《31 讲带你搞懂 SkyWalking》

徐郡明 资深技术专家

— 拉勾教育出品 —

加餐2：请求接待员——Server 那些事

前言

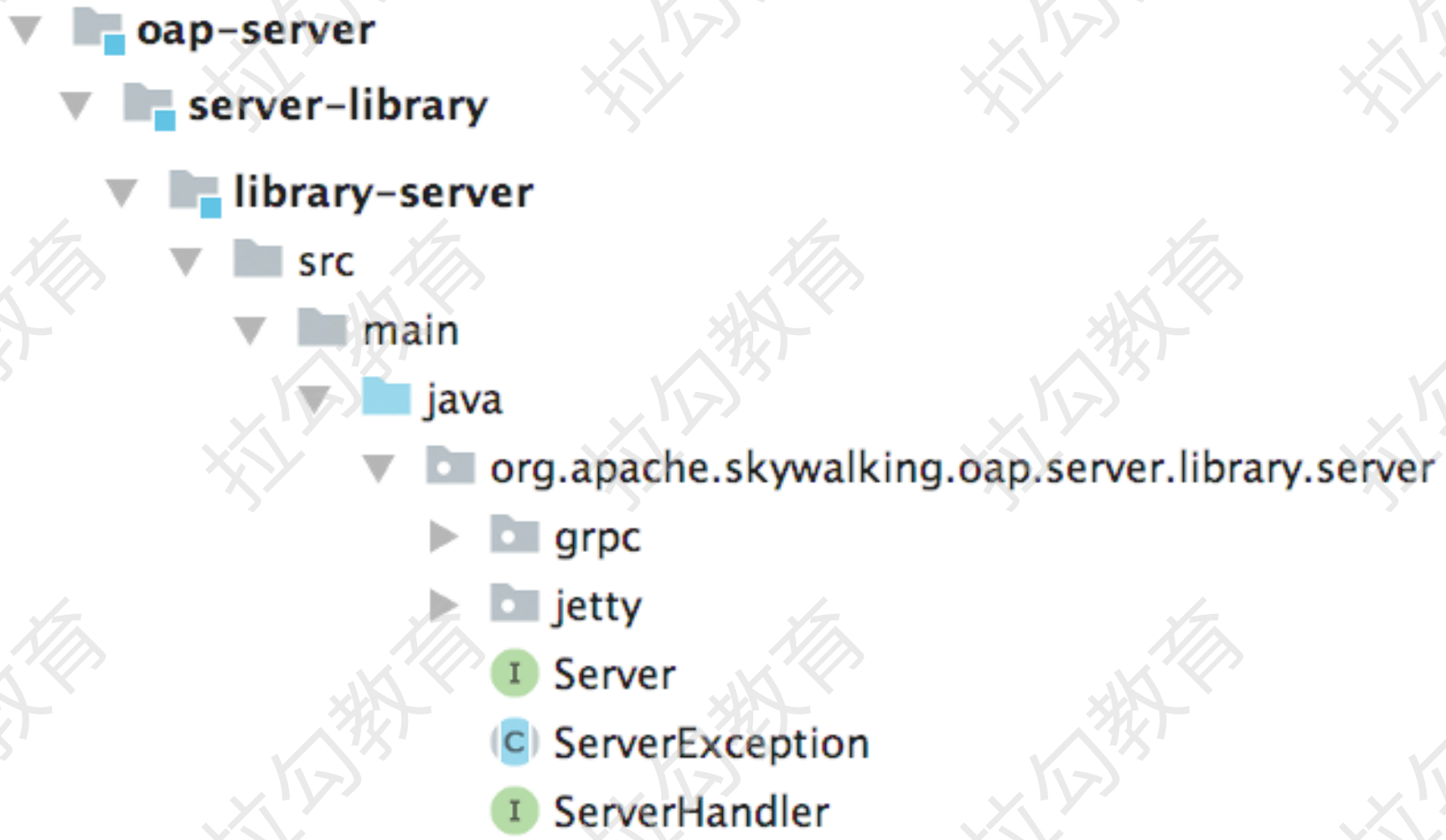
用户可以在 agent.config 文件中的 **collector.backend_service** 配置项

