DigitalTwin详细设计

文档修改记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本号** | **修订日期** | **修改内容** | **作者** | **审批人** |
| v1.0 | 2018-06-01 | 创建 | 陈昌辉 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 概述

## 背景

//TODO

## 目的

//TODO

## 范围

//TODO

## 术语

|  |  |  |
| --- | --- | --- |
| **序号** | **术语名称** | **术语定义** |
| 1 | DT | DigitalTwin（数字镜像）简写 |

# 需求描述及分析



## 需求描述

//TODO

## 需求分析

//TODO

# 概要设计



## 设计目标

DT是一个基于HTTP协议的无状态接口，面向Leap IoT的开发者。主要实现3部分功能：

1. 提供前端WEB界面，管理DT模板和DT实例对象；
2. 为Leap Stream Stage开发者提供DT模板访问API，通过该API可以在Leap Steam Stage中明确知道存储DT实例数据的格式；
3. 为上层外部应用提供RESTful访问接口，实现对DT实例的访问，监控以及反控。

## 架构设计

DT为无状态服务，集群化部署时支持水平扩容。

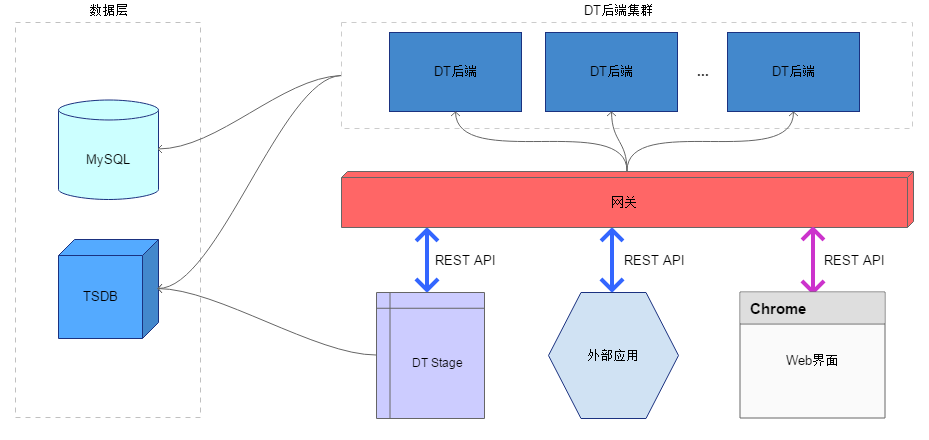


图3-1 DT架构设计

# 详细设计

## 4.1 模块设计

整个DT组件包括3个模块：前端，后端，以及运行与StreamSet中的Dt Stage。

DT前端和DT Stage都需要通过DT后端进行交互。

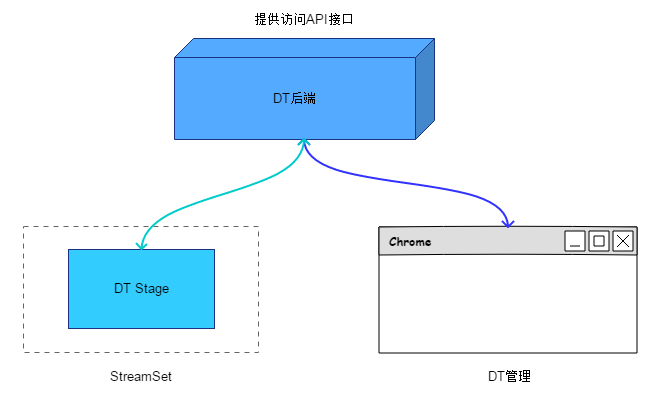


图4-1 DT模块设计

### 前端

前端是一个WEB界面，作为DT模板和DT实例的管理接口，面对的是DT管理者。

通过DT前端实现对DT模板和实例的管理功能。

### 后端

提供交互API，分为3类：前端API，Leap Stream访问API，上层应用API。 不同的API需要进行对应的权限管理和控制。

DT后端采用如下技术架构：Spring Boot + Spring Security + JWT。

### DT Stage

DT Stage运行在Stream Set中负责数据清洗工作，与DT后端进行交互。

## 功能设计

DT组件的功能包括：管理DT模板，管理DT实例，提供访问API。

### 管理的DT模板

管理DT模板是指通过前端WEB接口管理DT数据模板，包括对DT模板的创建，删除，修改等。DT模板在Leap Stream中使用，Leap Stream开发者通过相应的DT接口可以查询到当前定义的所有DT模板，并使用对应DT模板定义的数据格式将数据存储到TSDB中。

### 管理DT实例

DT必须继承一个DT模板，作为DT模板的实例化对象。

管理DT实例包括对DT实例的添加，删除，修改等，在DT前端可以通过Dashbord方式展示DT实例列表和统计，另外还支持通过DT实例对实际设备进行反向控制。

### 访问API

DT提供的访问API分为三部分：前端API，上层应用API，Leap Stream API。

前端API用于实现Web界面交互，具体包括DT模板管理，DT实例管理。

上层应用API是指以RESTful接口方式为上层应用提供访问和管理DT实例的入口。

Leap Stream API主要是给Leap Stream开发者提供访问DT模板的入口，Leap Stream开发者通过该接口访问到当前定义的所有DT模板，实现对存储DT数据的格式控制。

