文件编号：

项目编号：

密级：

服务框架设计交流

与

Demo演示

# 系统概述

## 设计目标

1. 打造一个松耦合、基于服务的架构，简化服务的发布与调用
2. 构建一个统一的模型，统一系统间的操作，从而能把精力集中于服务的接口与业务实现
3. 为服务提供水平伸缩能力

## 系统分层



分布式服务框架对中间件不做过多的假设与依赖，应用服务可以比较自由的选择部署服务的容器、中间件或框架

## 系统软件栈

#### 在Java一统天下的产品线中很可能出现的一种情况



#### 跨平台需求下的选择

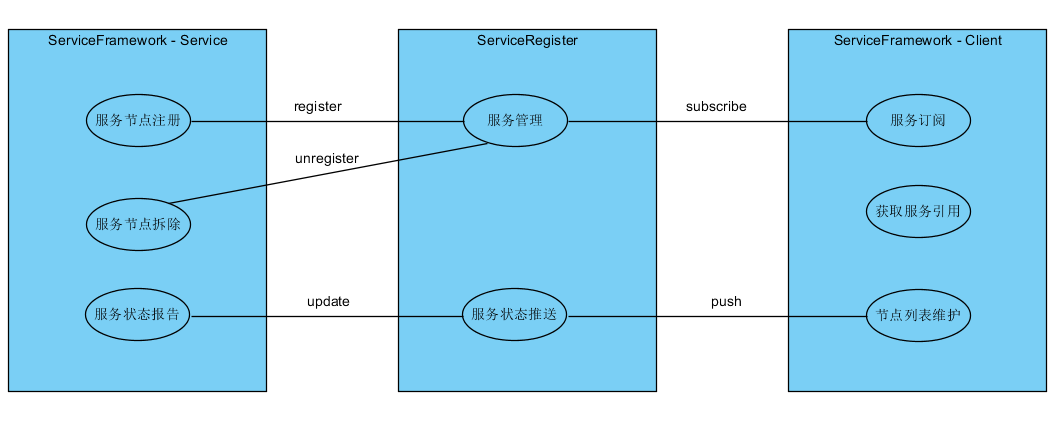


Thrift带来跨平台好处时同时也带来了“强入侵”，开发人员需要继承Thrift生成的接口类，并使用其提供的各种Server类来实现应用服务，是否能将一个POJO类直接部署为一个Thrift服务呢？这个也是DSF今后演进的一个方向。

# 设计与实现

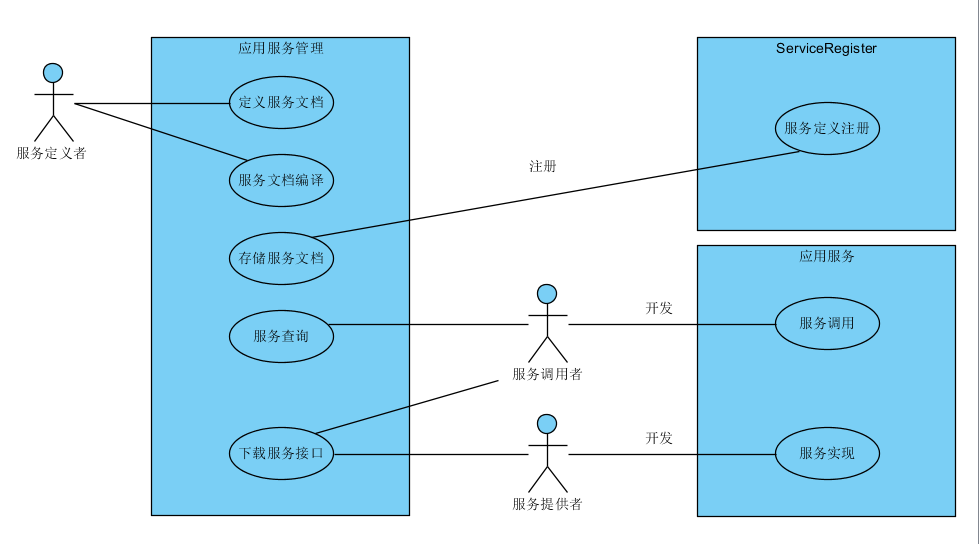
## Demo实现的用例

服务注册与订阅



## Demo未实现的用例

服务定义安装与查询



## 如何订阅并调用一个服务

对于调用者，仅需要知道服务的接口与名字

private String service1 = "tutorial.Calculator v1.0";