指定多个 OAP 服务的地址（逗号分隔）

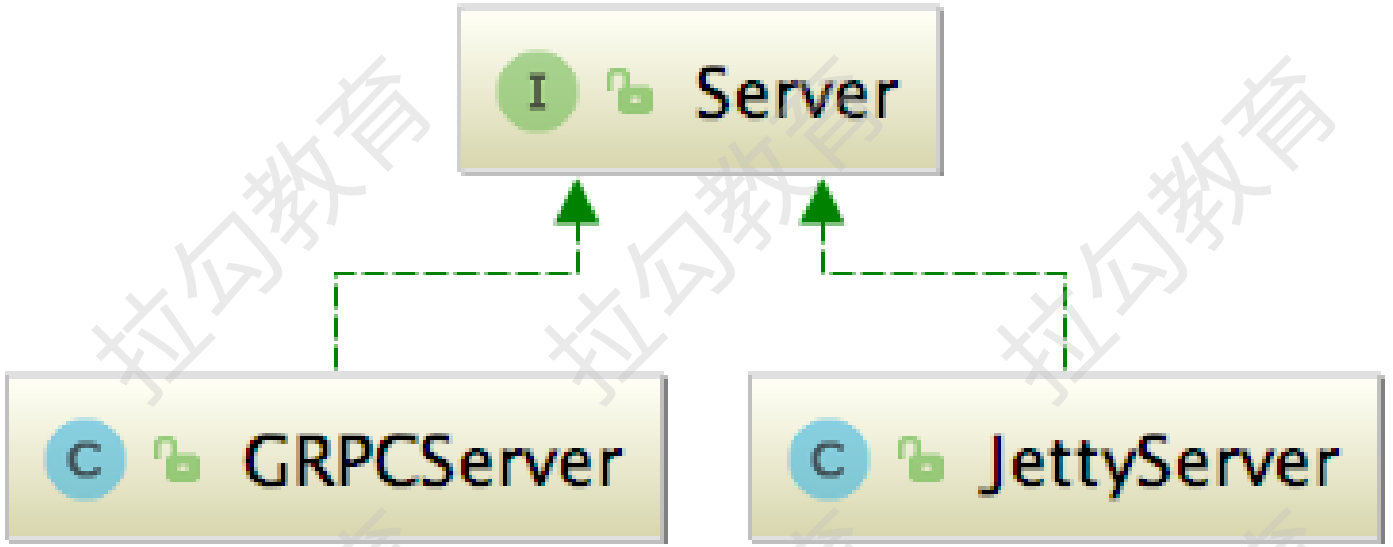
SkyWalking Agent 会切分该配置项，得到 OAP 服务列表

然后从其中随机选择一个 OAP 服务创建长连接，实现后续的数据上报

Server 核心实现



Server 核心实现

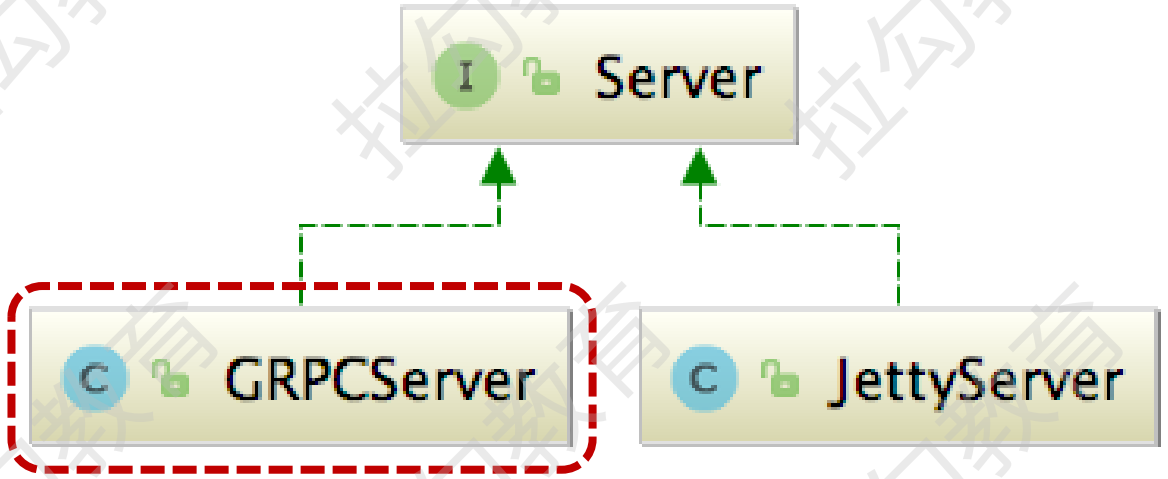


Server 核心实现

- **GRPCServer** 用于接收 SkyWalking Agent 发送的 gRPC 请求

  SkyWalking 6.x 中的 Trace 上报、JVM 监控上报、服务以及服务实例注册请求、心跳请求

  都是通过 gRPC 请求实现的

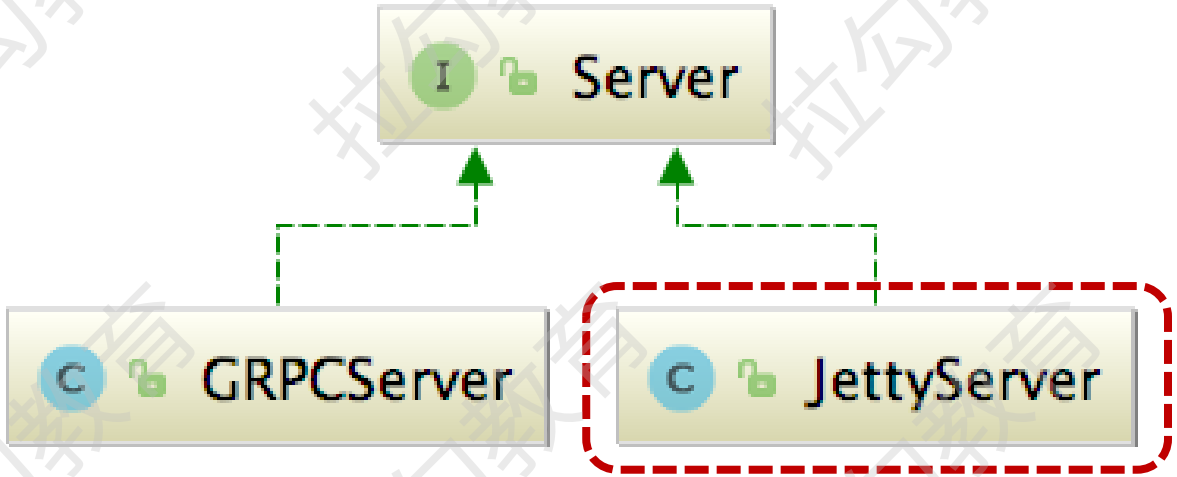


