

拉勾教育

— 互联网人实战大学 —

《31 讲带你搞懂 SkyWalking》

徐郡明 资深技术专家

— 拉勾教育出品 —

第21讲：Cluster 插件剖析 你想要的集群模式它都有

```
cluster:
  standalone:
    # zookeeper:
    #   nameSpace: ${SW_NAMESPACE:""}
    #   hostPort: ${SW_CLUSTER_ZK_HOST_PORT:localhost:2181}
    #   #Retry Policy
    #   baseSleepTimeMs: ${SW_CLUSTER_ZK_SLEEP_TIME:1000}
    #   maxRetries: ${SW_CLUSTER_ZK_MAX_RETRIES:3}
    # kubernetes:
    #   watchTimeoutSeconds: ${SW_CLUSTER_K8S_WATCH_TIMEOUT:60}
    #   namespace: ${SW_CLUSTER_K8S_NAMESPACE:default}
    #   labelSelector: ${SW_CLUSTER_K8S_LABEL:app=collector,release=skywalking}
    #   uidEnvName: ${SW_CLUSTER_K8S_UID:SKYWALKING_COLLECTOR_UID}
    # nacos:
    #   serviceName: ${SW_SERVICE_NAME:"SkyWalking_OAP_Cluster"}
    #   hostPort: ${SW_CLUSTER_NACOS_HOST_PORT:localhost:8848}
```