public void init() {

adaptor.subscribeService(service1); //订阅

}

public void call() {

Calculator.Iface client = adaptor.getClient(service1); //获取引用

client.add(5, 10); //调用服务

}

对于Spring框架，可以在xml配置文件中定义需要订阅的服务，并直接注入需要调用服务的Bean对象中

## Demo演示

Demo已实现的特性：位置无关性、对于应用服务实现者来说，无需做任何额外的开发即可实现服务集群

## 分布式服务框架设计原则

1. 框架不应挤占应用服务资源
2. 简单的通讯机制与协议，易实现易移植
3. 要能同时支持同步调用与消息推送
4. 简单轻量无依赖，不应限制应用服务平台的选择
5. 高可靠可恢复

## 注册服务器

注册服务器负责存储并管理服务信息，包括接口文档、节点信息等。

#### 开发平台

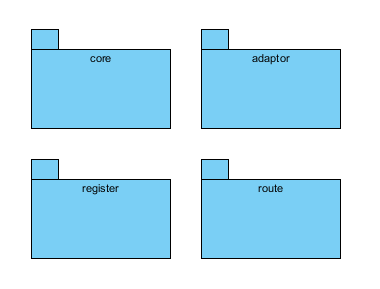
注册服务器功能单纯简单，要求稳定、高效、灵活、可定制扩展，被系统中绝大部分服务引用，因此将被设计成一个自持的轻量级服务器，不依赖于其他服务器与中间件。