Server 核心实现

- **JettyServer** 用于接收 SkyWalking Agent 以及用户的 Http 请求

  在 SkyWalking 5.x 版本中，上述交互还可以通过 Http 请求完成

  用户从 SkyWalking Rocketbot 界面发起的请求，也是由 JettyServer 处理的



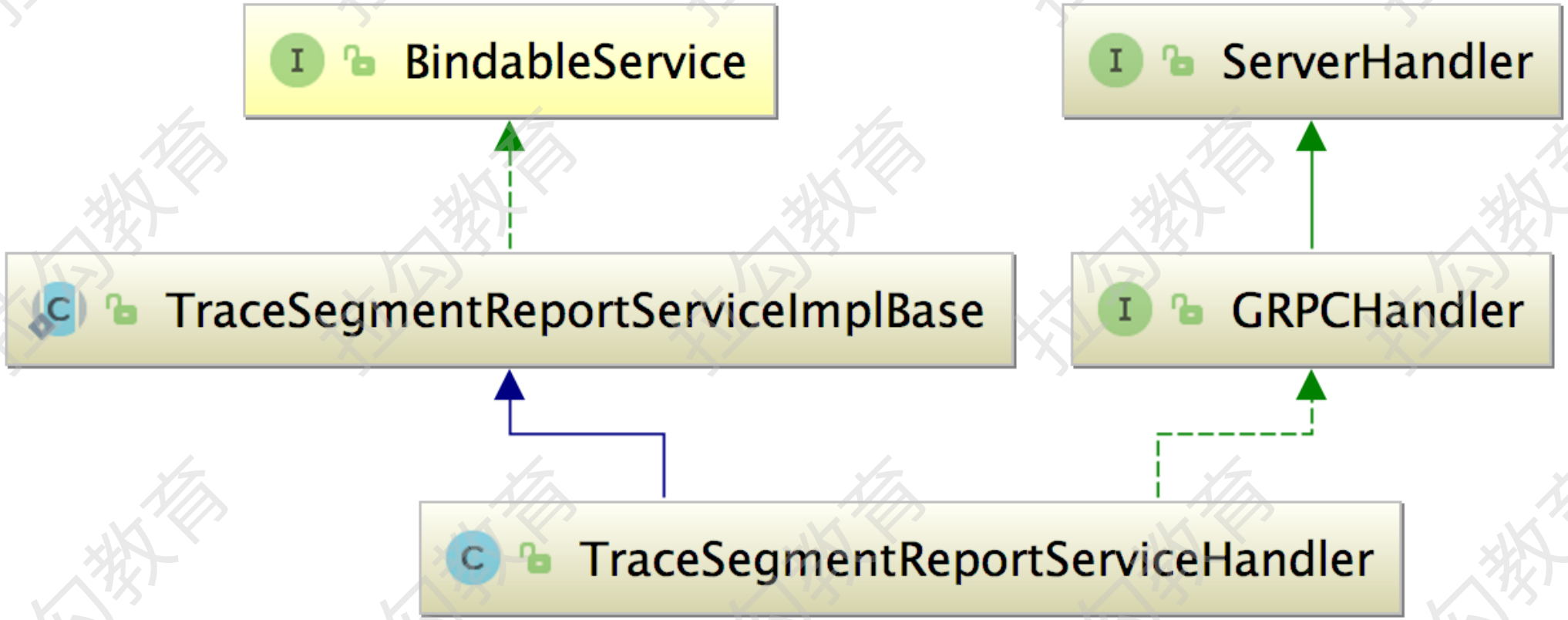
Server 核心实现


```
public void initialize() {
  //创建NettyServerBuilder，设置最大请求并发数、每个请求的最大长度等参数
  InetSocketAddress address = new InetSocketAddress(host, port);
  nettyServerBuilder = NettyServerBuilder.forAddress(address);
  nettyServerBuilder = nettyServerBuilder
          .maxConcurrentCallsPerConnection(maxConcurrentCallsPerConnection)
          .maxMessageSize(maxMessageSize);
}

public void start() throws ServerException {
  server = nettyServerBuilder.build(); //创建并启动io.grpc.Server
  server.start();
}
```

