板 |
| static/k8s/serviceTemplate.ftl | 网络组件配置文件模板 |

**编排视图界面插件文件**

|  |  |
| --- | --- |
| File Name | Description |
| static/qunee | qunee网络服务器授权插件文件，权限2-3个月 |

**4.1.1.2类说明**

**实体类**

|  |  |  |
| --- | --- | --- |
| Class Name | 类成员 | 说明 |
| K8sService.java | +initValue:String | 网络组件元素的初始值，用于识别网络组件实体对应在视图中的位置 |
| +relativeID:String | 应用ID用于识别网络组件属于哪个应用 |
| +serviceName：String | 网络服务名称 |
| +serviceNameSpace:String | 弃用属性，默认值为default |
| +replicas:String | 副本数量，默认值为1 |
| +labelName:String | 标签名称 |
| +labelValue:String | 标签值 |
| +realName:String | 弃用属性，默认值为default |
| +getInitValue():String  -initValue:String | 获取initValue的值 |
| +getRelativeID():String  -relativeID:String | 获取relativeID的值 |
| +getServceNam():String  -serviceName:String | 获取serviceName的值 |
| +getServceNamespace():String  -serviceNamespace:String | 获取serviceNamespace的值 |
| +getReplicas():String  -replicas:String | 获取replicas的值 |
| +getLabelName():String  -labelName:String | 获取labelName的值 |
| +getLabelValue():String  -labelValue:String | 获取labelValue的值 |
| +getRealName():String  -realName:String | 获取realName的值 |
| +setInitValue():void  -initValue:String | 修改initValue的值 |
| +setRelativeID():void  -relativeID:String | 修改relativeID的值 |
| +setServceNam():void  -serviceName:String | 修改serviceName的值 |
| +setServceNamespace():void  -serviceNamespace:String | 修改serviceNamespace的值 |
| +setReplicas():void  -replicas:String | 修改replicas的值 |
| +setLabelName():void  -labelName:String | 修改labelName的值 |
| +setLabelValue():void  -labelValue:String | 修改labelValue的值 |
| +setRealName():void  -realName:String | 修改realName的值 |
| ServicePort.java | +serviceID:String | 端口所属网络组件ID |
| +serviceName:String | 端口所属网络组件名称 |
| +relativeID:String | 端口所属应用ID |
| +Type:String | 端口类型 |
| +interPort:String | 内部端口号 |
| +targetPort:String | 目标端口号 |
| +nodePort:String | 节点映射端口号 |
| +portPotocol:String | 弃用属性，默认值为TCP |
| +getServiceID():String  -serviceID:String | 获取serviceID的值 |
| +getServiceName():String  -serviceName:String | 获取serviceID的值 |
| +getRelativeID():String  -relativeID:String | 获取relativeID的值 |
| +getType():String  -Type:String | 获取Type的值 |
| +getInterPort():String  -interPort:String | 获取interPort的值 |
| +getTargetPort():String  -targetPort:String | 获取targetPort的值 |
| +getNodePort():String  -nodePort:String | 获取nodePort的值 |
| +getPortPotocol():String  -portPotocol:String | 获取portPotocol的值 |
| +setServiceID():void  -serviceID:String | 修改serviceID的值 |
| +setServiceName():void  -serviceName:String | 修改serviceID的值 |
| +setRelativeID():void  -relativeID:String | 修改relativeID的值 |
| +setType():void  -Type:String | 修改Type的值 |
| +setInterPort():void  -interPort:String:String | 修改interPort的值 |
| +setTargetPort():void  -targetPort:String | 修改targetPort的值 |
| +setNodePort():void  -nodePort:String:String | 修改nodePort的值 |
| +setPortPotocol():void  -portPotocol:String:String | 修改portPotocol的值 |
| Deployment.java | +deploymentName:String | 应用模块名称 |
| +deploymentNamespace:String | 应用模块命名空间 |
| +replicas:String | 副本数量，默认值为1 |
| +labelName:String | 标签名称 |
| +labelValue:String | 标签值 |
| +initValue:String | 弃用属性，默认值为default |
| +relativeId:String | 应用ID用于识别应用模块属于哪个应用 |
| +realName:String | 弃用属性，默认值为default |
| +getInitValue():String  -initValue:String | 获取initValue的值 |
| +getRelativeID():String  -relativeID:String | 获取relativeID的值 |
| +getdeploymentName():String  -deploymentName:String | 获取deploymentName的值 |
| +getdeploymentNamespace():String  -deploymentNamespace:String | 获取deploymentNamesp的值 |
| +getReplicas():String  -replicas:String | 获取replicas的值 |
| +getLabelName():String  -labelName:String | 获取labelName的值 |
| +getLabelValue():String  -labelValue:String | 获取labelValue的值 |
| +getRealName():String  -realName:String | 获取realName的值 |
| +setInitValue():void  -initValue:String | 修改initValue的值 |
| +setRelativeID():void  -relativeID:String | 修改relativeID的值 |
| +setdeploymentName():void  -deploymentName:String | 修改deploymentName的值 |
| +setdeploymentNamespace():void  -deploymentNamespace:String | 修改deploymentNamesp的值 |
| +setReplicas():void  -replicas:String | 修改replicas的值 |
| +setLabelName():void  -labelName:String | 修改labelName的值 |
| +setLabelValue():void  -labelValue:String | 修改labelValue的值 |
| +setRealName():void  -realName:String | 修改realName的值 |
| Pod.java | +deploymentName:String | 应用模块名称 |
| +podName:String | pod名称 |
| +podNamespace:String | pod命名空间 |
| +podLabelName:String | 标签名称 |
| +labelValue:String | 标签值 |
| +relativeId:String | 应用ID用于识别容器组属于哪个应用 |
| +realName:String | 弃用属性，默认值为default |
| +getRelativeId():String  -relativeId:String | 获取relativeId的值 |
| +getPodName():String  -podName:String | 获取podName的值 |
| +getPodNamespace():String  -podNamespace:String | 获取podNamesp的值 |
| +getPodLabelName():String  -podLabelName:String | 获取podLabelName的值 |
| +getPodLabelValue():String  -podLabelValue:String | 获取podLabelValue的值 |
| +getRealName():String  -realName:String | 获取realName的值 |
| +setRelativeID():void  -relativeID:String | 修改relativeID的值 |
| +setPodName():void  -podName:String | 修改podName的值 |
| +setPodNamespace():void  -podNamespace:String | 修改podNamesp的值 |
| +setpodLabelName():void  -podlabelName:String | 修改podLabelName的值 |
| +setpodLabelValue():void  -podLabelValue:String | 修改podLabelValue的值 |
| +setRealName():void  -realName:String | 修改realName的值 |
| ContainerPort.java | +containerName:String | 端口所属容器名称 |
| +portName:String | 端口名称 |
| +deploymentId:String | 端口所输deployment的ID |
| +relativeID:String | 端口所属应用ID |
| +portNumber:String | 端口号 |
| +portPotocol:String | 弃用属性，默认值为TCP |
| +getContainerName():String  -containerName:String | 获取containerName的值 |
| +getPortName():String  -portName:String | 获取portName的值 |
| +getPortNumber():String  -portNumber:String | 获取portNumber的值 |
| +getPortPotocol():String  -portPotocol:String | 获取portPotocol的值 |
| +setDeploymentId():void  -deploymentId:String | 修改deploymentId的值 |
| +setRelativeID():void  -relativeID:String | 修改relativeID的值 |
| +setPortNumber():void  -portNumber:String | 修改portNumber的值 |
| Environment.java | +containerName:String | 环境变量所属容器名称 |
| +envvalName:String | 环境变量名称 |
| +envvalValue:String | 环境变量值 |
| +deploymentId:String | 端口所属deployment的ID |
| +relativeID:String | 端口所属应用ID |
| +getContainerName():String  -containerName:String | 获取containerName的值 |
| +getEnvvalName():String  -envvalName:String | 获取envvalName的值 |
| +getEnvvalValue():String  -envvalValue:String | 获取envvalValue的值 |
| +getDeploymentId():void  -deploymentId:String | 获取deploymentId的值 |
| +getRelativeID():void  -relativeID:String | 获取relativeID的值 |
| +setContainerName():String  -containerName:String | 修改containerName的值 |
| +setEnvvalName():String  -envvalName:String | 修改envvalName的值 |
| +setEnvvalValue():String  -envvalValue:String | 修改envvalValue的值 |
| +setDeploymentId():void  -deploymentId:String | 修改deploymentId的值 |
| +setRelativeID():void  -relativeID:String | 修改relativeID的值 |
| Application.java | +appName:String | 应用名称 |
| +Type:String | 默认值为default |
| +getType():String  -Type:String | 获取Type的值 |
| +getAppName():String  -appName:String | 获取appName的值 |
| +setType():void  -type:String | 修改Type的值 |
| +setAppName():void  -appName:String | 修改appName的值 |