cluster-standalone-plugin 的相关配置

cluster-zookeeper-plugin 相关配置

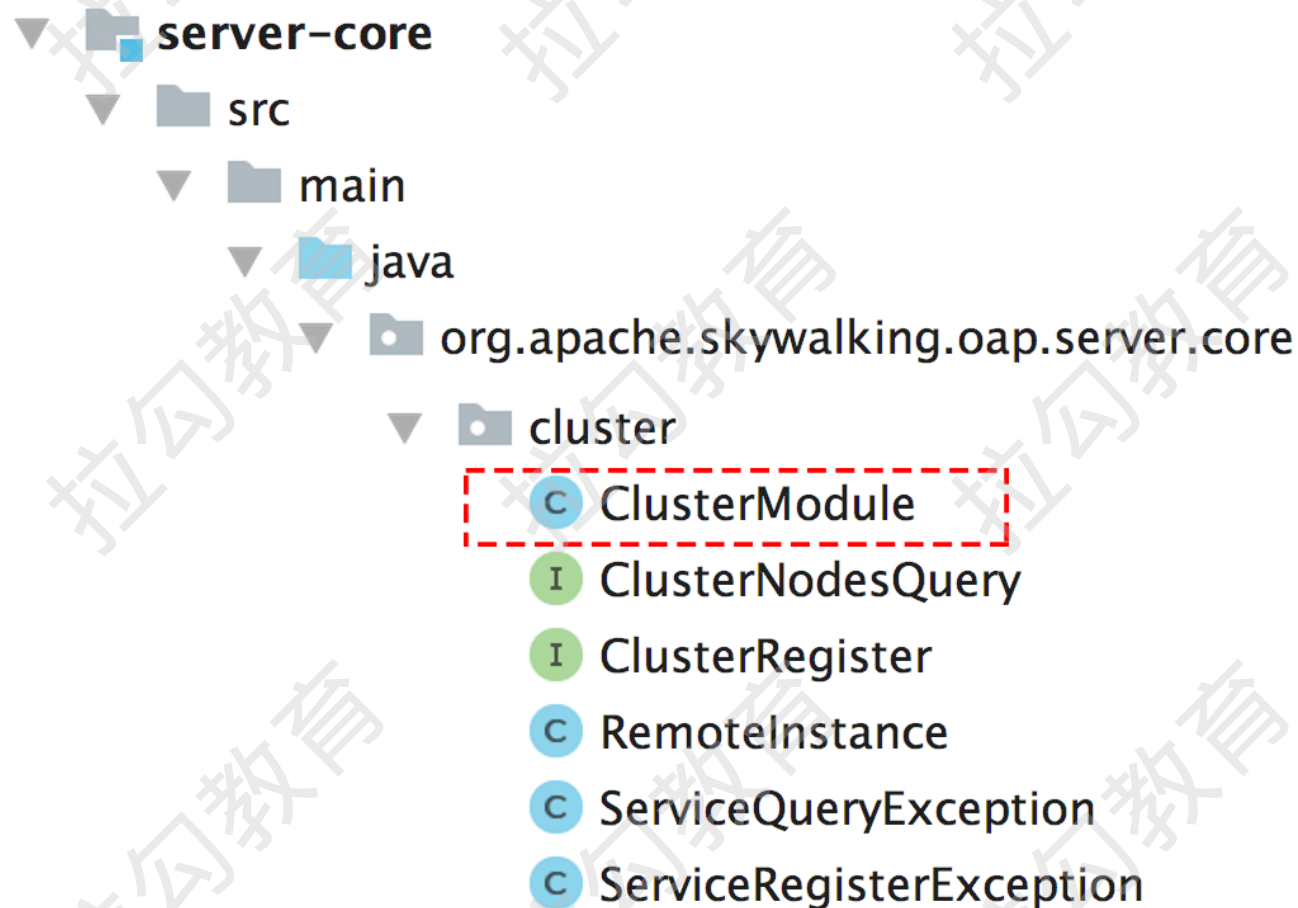
cluster-kubernetes-plugin 相关配置

cluster-nacos-plugin 相关配置

ClusterModule

拉勾教育

— 互联网人实战大学 —










ClusterRegister 接口中定义了注册集群中一个节点地址的方法

```
public interface ClusterRegister extends Service {  
    void registerRemote(RemoteInstance remoteInstance);  
}
```

在 ClusterNodesQuery 接口中定义了查询集群中所有远端节点地址的方法

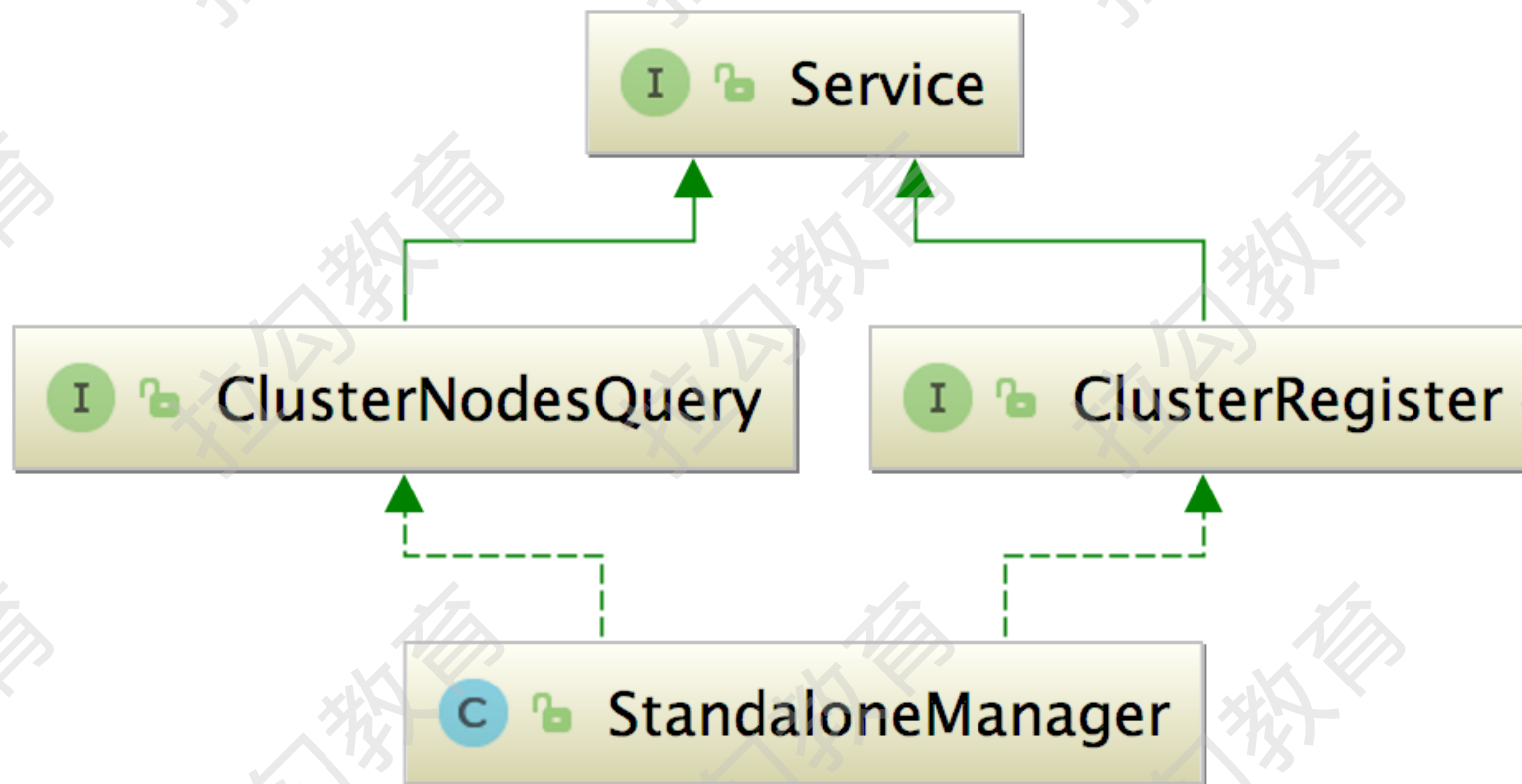
```
public interface ClusterNodesQuery extends Service {  
    List<RemoteInstance> queryRemoteNodes(); // 查询集群中全部节点  
}
```