Demo基于在高可靠性上有着优秀表现的Erlang平台开发。

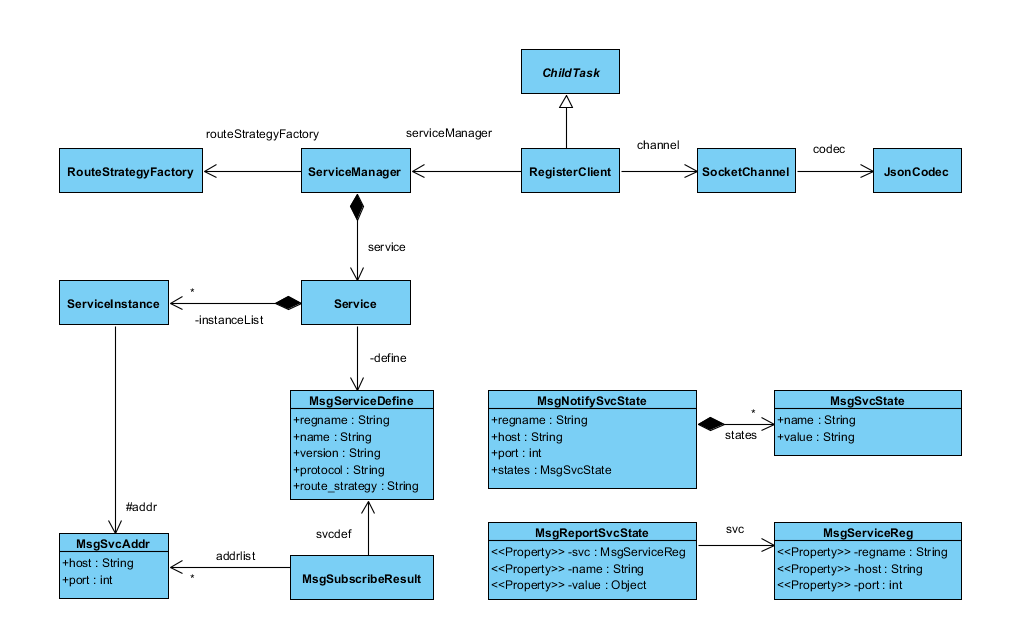
#### 服务器热备

注册服务器应具备热备或集群能力以避免产生单点故障

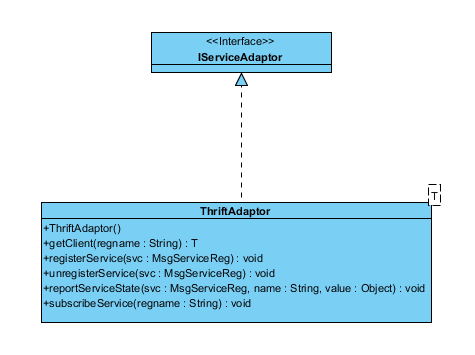
## 框架命名空间



#### register命名空间：与注册服务器通信，服务队列管理

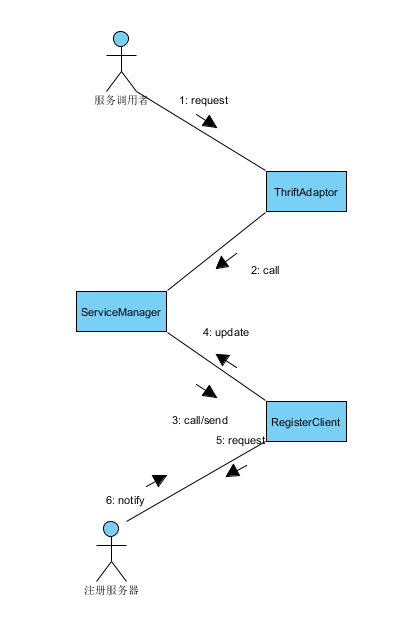


#### adaptor命名空间：服务注册接口，创建Thrift客户端或其装饰类，连接池管理



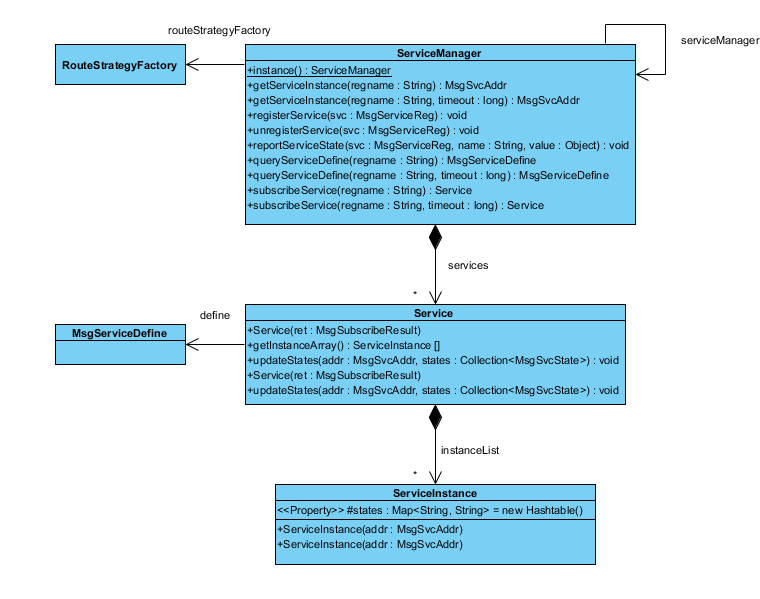
### 主要业务对象

#### 通讯概览



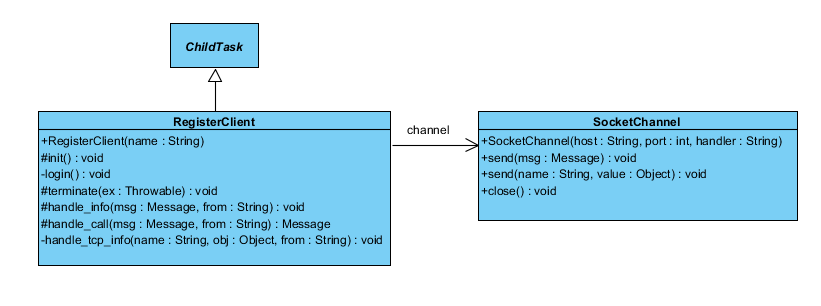
#### ServiceManager

管理服务节点列表，存储更新节点状态，并提供相应的查询接口

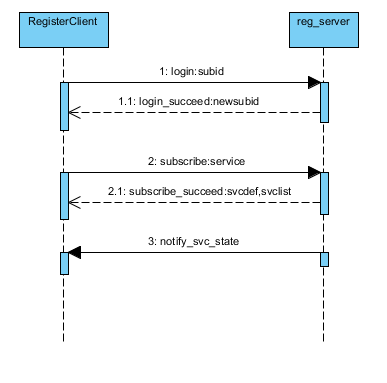


#### RegisterClient

负责与注册服务器通讯，接收到推送的服务状态则调用ServiceManager的方法进行更新



#### RegisterClient登陆与服务订阅序列图



#### 订阅号

##### 目的

1. 标识客户端
2. 避免每次客户端离线都需要重发所有订阅内容。服务端不了解客户端离线的原因，如果客户端是因为退出离线则需要更新所有订阅的内容，如果客户端是掉线则无需重发订阅内容