**存储层**

|  |  |  |
| --- | --- | --- |
| Class Name | 类成员 | 说明 |
| k8sServiceDao.java | +sqlSession：Sqlsession | 用于执行数据库sql命令的对象 |
| +findOne():k8sService  -params:Map<String,Object>  -service：k8sService | 根据relativeID、initValue查找对应网络组件的信息 |
| +findAll():List<Map<String,Object>>  -relativeID:String  -list:List<Map<String,Object>> | 查找应用所有的网络组件信息 |
| +create():int  -service:k8sService  -sqlSession.insert(String,k8sService):int | 新建并插入一条网络组件信息 |
| +delete():int  -params:Map<String,Object>  -sqlSession.delete(String,Map<String,Object>):int | 根据自身ID删除一条网络组件的信息 |
| +deleteAll():int  -relativeID:String  -sqlSession.delete(String,String):int | 根据relativeID删除一条网络组件的信息 |
| +update():int  -service:k8sService  -sqlSession.update(String,k8sService):int | 更新一条网络组件的信息 |
| ServicePortsDao.java | +sqlSession：Sqlsession | 用于执行数据库sql命令的对象 |
| +findOne():ServicePort  -params:Map<String,Object>  -sqlSession.selectOne(String,Map<String,Object>):int | 查找一条网络端口的配置信息 |
| +findAll():List<Map<String,Object>>  -params:Map<String,Object>  -list:List<Map<String,Object>> | 返回网络组件的所有端口 |
| +create():int  -servicePorts:ServicePorts  -sqlSession.insert(String,ServicePorts):int | 新建并插入一条网络端口信息 |
| +delete():int  -id:String  -sqlSession.delete(String,String):int | 根据id删除端口 |
| +deleteServicePort():int  -params:Map<String,Object>  -sqlSession.delete(String,Map<String,Object>):int | 和网络组件一起删除 |
| +deleteAll():int  -relativeID:String  -sqlSession.delete(String,String):int | 和应用一起删除 |
| DeploymentDao.java | +sqlSession：Sqlsession | 用于执行数据库sql命令的对象 |
| +findOne():Deployment  -params:Map<String,Object>  -sqlSession.selectOne(String,Map<String,Object>):int | 根据relativeID、initValue查找对应应用模块的信息 |
| +findAll():List<Map<String,Object>>  -relativeID:String  -list:List<Map<String,Object>> | 查找应用所有的应用模块信息 |
| +create():int  -deployment:Deployment  -sqlSession.insert(String,Deployment):int | 新建并插入一条应用模块信息 |
| +delete():int  -params:Map<String,Object>  -sqlSession.delete(String,Map<String,Object>):int | 根据自身ID删除一条应用模块信息 |
| +deleteAll():int  -relativeID:String  -sqlSession.delete(String,String):int | 根据relativeID删除一条应用模块的信息 |
| +update():int  -deployment:Deployment  -sqlSession.update(String,Deployment):int | 更新一条应用模块的信息 |
| PodDao.java | +sqlSession：Sqlsession | 用于执行数据库sql命令的对象 |
| +findOne():Pod  -id:String  -sqlSession.selectOne(String,String):int | 根据relativeID、initValue查找对应容器组信息 |
| +create():int  -pod:Pod  -sqlSession.insert(String,Pod):int | 新建并插入一条容器组信息 |
| +delete():int  -id:String  -sqlSession.delete(String,String):int | 根据自身ID删除一条容器组信息 |
| +deleteAll():int  -relativeID:String  -sqlSession.delete(String,String):int | 根据relativeID删除一条容器组信息 |
| +update():int  -pod:Pod  -sqlSession.update(String,Pod):int | 更新一条容器组的信息 |
| ContainerDao.java | +sqlSession：Sqlsession | 用于执行数据库sql命令的对象 |
| +findOne():Container  -id:String  -sqlSession.selectOne(String,String):int | 根据relativeID、initValue查找对应容器信息 |
| +create():int  -Container:Container  -sqlSession.insert(String,Container):int | 新建并插入一条容器信息 |
| +delete():int  -id:String  -sqlSession.delete(String,String):int | 根据自身ID删除一条容器信息 |
| +deleteAll():int  -relativeID:String  -sqlSession.delete(String,String):int | 根据relativeID删除一条容器信息 |
| +update():int  -container:Container  -sqlSession.update(String,Container):int | 更新一条容器的信息 |
| ContainerPortDao.java | +sqlSession：Sqlsession | 用于执行数据库sql命令的对象 |
| +findOne():Container  -params:Map<String,Object>  -sqlSession.selectOne(String,Map<String,Object>):int | 查找一条容器端口的配置信息 |
| +update():int  -container:Container  -sqlSession.update(String,Container) | 修改容器端口 |
| +create():int  -container:Container  -sqlSession.insert(String,Container):int | 新建并插入一条容器端口信息 |
| +delete():int  -id:String  -sqlSession.delete(String,String):int | 根据id删除端口 |
| +deleteAll():int  -relativeID:String  -sqlSession.delete(String,String):int | 删除所有容器端口信息 |
| EnvironmentDao.java | +sqlSession：Sqlsession | 用于执行数据库sql命令的对象 |
| +findOne():Environment  -params:Map<String,Object>  -sqlSession.selectOne(String,Map<String,Object>):Environment | 查找一条容器环境变量的配置信息 |
| +findAll():List<Map<String,Object>>  -params:Map<String, Object>  -list:List<Map<String,Object>> | 查找容器所有的环境变量信息 |
| +update():int  -container:Container  -sqlSession.update(String,Container) | 修改容器环境变量 |
| +create():int  -environment:Environment  -sqlSession.insert(String,Environment):int | 新建并插入一条容器环境变量信息 |
| +delete():int  -id:String  -sqlSession.delete(String,String):int | 根据id删除环境变量 |
| +deleteAll():int  -relativeID:String  -sqlSession.delete(String,String):int | 删除所有容器环境变量信息 |
| +deleteAllThis():int  -params:Map<String, Object>  -sqlSession.delete(String,Map<String, Object>):int | 根据relativeID、initValue删除当前容器所有环境变量信息 |
| ApplicationDao.java | +sqlSession：Sqlsession | 用于执行数据库sql命令的对象 |
| +findOne():Application  -appName:String  -sqlSession.selectOne(String):Application | 查看某个应用信息 |
| +findAll():List<Map<String,Object>>  -list:List<Map<String,Object>> | 查看所有应用的信息 |
| +update():int  -id:String  -sqlSession.update(String,String):int | 更新一条应用信息 |
| +create():int  -app:Application  -sqlSession.insert(String,Application):int | 新建并插入一条应用信息 |
| +delete():int  -id:String  -sqlSession.delete(String,String):int | 根据自身id删除一条应用信息 |