- ▼  **server-cluster-plugin**
 - ▶  **cluster-consul-plugin**
 - ▶  **cluster-etcd-plugin**
 - ▶  **cluster-kubernetes-plugin**
 - ▶  **cluster-nacos-plugin**
 - ▶  **cluster-standalone-plugin**
 - ▶  **cluster-zookeeper-plugin**

cluster-standalone-plugin 模块

拉勾教育

— 互联网人实战大学 —



为了避免 **curator-recipes** 包过于膨胀

Curator 将很多其他解决方案都拆出来

作为单独的一个包，命名方式就是 **curator-x-***

例如：curator-x-discovery、curator-x-rpc



- **ServiceInstance**

是 curator-x-discovery 扩展包对服务实例的抽象

ServiceInstance 由 name、id、address、port 以及一个可选的 payload 属性构成

```
base path
|_____ service A name
|_____ instance 1 id --> (serialized ServiceInstance)
|_____ instance 2 id --> (serialized ServiceInstance)
|_____ ...
|_____ service B name
|_____ instance 1 id --> (serialized ServiceInstance)
|_____ instance 2 id --> (serialized ServiceInstance)
|_____ ...
|_____ ...
```

curator-x-discovery 扩展库

拉勾教育

— 互联网人实战大学 —

- **ServiceProvider**

是 curator-x-discovery 扩展包的核心，它提供了多种不同策略的服务发现方式

具体策略有：轮询调度、随机和黏性（总是选择相同的一个）

得到 ServiceProvider 对象之后，可以调用其 getInstance() 方法

按照指定策略获取 ServiceInstance 对象（即发现可用服务实例）

还可以调用 getAllInstances() 方法，获取所有 ServiceInstance 对象（即获取全部可用服务实例）

curator-x-discovery 扩展库

- **ServiceDiscovery**

是 curator-x-discovery 扩展包的入口类

开始必须调用 start() 方法，当使用完成应该调用 close() 方法进行销毁

- **ServiceCache**

如果程序中会频繁地查询 ServiceInstance 对象、添加 ServiceCache 缓存，ServiceCache 会在内存中缓存 ServiceInstance 实例的列表，并且添加相应的 Watcher 来同步更新缓存