##### 订阅号生成

服务端生成一个唯一序列号

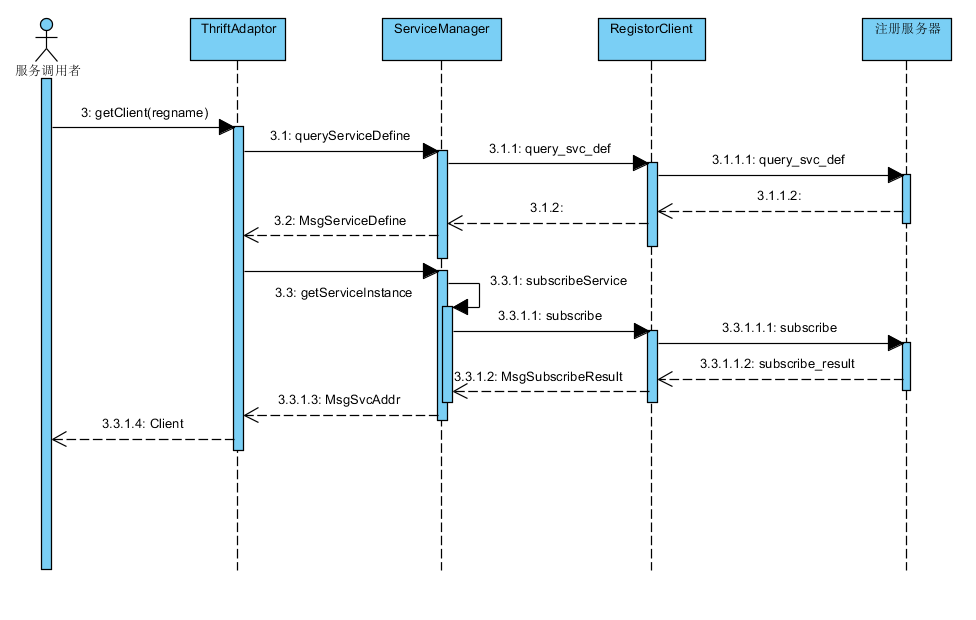
##### 生命周期

客户端：进程

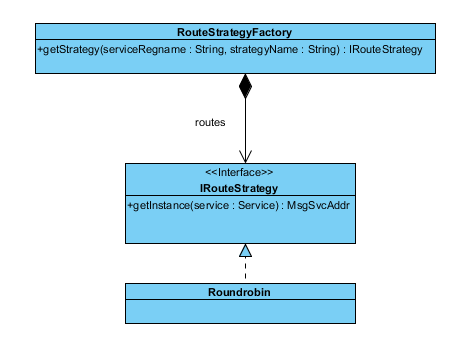
服务端：离线超时

离线超时时间不宜设置过长，通常设置为数分钟，以免服务端维护过多无效的信息， 比如因离线而保存在队列中等待推送的消息

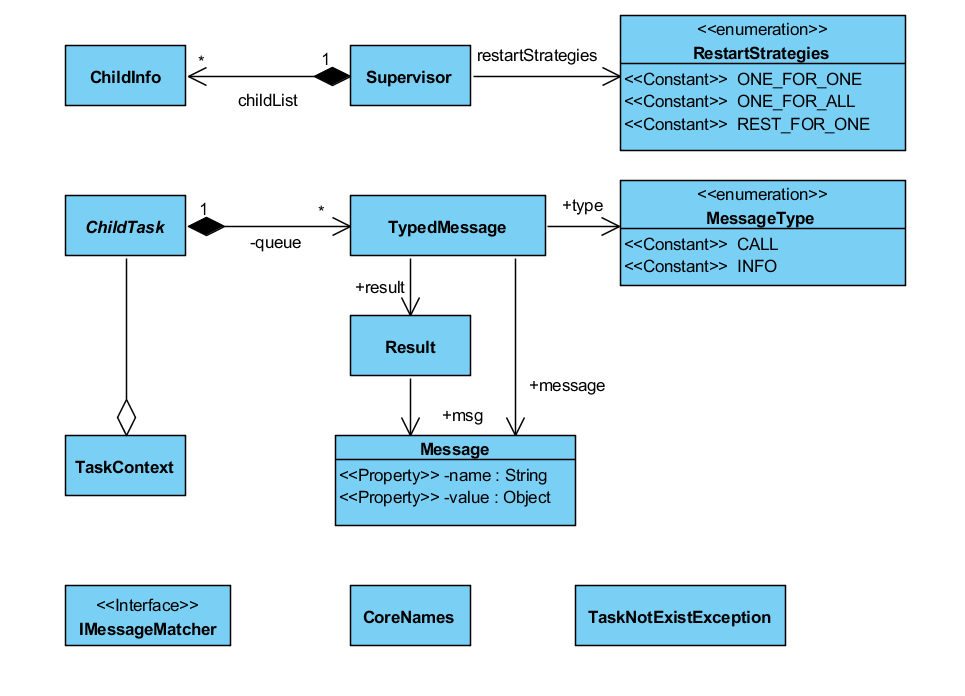
#### 客户端获取服务接口序列图



#### route命名空间：路由策略

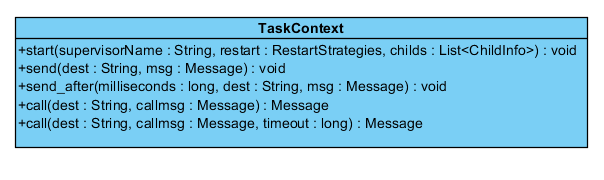