**服务层**

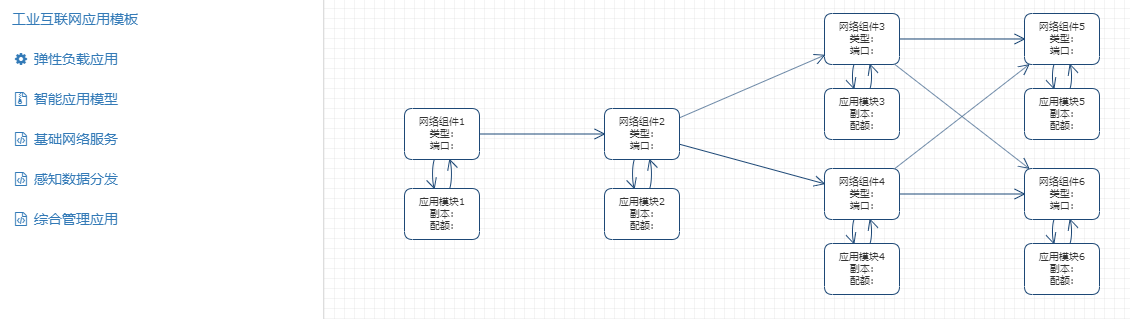
|  |  |  |
| --- | --- | --- |
| Class Name | 类成员 | 说明 |
| k8sServiceService.java | +k8sServiceDao:k8sServiceDao | 操作数据库的Dao层对象 |
| +create():int  -k8sService:k8sService  -k8sServiceDao.create(k8sService):int | 调用Dao层create函数新建并插入一条网络组件信息 |
| +delete():int  -params:Map<String,Object>  -k8sServiceDao.delete(Map<String,Object>):int | 调用Dao层delete函数根据自身ID删除一条网络组件的信息 |
| +deleteAll():int  -initValue:String  -k8sServiceDao.deleteAll(String):int | 调用Dao层deleteAll函数根据relativeID删除一条网络组件的信息 |
| +update():int  -k8sService:k8sService  -k8sServiceDao.update(k8sService):int | 调用Dao层update函数更新一条网络组件的信息 |
| +findOne():k8sService  -params:Map<String,Object>  -k8sServiceDao.findOne(Map<String,Object>):k8sService | 调用Dao层findOne函数根据relativeID、initValue查找对应网络组件的信息 |
| +findAll():List<Map<String,Object>>  -relativeID:String  -list:List<Map<String,Object>> | 调用Dao层findAll函数查找应用所有的网络组件信息 |
| ServicePortsService.java | +servicePortsDao:ServicePortsDao | 操作数据库的Dao层对象 |
| +create():int  -servicePorts:ServicePorts  -ServicePortsDao.create(ServicePorts) |  |
| +delete():int  -id:String  -ServicePortsDao.delete(String):int | 调用Dao层delete函数根据id删除端口 |
| +deleteServicePort():int  -params:Map<String,Object>  -ServicePortsDao.deleteServicePort(Map<String,Object>):int | 调用Dao层deleteServicePort函数和网络组件一起删除 |
| +deleteAll():int  -relativeID:String  -ServicePortsDao.deleteAll(String):int | 调用Dao层deleteAll函数和应用一起删除 |
| +findOne():ServicePorts  -params:Map<String,Object>  -ServicePortsDao.findOne(Map<String,Object>):ServicePorts | 调用Dao层findOne函数根据relativeID、initValue查找对应网络端口的信息 |
| +findAll():List<Map<String,Object>>  -params:Map<String,Object>  -list:List<Map<String,Object>> | 调用Dao层findAll函数查找应用所有的网络端口信息 |
| DeploymentService.java | +deploymentDao:DeploymentDao | 操作数据库的Dao层对象 |
| +create():int  -deployment:Deployment  -deploymentDao.create(Deployment):int | 调用Dao层create函数新建并插入一条应用模块信息 |
| +delete():int  -id:String  -deploymentDao.delete(String):int | 调用Dao层delete函数根据自身ID删除一条应用模块的信息 |
| +deleteAll():int  -relativeId:String  -deploymentDao.deleteAll(String):int | 调用Dao层deleteAll函数根据relativeID删除所有应用模块的信息 |
| +update():int  -deployment:Deployment  -deploymentDao.update(Deployment):int | 调用Dao层update函数更新一条应用模块的信息 |
| +findOne():Deployment  -params:Map<String,Object>  -deploymentDao.findOne(Map<String,Object>):Deployment | 调用Dao层findOne函数根据relativeID、initValue查找对应应用模块的信息 |
| +findAll():List<Map<String,Object>>  -relativeID:String  -list:List<Map<String,Object>> | 调用Dao层findAll函数查找应用所有的应用模块信息 |
| PodService.java | +podDao:PodDao | 操作数据库的Dao层对象 |
| +create():int  -pod:Pod  -podDao.create(Pod):int | 调用Dao层create函数新建并插入一条容器组信息 |
| +delete():int  -id:String  -podDao.delete(String):int | 调用Dao层delete函数根据自身ID删除一条容器组的信息 |
| +deleteAll():int  -relativeId:String  -podDao.deleteAll(String):int | 调用Dao层deleteAll函数根据relativeID删除所有容器组的信息 |
| +update():int  -pod:Pod  -podDao.update(Pod):int | 调用Dao层update函数更新一条容器组的信息 |
| +findOne():Pod  -id:String  -podDao.findOne(String):Pod | 调用Dao层findOne函数根id查找对应容器组的信息 |
| ContainerService.java | +containerDao:ContainerDao | 操作数据库的Dao层对象 |
| +create():int  -container:Container  -containerDao.create(Container):int | 调用Dao层create函数新建并插入一条容器信息 |
| +delete():int  -id:String  -containerDao.delete(String):int | 调用Dao层delete函数根据自身ID删除一条容器的信息 |
| +deleteAll():int  -relativeId:String  -containerDao.deleteAll(String):int | 调用Dao层deleteAll函数根据relativeID删除所有容器的信息 |
| +update():int  -container:Container  -containerDao.update(Container):int | 调用Dao层update函数更新一条容器的信息 |
| +findOne():Container  -id:String  -containerDao.findOne(String):Container | 调用Dao层findOne函数根id查找对应容器的信息 |
| ContainerPortService.java | +containerPortDao:ContainerPortDao | 操作数据库的Dao层对象 |
| +create():int  -containerPort:ContainerPort  -ContainerPortDao.create(ContainerPort) | 调用Dao层create函数新建并插入一条容器端口信息 |
| +delete():int  -id:String  -ContainerPortDao.delete(String):int | 调用Dao层delete函数根据id删除端口 |
| +deleteAllThis():int  -params:Map<String, Object>  -ContainerPortDao.deleteAllThis(Map<String, Object>):int | 调用Dao层deleteAllThis函数删除当前容器所有端口信息 |
| +deleteAll():int  -relativeID:String  -ContainerPortDao.deleteAll(String):int | 调用Dao层deleteAll函数删除所有容器端口 |
| +findOne():ContainerPort  -params:Map<String,Object>  -ContainerPortDao.findOne(Map<String,Object>):ContainerPort | 调用Dao层findOne函数根据relativeID、initValue查找对应容器端口的信息 |
| +findAll():List<Map<String,Object>>  -params:Map<String,Object>  -list:List<Map<String,Object>> | 调用Dao层findAll函数查找应用所有的容器端口信息 |
| EnvironmentService.java | +environmentDao:EnvironmentDao | 操作数据库的Dao层对象 |
| +create():int  -environment:Environment  -environmentDao.create(Environment) | 调用Dao层create函数新建并插入一条容器环境变量信息 |
| +delete():int  -id:String  -EnvironmentDao.delete(String):int | 调用Dao层delete函数根据id删除环境变量 |
| +deleteAllThis():int  -params:Map<String, Object>  -environmentDao.deleteAllThis(Map<String, Object>):int | 调用Dao层deleteAllThis函数删除当前容器所有环境变量信息 |
| +deleteAll():int  -relativeID:String  -EnvironmentDao.deleteAll(String):int | 调用Dao层deleteAll函数删除所有容器环境变量 |
| +findOne():Environment  -params:Map<String,Object>  -environmentDao.findOne(Map<String,Object>):Environment | 调用Dao层findOne函数根据relativeID、initValue查找对应容器环境变量的信息 |
| +findAll():List<Map<String,Object>>  -params:Map<String,Object>  -list:List<Map<String,Object>> | 调用Dao层findAll函数查找应用所有的容器环境变量 |
| ApplicationService.java | +applicationDao:ApplicationDao | 操作数据库的Dao层对象 |
| +findOne():Application  -appName:String  -ApplicationDao.findtOne(String):Application | 调用Dao层的findOne函数查看某个应用信息 |
| +findAll():List<Map<String,Object>>  -list:List<Map<String,Object>> | 调用Dao层的findAll函数查看所有应用的信息 |
| +update():int  -id:String  -ApplicationDao.update(String,String):int | 调用Dao层的update函数更新一条应用信息 |
| +create():int  -app:Application  -ApplicationDao.create(String,Application):int | 调用Dao层的create函数新建并插入一条应用信息 |
| +delete():int  -id:String  -ApplicationDao.delete(String,String):int | 调用Dao层的delete函数根据自身id删除一条应用信息 |