查询 ServiceCache 的方式也是 getInstances() 方法

另外，ServiceCache 上还可以添加 Listener 来监听缓存变化

cluster-zookeeper-plugin 模块

拉勾教育

— 互联网人实战大学 —

cluster-zookeeper-plugin 模块中的 ModuleProvider SPI 文件中指定的实现类是

ClusterModuleZookeeperProvider

其对应的 ModuleConfig 实现类是ClusterModuleZookeeperConfig 类

```
private String nameSpace; //命名空间，即Zk节点的路径，默认值为"/skywalking"  
private String hostPort; // Zookeeper集群地址  
private int baseSleepTimeMs; //两次重试之间的初始间隔时间，后面间隔会指数增长  
private int maxRetries; //最大重试次数
```

cluster-zookeeper-plugin 模块

拉勾教育

— 互联网人实战大学 —

```
public void prepare(){
    RetryPolicy retryPolicy = //重试策略
        new ExponentialBackoffRetry(config.getBaseSleepTimeMs(),
            config.getMaxRetries());
    // 创建Curator客户端
    client = CuratorFrameworkFactory.newClient(config.getHostPort(),
        retryPolicy);
    // 存储ServiceInstance实例的节点路径
    String path = BASE_PATH + (StringUtil.isEmpty(
        config.getNamespace()) ? "" : "/" + config.getNamespace());
    // 创建ServiceDiscovery
    serviceDiscovery = ServiceDiscoveryBuilder.builder(
        RemoteInstance.class).client(client) // 依赖Curator客户端
```

cluster-zookeeper-plugin 模块

拉勾教育

— 互联网人实战大学 —

```
.basePath(path) // 管理的Zk路径
.watchInstances(true) // 当ServiceInstance加载
// 这里的SWInstanceSerializer是将RemoteInstance序列化成Json
.serializer(new SWInstanceSerializer()).build();
client.start(); // 启动Curator客户端
client.blockUntilConnected(); // 阻塞当前线程，等待连接成功
serviceDiscovery.start(); // 启动ServiceDiscovery
// 创建ZookeeperCoordinator对象
ZookeeperCoordinator coordinator =
    new ZookeeperCoordinator(config, serviceDiscovery);
// 注册ClusterRegister、ClusterNodesQuery实现
this.registerServiceImplementation(ClusterRegister.class,
    coordinator);
```