### 数据清洗

数据清洗的功能通过DT Stage这个模块实现。

## 接口设计

接口部分包括三部分：前端API，Leap Stream API，为上层应用提供的外部API。

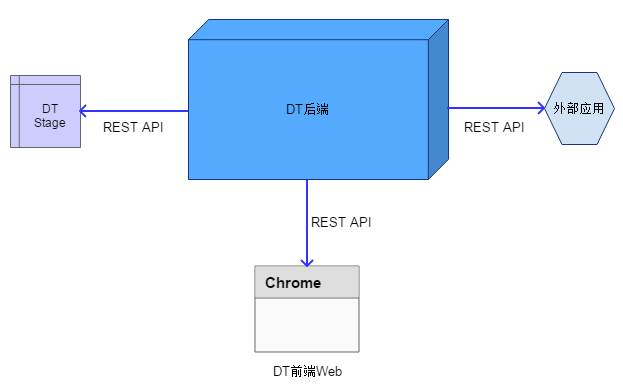


图4-2 DT接口设计

### 前端API

前端API为DT前端界面管理DT模板和实例的访问接口，基于HTTP协议实现。由于HTTP协议本身的无状态性，为了DT在部署时水平扩容更加灵活和方便，在前端API设计时采用JWT进行身份认证。

#### DT模板管理

1. 查询DT模板列表
2. 查询DT模板详情
3. 添加DT模板
4. 更新DT模板
5. 删除DT模板

#### DT实例管理

1. 查询DT实例列表
2. 查询DT实例详情
3. 添加DT实例
4. 编辑DT实例
5. 删除DT实例

### Leap Stream Stage API

Leap Stream Stage API是为Leap Stream Stage开发者提供访问DT模板的接口，

Leap StreamStage开发者通过该接口获取到DT模板定义，并根据该模板控制存储到TSDB的

数据格式。

### 外部API

外部API是为上层应用提供的访问以及操作DT实例的接口，以RESTFul形式提供，该接口主要是为了那些基于DT进行应用开发的场景准备的。

## 存储设计

### 整体设计

在DT的架构中，整个存储设计包含2部分。其一，对于DT实例数据存储格式的元数据定义，这部分数据存储在MySQL中；其二，DT实例的数据本身存放到TSDB中。

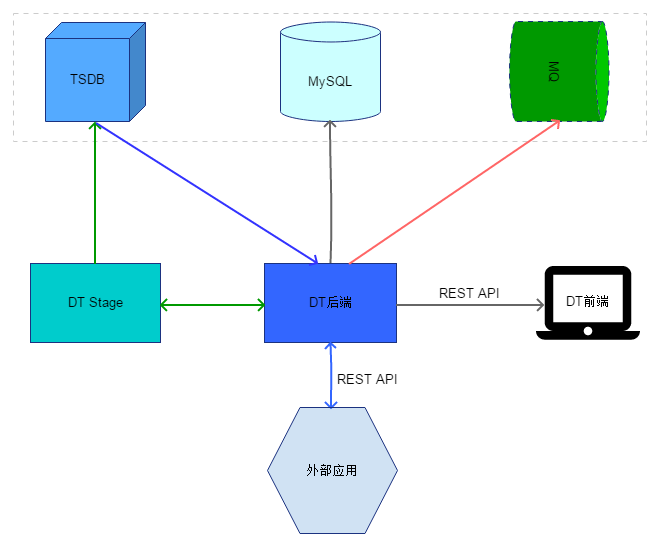


图 4.1 DT存储设计

### 数据结构

#### MySQL表结构

在MySQL中存储的是DT实例数据的存储格式，这部分数据是固定的。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 数据类型 | 默认值 | 注释 |
| id | BIGINT | 0 | 模板ID |
| name | VARCHAR | “” | 模板名称 |
| desp | VARCHAR | “” | 模板描述 |
| label | VARCHAR | “” | 模板标签 |
| cime | TIMESTAMP | now() | 创建时间戳 |
| mtime | TIMESTAMP | now() | 更新时间戳 |

表4.1 DT模板表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 数据类型 | 默认值 | 注释 |
| id | BIGINT | 0 | 属性ID |
| tpl | BIGINT | 0 | 模板ID |
| name | VARCHAR | “” | 属性名称 |
| desp | VARCHAR | “” | 属性描述 |
| datatype | VARCHAR | “” | 数据类型 |
| value | VARCHAR | “” | 默认值 |
| cime | TIMESTAMP | now() | 创建时间戳 |
| mtime | TIMESTAMP | now() | 更新时间戳 |

表 4.2 DT模板属性表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 数据类型 | 默认值 | 注释 |
| id | BIGINT | 0 | 实例ID |
| name | VARCHAR | 0 | 实例名称 |
| desp | VARCHAR | “” | 实例描述 |
| label | VARCHAR | “” | 实例标签 |
| tplid | BIGINT | 0 | 所继承模板ID |
| tplname | VARCHAR | “” | 所继承模板名称 |
| ctime | TIMESTAMP | now() | 创建时间戳 |
| mtime | TIMESTAMP | now() | 更新时间戳 |

表4.3 DT实例表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 数据类型 | 默认值 | 注释 |
| id | BIGINT | 0 | 属性ID |
| itcid | BIGINT | 0 | DT实例ID |
| name | VARCHAR | “” | 属性名 |
| desp | VARCHAR | “” | 属性描述 |
| value | VARCHAR | “” | 属性值 |
| datatype | VARCHAR | “” | 数据类型 |
| metric | VARCHAR | “” | 指标 |
| tags | VARCHAR | “” | tags列表 |
| ctime | TIMESTAMP | now() | 创建时间戳 |
| mtime | TIMESTAMP | now() | 更新时间戳 |

表 4.4 DT实例属性表

#### TSDB存储结构

在TSDB中存储的是DT实际数据，存储的实际格式根据对应的DT模板而定。不同的DT模板对应着不同的数据存储格式，而且这个格式可能是动态变化的。

## 部署设计

//TODO

# 附录

无。