**控制层**

|  |  |  |
| --- | --- | --- |
| Class Name | 类成员 | 说明 |
| ServiceArrangeController.java | +MAIN:String | main.jsp的路径 |
| +CREATESERVICE:String | createService的路径 |
| +UPDATESERVICE:String | updateService.jsp的路径 |
| +UPDATEDEPLOYMENT:String | updateDeployment.jsp的路径 |
| +CREATEDDEPLOYMENT:String | ceateDeployment.jsp的路径 |
| +CREATECONTAINER:String | createContainer.jsp的路径 |
| +CREATEENV:String | createEnv.jsp的路径 |
| +CREATEPORT:String | createServicePort的路径 |
| +SRUEARRANGE:String | sureArrange.jsp的路径 |
| +UPDATEARRANGE:String | updateArrange.jsp的路径 |
| +servicePortsService:servicePortsService | 网络端口服务调用对象 |
| +serviceService:k8sServiceService | 网络组件服务调用对象 |
| +appService:ApplicationService | 应用服务调用对象 |
| +deploymentservice:DeploymentService | 应用模块服务调用对象 |
| +podService:PodService | 容器组服务调用对象 |
| +containerService:ContainerService | 容器服务调用对象 |
| +containerPortService:ContainerPortService | 容器端口服务调用对象 |
| +environmentService:EnvironmentService | 容器环境变量服务调用对象 |
| +httpClientTool:HttpClientTool | Http接口调用对象 |
| +list():String  -page:Page  -model:Model  -MAIN:String | 通过HTTP访问k8s接口获取应用服务部署的状态信息，并刷新到main.jsp |
| +preCreate():String  -model:Model  -nodeName:String  -type:String  -relativeID:String | 单击微服务拓扑上的节点判断需要创建的应用模块或者网络服务，并准备相应的数据。 |
| +preUpdate():String  -model:Model  -nodeName:String  -type:String  -relativeID:String | 双击微服务拓扑上的节点判断需要更新的应用模块或者网络服务，并准备相应的数据。 |
| +createNode():String  -model:Model  -service:k8sService | 新建网络服务的配置信息并写入数据库 |
| +updateService():String  -model:Model  -service:k8sService | 更新数据库中对应的网络服务的信息 |
| +insertServicePort():String  -serviceID:String  -serviceName:String  -relativeID:String  -Type:String  -interPort:String  -targetPort:String  -id:String  -nodePort:String | 新建网络端口的配置信息并插入数据库 |
| +createDeployment():String  -model:Model -createDeploymentEntity:CreateDeployment | 新建应用模块的信息并写入数据库 |
| +updateDeployment():String  -model:Model -createDeploymentEntity:CreateDeployment | 更新数据库中对应应用模块的配置信息 |
| +preCreatePort():String  -model:Model  -CREATEPORT:String | 单击添加按钮，弹出网络端口配置窗口 |
| +createPort():String  -model:Model  -servicePorts:ServicePorts | 新建网络端口配置并写入数据库 |
| +updatePort():String  -serviceName:String  -type:String  -relativeID:String | 更新网络端口的配置信息 |
| +preSureArrange():String  -model:Model  -relativeID:String  -tptype:String  -SRUEARRANGE:String | 应用的微服务配置完成后单击确认弹出确认部署弹窗 |
| +preUpdateArrange():String  -model:Model  -relativeID:String  -tptype:String  -UPDATEARRANGE:String | 在更新界面单击确认按钮弹出确认更新应用部署弹窗 |
| +sureArrangeTest():String  -model:Model  -app:Application  -request:HttpServletRequest | 确认部署配置，按照微服务拓扑中的配置信息部署应用所需的应用模块和网络服务 |
| +convertFileToString():String  -path:String  -fileAsString:String | 文件内容转换成字符串 |
| +updateArrange()  -model:Model  -app:Application  -request:HttpServletRequest | 更新部署配置，按照更新后的微服务拓扑中的配置信息部署应用模块和网络服务 |
| +preCreateContainerPort():String  -model:Model  -CREATECONTAINERPORT | 单击添加按钮，弹出容器端口配置弹窗 |
| +createContainerPort():String  -deploymentId:String  -relativeId:String  -containerName:String  -portName:String  -portNumber:String  -id:String | 新建容器端口配置信息并写入数据库 |
| +updateContainerPort():String  -deploymentId:String  -relativeId:String  -containerName:String  -portName:String  -portNumber:String  -id:String | 更新容器端口配置信息 |
| +preCreateEnv():String  -model:model  -CREATEENV:String | 单击添加按钮，弹出容器环境变量配置窗口 |
| +createEnv():String  -deploymentId:String  -relativeId:String  -containerName:String  -envvalName:String  -envvalValue:String  -id:String | 新建容器环境变量配置并写入数据库 |
| +updateEnv():String  -deploymentId:String  -relativeId:String  -containerName:String  -envvalName:String  -envvalValue:String  -id:String | 更新容器环境变量配置 |
| +deleteNode():String  -initValue:String  -type:String  -relativeId:String | 删除微服务拓扑上应用模块或者网络服务的配置信息 |
| +stopApplication():String  -relativeId:String | 停止应用在集群上的部署 |
| +deleteApplication():String  -relativeId:String | 在集群和数据库上删除应用 |
| +startApplication():String  -relativeId:String  -request:HttpServletRequest | 重启集群上的应用（可能存在数据丢失） |
| +updateApplicationDeployment():String  -relativeId:String | 更新应用模块的部署配置 |
| +updateApplicationService():String  -relativeId:String | 更新应用网络服务的部署配置 |

#### 4.1.2组件说明

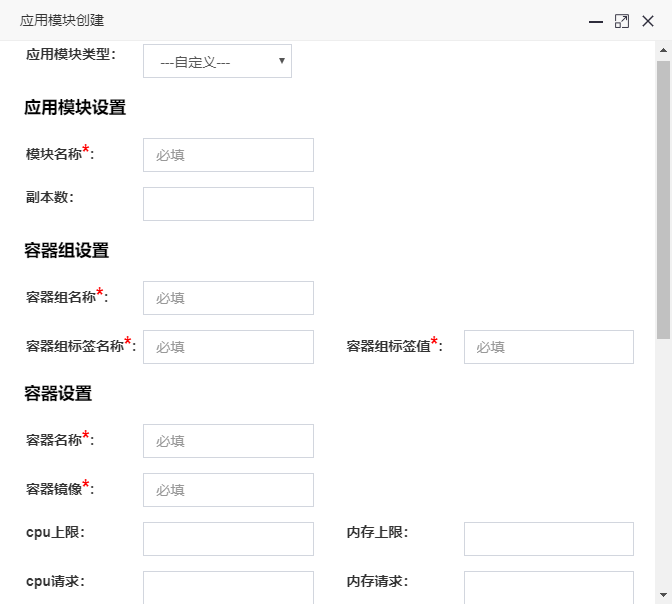
应用模式选择



1.main.jsp中，其实现从工业互联网应用模板中选择一种应用模板并在右侧进行拓扑图显示的功能。

|  |
| --- |
| 画面说明 |
| 用于应用模式的选择，并显示相应拓扑图 。其主要分为弹性负载应用、智能应用模型、基础网络服务、感知数据分发、综合管理应用等几种模式。 |
| 功能说明 |
| 【网络组件】  创建网络组件(service)，点击弹出网络组件的配置界面。  【应用模块】  创建应用模块(deployment)，点击弹出应用模块的配置界面。 |

创建应用部署





2.createDeployment.jsp为应用模块创建界面，负责Deployment的创建功能。

点击容器端口上方的添加按钮弹出新增容器端口界面createContainerPort.jsp。

点击容器环境变量上方的添加按钮弹出新增容器环境变量界面createEnv.jsp。

|  |
| --- |
| 画面说明 |
| 用于应用模块的创建。 |
| 输入项说明 |
| 应用模块类型：提供事先预设的应用设置。 |
| 模块名称：Deployment的名称。 |
| 副本数：部署后生成的副本数。 |
| 容器组名称：相应pod的名称。 |
| 容器组标签名称、容器组标签值：自定义pod的标签。 |
| 容器名称：自定义容器的名称。 |
| 容器镜像：容器使用的镜像地址。 |
| cpu上限、cpu请求、内存上限、内存请求：限制Deployment使用的cpu和内存资源。 |
| 添加（容器端口）：弹出新增容器端口界面。 |
| 添加（容器环境变量）：弹出新增容器环境变量界面。 |
| 功能说明 |
| 【确定】  保存所新增的应用模块内容，并显示在编排模板上。 |