cluster-zookeeper-plugin 模块

拉勾教育

— 互联网人实战大学 —

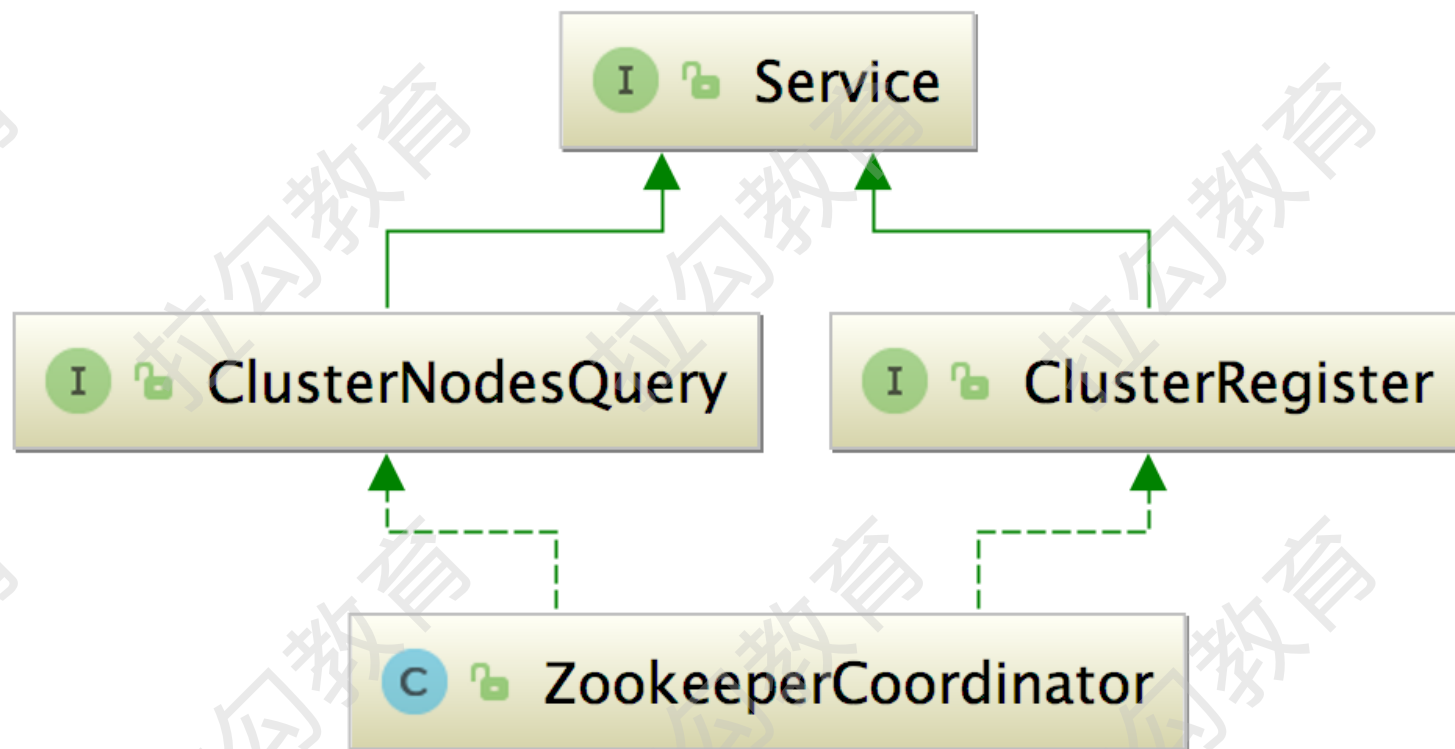
```
.serializer(new SWInstanceSerializer()).build();
client.start(); // 启动Curator客户端
client.blockUntilConnected(); //阻塞当前线程，等待连接成功
serviceDiscovery.start(); // 启动ServiceDiscovery
// 创建ZookeeperCoordinator对象
ZookeeperCoordinator coordinator =
    new ZookeeperCoordinator(config, serviceDiscovery);
// 注册ClusterRegister、ClusterNodesQuery实现
this.registerServiceImplementation(ClusterRegister.class,
    coordinator);
this.registerServiceImplementation(ClusterNodesQuery.class,
    coordinator);
}
```


cluster-zookeeper-plugin 模块

拉勾教育

— 互联网人实战大学 —

ZookeeperCoordinator 同时实现了 ClusterRegister、ClusterNodesQuery 两个接口



cluster-zookeeper-plugin 模块

拉勾教育

— 互联网人实战大学 —

```
synchronized void registerRemote(RemoteInstance remoteInstance){  
    String remoteNamePath = "remote";  
    // 将RemoteInstance对象转换成ServiceInstance对象  
    ServiceInstance<RemoteInstance> thisInstance = ServiceInstance.  
        <RemoteInstance>builder().name(remoteNamePath)  
        .id(UUID.randomUUID().toString()) // id是随机生成的UUID  
        .address(remoteInstance.getAddress().getHost())  
        .port(remoteInstance.getAddress().getPort())  
        .payload(remoteInstance).build();  
    //将ServiceInstance写入到Zookeeper中  
    serviceDiscovery.registerService(thisInstance);  
    //创建ServiceCache, 监Zookeeper相应节点的变化, 也方便后续的读取  
    serviceCache = serviceDiscovery.serviceCacheBuilder()  
        .name(remoteNamePath).build();  
    serviceCache.start(); // ??ServiceCache  
}
```

基于 ZooKeeper 的配置管理

拉勾教育

— 互联网人实战大学 —

- 介绍ClusterModule 的相关内容
- 介绍cluster-standalone-plugin 模块对单机模式的支持
- 介绍 cluster-zookeeper-plugin 模块如何依赖 Apache Curator 扩展包 curator-x-discovery 以及 ZooKeeper 集群实现 OAP 集群的功能



Next: 加餐2 《请求接待员——Server 那些事》

拉勾教育

— 互联网人实战大学 —



关注拉勾「教育公众号」
获取更多课程信息