#### core命名空间：线程间异步及同步通信，线程监控与重启



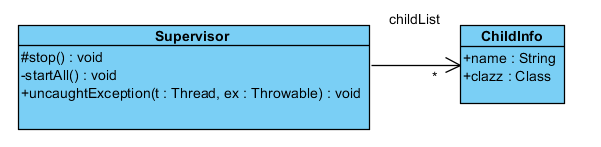
#### TaskContext

上下文环境类，ChildTask类引用TaskContext单例，向其他ChildTask发送消息、发起调用



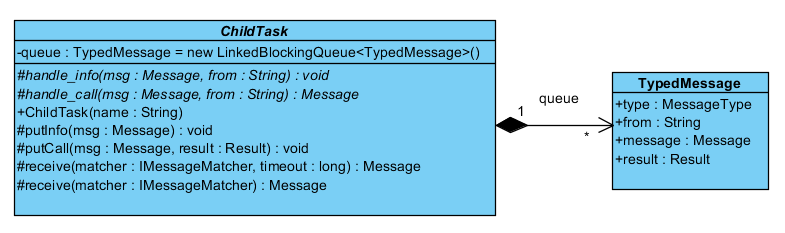
#### Supervisor

启动并监控ChildTask，根据策略重启线程



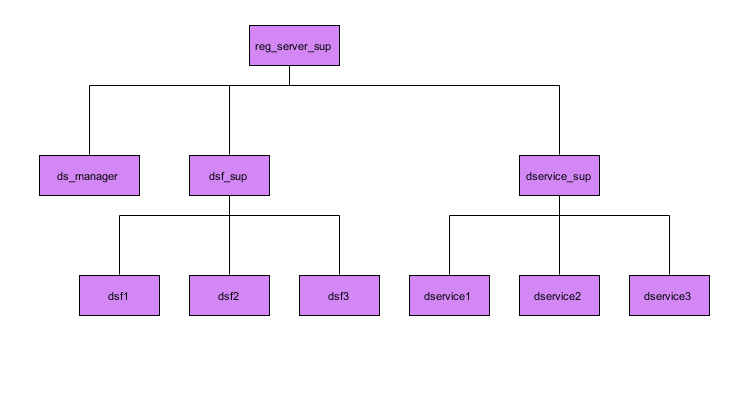
#### ChildTask

维护消息队列(MailBox)，接收并处理消息



## 注册服务器

### 监控树



### 注册服务器主要业务序列图