容器端口配置



3.createContainerPort.jsp为新增容器端口界面，其通过点击应用模块创建界面中容器端口上方的添加按钮进入。

|  |
| --- |
| 画面说明 |
| 用于新增需要暴露的端口。 |
| 输入项说明 |
| 端口名称：自定义端口的名称。 |
| 端口号：容器需要监听的端口号。 |
| 功能说明 |
| 【确定】  保存所新增的容器端口，并添加到应用模块创建界面的容器端口列表中。 |

模块环境变量配置



4.createEnv.jsp为新增环境变量界面，其通过点击应用模块创建界面中容器环境变量上方的添加按钮进入。

|  |
| --- |
| 画面说明 |
| 用于新增容器运行前需设置的环境变量。 |
| 输入项说明 |
| 环境变量名称：设置环境变量名。 |
| 环境变量值：设置环境变量值。 |
| 功能说明 |
| 【确定】  保存所新增的容器环境变量，并添加到应用模块创建界面的容器环境变量列表中。 |

部署应用列表

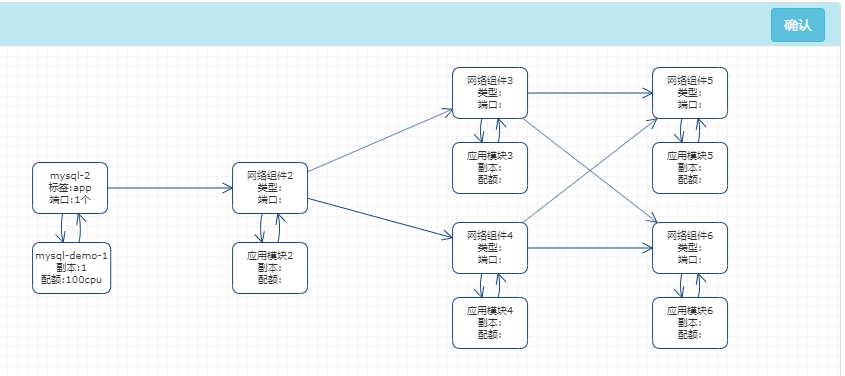
5.main.jsp中，负责显示每一个被部署应用的名称、类型、服务数量、模块数量、创建时间和状态等信息，并提供应用的查看、修改、删除、启动、停止的功能。



|  |
| --- |
| 画面说明 |
| 用于显示应用的部署情况和提供相应修改操作。 |
| 功能说明 |
| 【查看】  在上方的应用编排模板处显示当前应用的应用模块和网络组件信息。  【修改】  对于已被停止的应用方可进行应用模块和网络组件的修改。  【删除】  对于已被停止的应用方可进行应用的删除。  【启动】  对于已被停止的应用进行重新启动。  【停止】  停止正在运行的应用。 |

查看应用

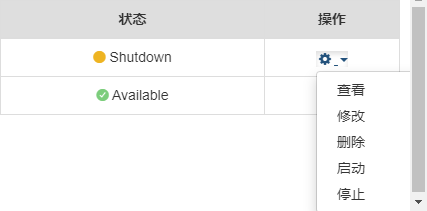
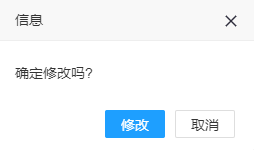
6.main.jsp应用列表的操作列表中，对已部署的应用进行查看。在部署情况中点击需要进行操作的应用右方的操作按钮打开操作菜单，选择查看在应用编排图中显示应用信息。



|  |
| --- |
| 画面说明 |
| 用于显示当前应用的应用模块和网络组件信息。 |

修改应用

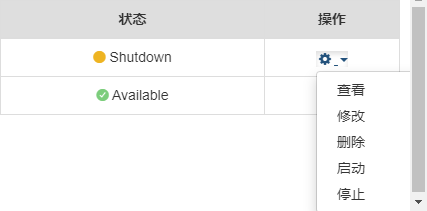
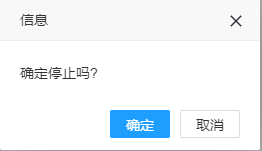
7.main.jsp应用列表的操作列表中，对已停止的应用进行参数修改，修改主要涉及updateDeployment.jsp、updateService.jsp、updateServicePort.jsp、updateArrange.jsp等页面，流程操作与应用创建部分相同。

|  |
| --- |
| 画面说明 |
| 用于进行应用的修改。 |
| 功能说明 |
| 【修改】  要修改应用请先确保应用处于停止状态，否则修改按钮无法点击；确认应用处于停止状态后点击在部署情况中点击需要进行操作的应用右方的操作按钮打开操作菜单，选择修改弹出修改确认窗口并确认，在应用编排图中显示应用信息，操作与应用创建部分相同。 |

停止应用

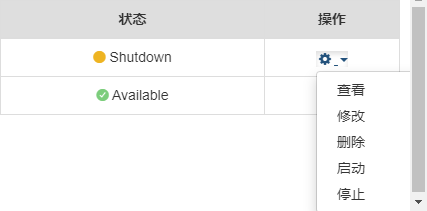
8.main.jsp应用列表的操作列表中，停止正在运行的应用。

|  |
| --- |
| 画面说明 |
| 用于停止正在运行的应用。 |
| 功能说明 |
| 【停止】  点击在部署情况中点击需要进行操作的应用右方的操作按钮打开操作菜单，选择停止弹出停止确认窗口并点击确定以停止应用。 |

应用重启

9.main.jsp应用列表的操作列表中对于已被停止的应用进行重启动。s

|  |
| --- |
| 画面说明 |
| 重新启动已被停止的应用。 |
| 功能说明 |
| 【启动】  要启动应用请先确保应用处于停止状态，否则启动按钮无法点击；确认应用处于停止状态后点击在部署情况中点击需要进行操作的应用右方的操作按钮打开操作菜单，选择启动弹出启动确认窗口并进行确认。应用重启后状态会显示在状态栏中。 |

#### 4.1.3数据库说明

|  |  |
| --- | --- |
| Table ID | Table Name |
| CONTAINER | 容器主表 |
| CONTAINERPORT | 容器端口表 |
| CONTAINERENV | 容器环境变量表 |
| POD | 容器组（微服务）表 |
| DEPLOYMENT | 应用模块表 |
| SERVICE | 网络组件表 |
| SERVICEPORT | 网络端口表 |
| APPLICATION | 应用表 |

#### 4.1.4与其他接口说明

|  |  |
| --- | --- |
| Interface | Description |
| freemarker.template.Configuration | 通过模板自动生成yaml文件 |
| freemarker.template.Template |
| freemarker.template.TemplateException |
| org.apache.commons.httpclient.methods.PostMethod | 通过url发送yaml文件部署组件 |
| org.apache.commons.httpclient.methods.DeleteMethod | 通过url发送yaml文件停止组件 |
| "http://172.18.194.2:8087/apis/apps/v1/namespaces/default/deployments" | deployment操作url |
| "http://172.18.194.2:8087/api/v1/namespaces/default/services" | service操作url |

### 4.2应用状态监控

#### 4.2.1模块架构

|  |  |
| --- | --- |
| Jsp Name | Description |
| ServiceOverview/main.jsp | 服务信息总览 |
| StatusMonitor/main.jsp | 应用状态信息监控 |

|  |  |
| --- | --- |
| Class Name | Class Name |
| ServiceOverview.java | 服务总览的实体类 |
| StatusMonitor.java | 应用状态监控的实体类 |
| NoticeDao.java | 存储层 |
| NoticeService.java | 业务层,编写业务逻辑 |
| NoticeApproveController.java | 控制层 |
| ServiceOverviewMapper.xml | 服务总览的配置信息的查询/添加/修改等SQL语句 |
| StatusMonitorMapper.xml | 应用状态监控的配置信息的查询/添加/修改等SQL语句 |
| ServiceOverviewDao.java | 服务总览配置存储层 |
| StatusMonitorDao.java | 应用状态监控配置存储层 |
| ServiceOverviewService.java | 服务总览配置服务层 |
| StatusMonitorService.java | 应用状态监控配置服务层 |
| ServiceOverviewController.java | 服务总览控制层 |
| StatusMonitorController.java | 应用状态监控控制层 |

**实体类**

|  |  |  |
| --- | --- | --- |
| Class Name | 类成员 | 说明 |
| ServiceOverview.java | +appID:String | 应用ID编号 |
| +createDateTime:String | 创建时间 |
| +appName：String | 应用名称 |
| +getAppID():String  -appID:String | 获取appID的值 |
| +getAppName():String  -appName:String | 获取appName的值 |
| +getCreateDateTime():String  -createDateTime:String | 获取createDateTime的值 |
| +setAppID():void  -appID:String | 修改AppID的值 |
| +setAppName():void  -appName:String | 修改appName的值 |
| +setCreateDateTime():void  -createDateTime:String | 修改createDateTime的值 |
| StatusMonitor.java | +containerImage:String | 容器镜像 |
| +containerName:String | 容器名 |
| +cpuLimit:String | CPU资源限额 |
| +cpuRequest:String | CPU需求 |
| +ID:String | 标识号 |
| +imagePolicy:String | 镜像策略 |
| +memLimit:String | 内存资源限额 |
| +memRequest:String | 内存需求 |
| +readOnly:String | 是否只读 |
| +shareVolumeName:String | 共享卷名 |
| +volumeContainerPath:String | 共享卷容器路径 |
| +getContainerImage():String  -containerImage:String | 获取containerImage的值 |
| +getContainerName():String  -containerName:String | 获取containerName的值 |
| +getCpuLimit():String  -cpuLimit:String | 获取cpuLimit的值 |
| +getCpuRequest():String  -cpuRequest:String | 获取cpuRequest的值 |
| +getID():String  -ID:String | 获取ID的值 |
| +getImagePolicy():String  -imagePolicy:String | 获取imagePolicy的值 |
| +getMemLimit():String  -memLimit:String | 获取portPotocol的值 |
| +getMemRequest():String  -memRequest:String | 获取memRequest的值 |
| +getReadOnly():String  -readOnly:String | 获取readOnly的值 |
| +getShareVolumeName():String  -shareVolumeName:String | 获取shareVolumeName的值 |
| +getVolumeContainerPath():String  -volumeContainerPath:String | 获取volumeContainerPath的值 |
| +setContainerImage():void  -containerImage:String | 修改containerImage的值 |
| +setContainerName():void  -containerName:String | 修改containerName的值 |
| +setCpuLimit():void  -cpuLimit:String | 修改cpuLimit的值 |
| +setCpuRequest():void  -cpuRequest:String | 修改cpuRequest的值 |
| +setID():void  -ID:String | 修改ID的值 |
| +setImagePolicy():void  -imagePolicy:String | 修改imagePolicy的值 |
| +setMemLimit():void  -memLimit:String | 修改memLimit的值 |
| +setMemRequest():void  -memRequest:String | 修改memRequest的值 |
| +setReadOnly():void  -readOnly:String | 修改readOnly的值 |
| +setShareVolumeName():void  -shareVolumeName:String | 修改shareVolumeName的值 |
| +setVolumeContainerPath():void  -volumeContainerPath:String | 修改volumeContainerPath的值 |

**存储层**

|  |  |  |
| --- | --- | --- |
| Class Name | 类成员 | 说明 |
| ServiceOverviewDao.java | +sqlSession：Sqlsession | 用于执行数据库sql命令的对象 |
| +findAll():List<Map<String,Object>>  -params:String  -list:List<Map<String,Object>> | 查找应用所有的应用信息 |
| StatusMonitorDao.java | +sqlSession：Sqlsession | 用于执行数据库sql命令的对象 |
| +findOne():ServiceOverview  -appName:String  -sqlSession.selectOne(String,String):int | 查找一条应用的信息 |
| +findAll():List<Map<String,Object>>  -params:String  -list:List<Map<String,Object>> | 返回所有应用 |
| +findDeployment():List<Map<String,Object>>  -appID:String  -list:List<Map<String,Object>> | 查找所有部署 |
| +findService():List<Map<String,Object>>  -appID:String  -list:List<Map<String,Object>> | 查找所有服务 |

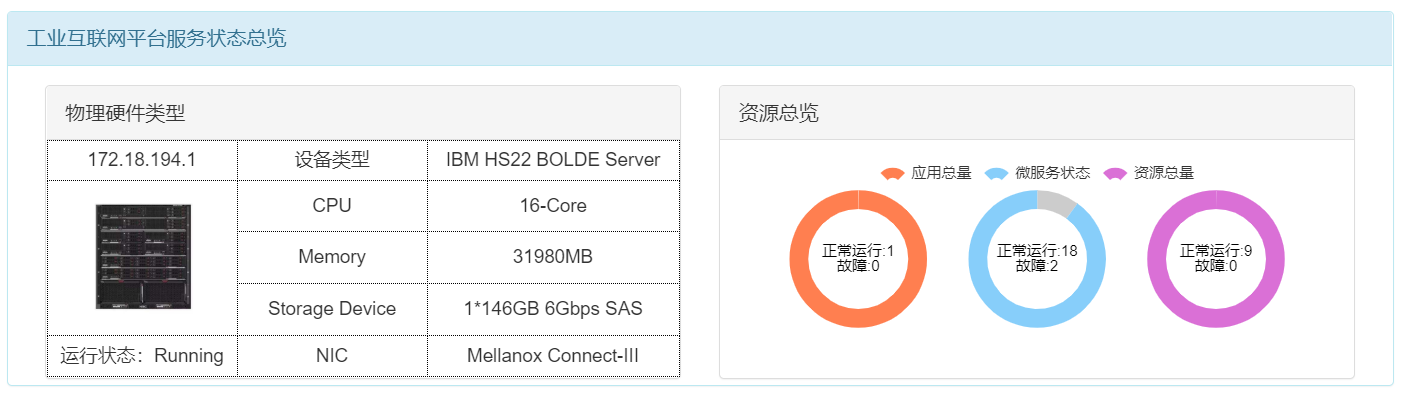
**服务层**

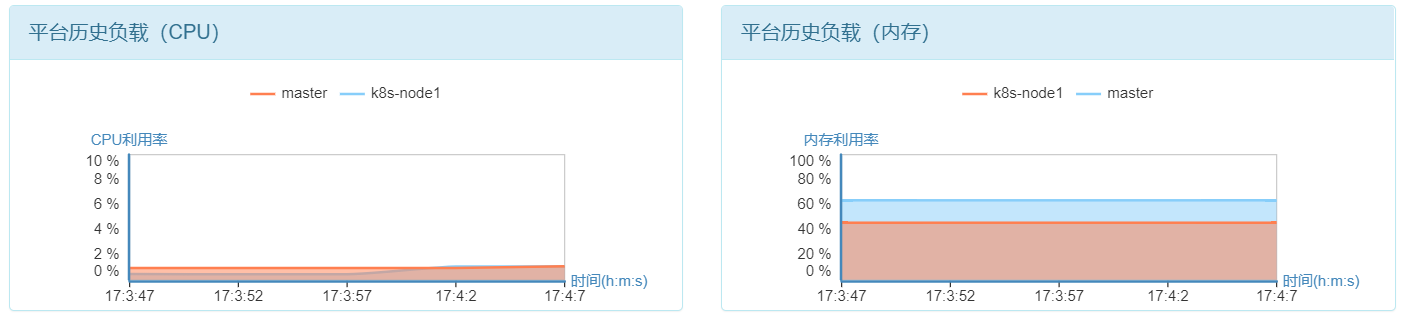
|  |  |  |
| --- | --- | --- |
| Class Name | 类成员 | 说明 |
| ServiceOverviewService.java | +sdf:SimpleDateFormat | 简单日期对象 |
| +httpClientTool:HttpClientTool | http通信对象 |
| +serviceOverviewDao:ServiceOverviewDao | 操作数据库的Dao层对象 |
| +StatusMonitorService:StatusMonitorService | 应用负载监控的Service层对象 |
| +findAll():List<Map<String,Object>>  -params:String  -list:List<Map<String,Object>> | 调用Dao层findAll函数查找所有应用信息 |
| +getClusterNode():List<Map<String,Object>>  -dataList:List<Map<String, Object>> | 获取所有节点信息 |
| +getCPULoad():List<Map<String, Object>>  -dataList:List<Map<String, Object>> | 获取CPU负载 |
| +getOneCPULoad():List<Map<String, Object>>  -dataList:List<Map<String, Object>> | 获取各个业务CPU负载情况 |
| +getPie():List<String>  -dataList:List<String> | 获取资源总览部分饼状图数据信息 |
| +getPods():List<Map<String, Object>>  -dataList:List<Map<String, Object>> | 获取容器（微服务）信息 |
| +getPodsCPU():Map<String,List<Map<String, Object>>>  -datalist:Map<String,List<Map<String, Object>>> | 获取高负载服务列表 |
| StatusMonitorService.java | +sdf:SimpleDateFormat | 简单日期对象 |
| +httpClientTool:HttpClientTool | http通信对象 |
| +statusMonitorDao:StatusMonitorDao | 操作数据库的Dao层对象 |
| +findAll():List<Map<String,Object>>  -appID:String  -list:List<Map<String,Object>> | 根据appID查找所有相关应用信息 |
| findDeployment():List<Map<String,Object>>  -appID:String  -statusMonitorDao.findDeployment(appId):List<Map<String,Object>> | 根据appID查找所有相关部署 |
| findOne():List<Map<String,Object>>  -appName:String  -statusMonitorDao.findOne(appName):List<Map<String,Object>> | 根据appName查找对应应用信息 |
| findService():List<Map<String,Object>>  -appID:String  -statusMonitorDao.findService(appId):List<Map<String,Object>> | 根据appId查询对应服务信息 |
| +getDeployMent():List<Map<String,Object>>  -dataList:List<Map<String, Object>> | 获取部署信息 |
| +getCPULoad():List<Map<String, Object>>  -dataList:List<Map<String, Object>> | 获取CPU负载信息 |
| +getOneCPULoad():List<Map<String, Object>>  -dataList:List<Map<String, Object>> | 获取获取各个业务CPU负载情况 |
| +getLog():List<String>  -dataList:List<String> | 获取日志记录 |
| +getPods():List<Map<String, Object>>  -dataList:List<Map<String, Object>> | 获取容器（微服务）信息 |
| +getPodsStatus():Map<String,List<Map<String, Object>>>  -datalist:Map<String,List<Map<String, Object>>> | 获取微服务状态 |
| +getSerDeploy():List<Map<String, Object>>  -datalist:List<Map<String, Object>> | 获取服务资源弹性伸缩状态 |
| +getService():List<Map<String, Object>>  -datalist:List<Map<String, Object>> | 获取服务状态 |

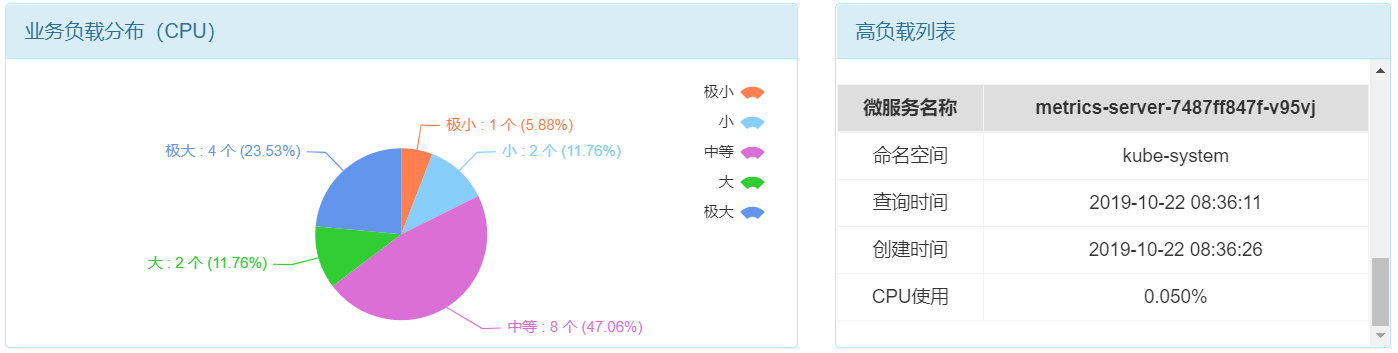
**控制层**

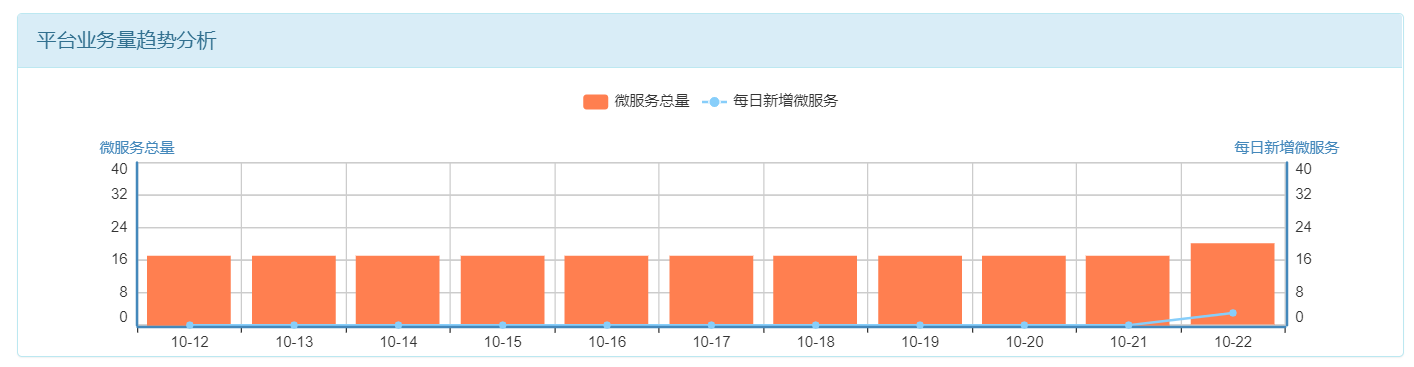
|  |  |  |
| --- | --- | --- |
| Class Name | 类成员 | 说明 |
| ServiceOverviewController.java | +serviceOverviewService:ServiceOverviewService | 服务总览的Service层对象 |
| +statusMonitorService:StatusMonitorService | 应用负载监控的Service层对象 |
| +getCPULoad():String  -ajaxObject.setBackMsg(JSON.toJSONString(data)):String | 传输CPU负载信息 |
| +getOneCPULoad():String  -ajaxObject.setBackMsg(JSON.toJSONString(data)):String | 传输各个业务CPU负载情况 |
| +getPod():String  -ajaxObject.setBackMsg(JSON.toJSONString(data)):String | 传输容器（微服务）信息 |
| +getPodsCPUData():String  -ajaxObject.setBackMsg(JSON.toJSONString(data)):String | 传输高负载服务列表 |
| +getServicePie3():String  -ajaxObject.setBackMsg(JSON.toJSONString(data)):String | 传输资源总览部分饼状图数据信息 |
| StatusMonitorController.java | +serviceOverviewService:ServiceOverviewService | 服务总览的Service层对象 |
| +statusMonitorService:StatusMonitorService | 应用负载监控的Service层对象 |
| +applicationID:String | 应用ID号 |
| +getCPULoad():String  -ajaxObject.setBackMsg(JSON.toJSONString(data)):String | 传输CPU负载信息 |
| +getLogData():String  -ajaxObject.setBackMsg(JSON.toJSONString(data)):String | 传输日志信息 |
| +getOneCPULoad():String  -ajaxObject.setBackMsg(JSON.toJSONString(data)):String | 传输各个业务CPU负载情况 |
| +getPod():String  -ajaxObject.setBackMsg(JSON.toJSONString(data)):String | 传输容器（微服务）信息 |
| +getPodsStatusData():String  -ajaxObject.setBackMsg(JSON.toJSONString(data)):String | 传输微服务状态 |

#### 4.2.2组件说明







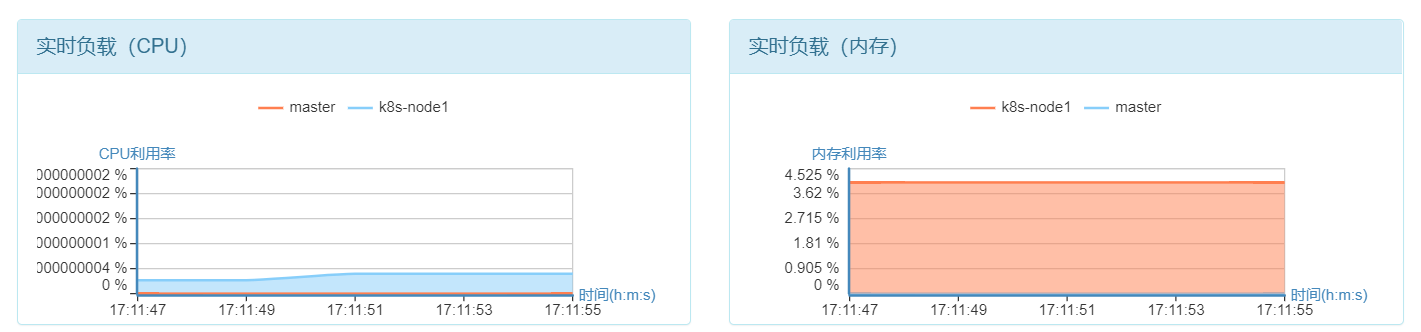


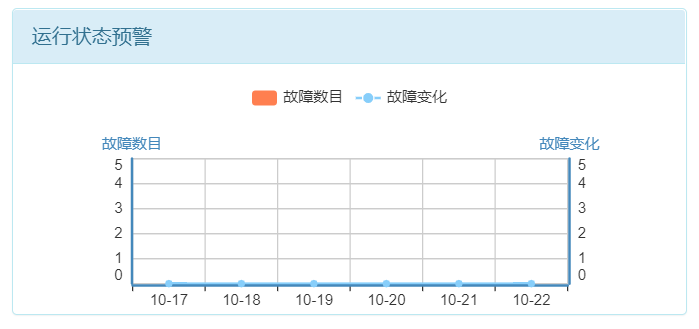
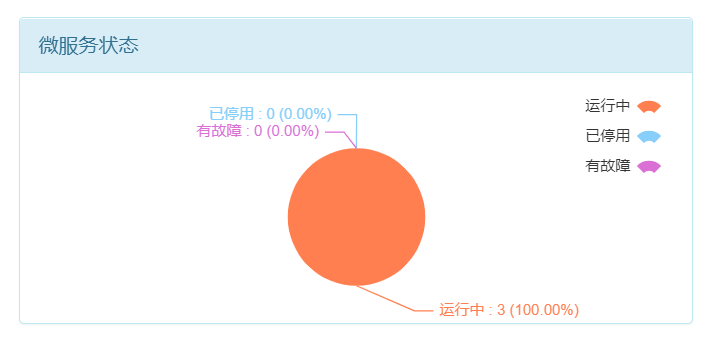


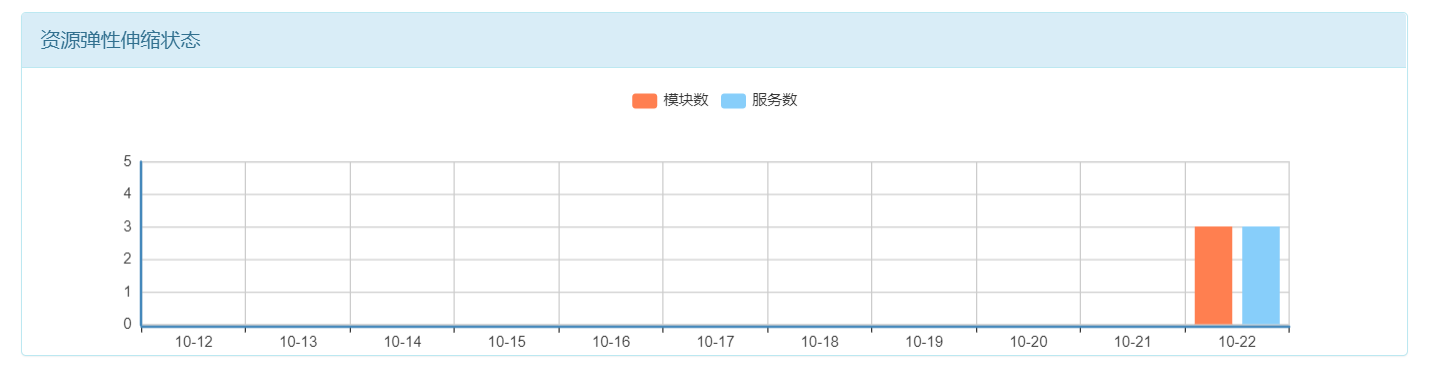


1. serviceOverview/main.jsp 为服务总览页面，提供了服务器硬件、CPU、内存资源、微服务、应用等信息的查阅功能，同时提供了平台应用的详细信息的跳转接口。

|  |
| --- |
| 画面说明 |
| 用于显示服平台的总览信息 |
| 功能说明 |
| 【工业互联网平台服务状态总览】  显示所用物理设备信息以及平台所有资源状态，包括应用总量、微服务状态和资源总量。 |
| 【平台历史负载】  动态显示平台各个节点的CPU以及内存使用情况。 |
| 【业务负载分布及高负载列表】  显示平台各业务所占CPU的使用情况，同时描述CPU使用率较高的业务的详细信息，包括命名空间、创建时间、查询时间、CPU使用率等。 |
| 【平台业务量趋势分析】  显示平台历史近十天内微服务数目及变化情况。 |
| 【平台资源总览】  显示平台各个物理节点的相关信息，包括CPU资源总数（核）、内存资源总数（字节）、内核版本、创建时间、操作系统信息、内部IP地址、镜像数目等。 |
| 【平台应用总览】   1. 显示平台目前运行的应用相关信息，包括应用名称、微服务名、微服务类型、工作节点、创建时间、容器名、容器镜像、服务端口、服务状态等。 2. 点击各个应用的名称，可进入应用负载监控界面。 |















2.statusMonitor/main.jsp 为应用负载监控页面，提供了服务实时状态显示，包括CPU负载、内存负载、微服务状态、运行状态、资源弹性伸缩状态、容器模板、应用模块信息、微服务信息、服务状态等。

|  |
| --- |
| 画面说明 |
| 用于显示应用状态的相关信息 |
| 功能说明 |
| 【实时负载】  动态显示应用的当前实时负载情况，包括CPU和内存利用率。 |
| 【微服务状态】  显示应用所包含的微服务状态。 |
| 【运行状态预警】  显示应用运行过程中出现的故障数目以及变化情况。 |
| 【资源弹性伸缩状态】  显示应用所包含的模块以及服务数目近十天内的变化情况。 |
| 【容器模板】  显示应用所使用的容器模板信息，包括CPU资源限额（核）、内存资源限额（MB）、共享卷、共享卷容器路径、是否只读、镜像拉取策略等。 |
| 【应用模块信息】  显示应用所包含的应用模块信息，包括模块名称、命名空间、模块标识、运行状态、容器数目、创建时间、组策略等。 |
| 【微服务信息】  显示应用所包含的微服务信息，包括微服务名、微服务类型、工作节点、创建时间、容器、容器镜像、服务端口、工作状态等。 |
| 【服务状态】  显示应用所包含的服务状态信息，包括名称、命名空间、识别号、内部端口、外部端口、创建时间等。 |

#### 4.2.3数据库说明

|  |  |
| --- | --- |
| Table ID | Table Name |
| CONTAINER | 容器主表 |
| CONTAINERPORT | 容器端口表 |
| CONTAINERENV | 容器环境变量表 |
| POD | 容器组（微服务）表 |
| DEPLOYMENT | 应用模块表 |
| SERVICE | 网络组件表 |
| SERVICEPORT | 网络端口表 |
| APPLICATION | 应用表 |

#### 4.2.4与其他接口说明

|  |  |
| --- | --- |
| Interface | Description |
| freemarker.template.Configuration | 通过模板自动生成yaml文件 |
| freemarker.template.Template |
| freemarker.template.TemplateException |
| org.apache.commons.httpclient.methods.PostMethod | 通过url发送yaml文件部署组件 |
| org.apache.commons.httpclient.methods.DeleteMethod | 通过url发送yaml文件停止组件 |
| "http://172.18.194.2:8087/apis/apps/v1/namespaces/default/deployments" | deployment操作url |
| "http://172.18.194.2:8087/api/v1/namespaces/default/services" | service操作url |