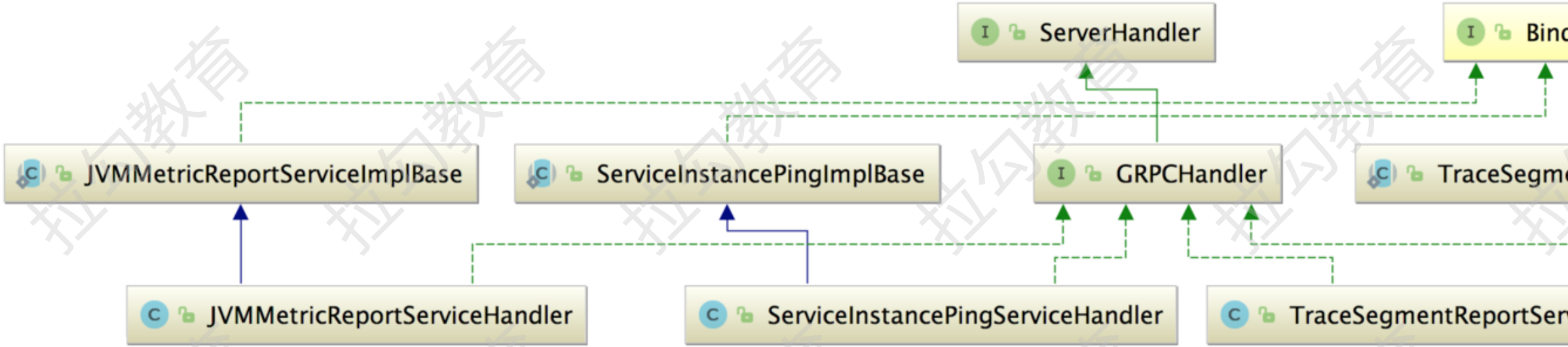
Server 核心实现


```
public void initialize() {
    //创建org.eclipse.jetty.server.Server对象
    server = new org.eclipse.jetty.server.Server(
        new InetSocketAddress(host, port));
    //创建ServletContextHandler对象，contextPath是其处理的路径
    servletContextHandler = new ServletContextHandler(NO_SESSIONS);
    servletContextHandler.setContextPath(contextPath);
    server.setHandler(servletContextHandler);
}

public void start() throws ServerException {
    server.start();
}
```

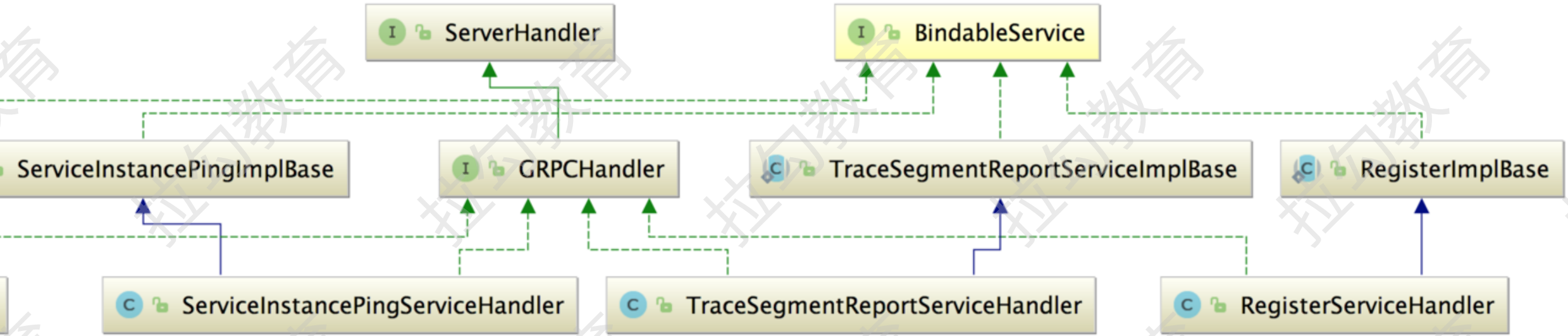
Server 核心实现


BindableService
ServerHandler
TraceSegmentReportServiceImplBase
GRPCHandler
TraceSegmentReportServiceHandler

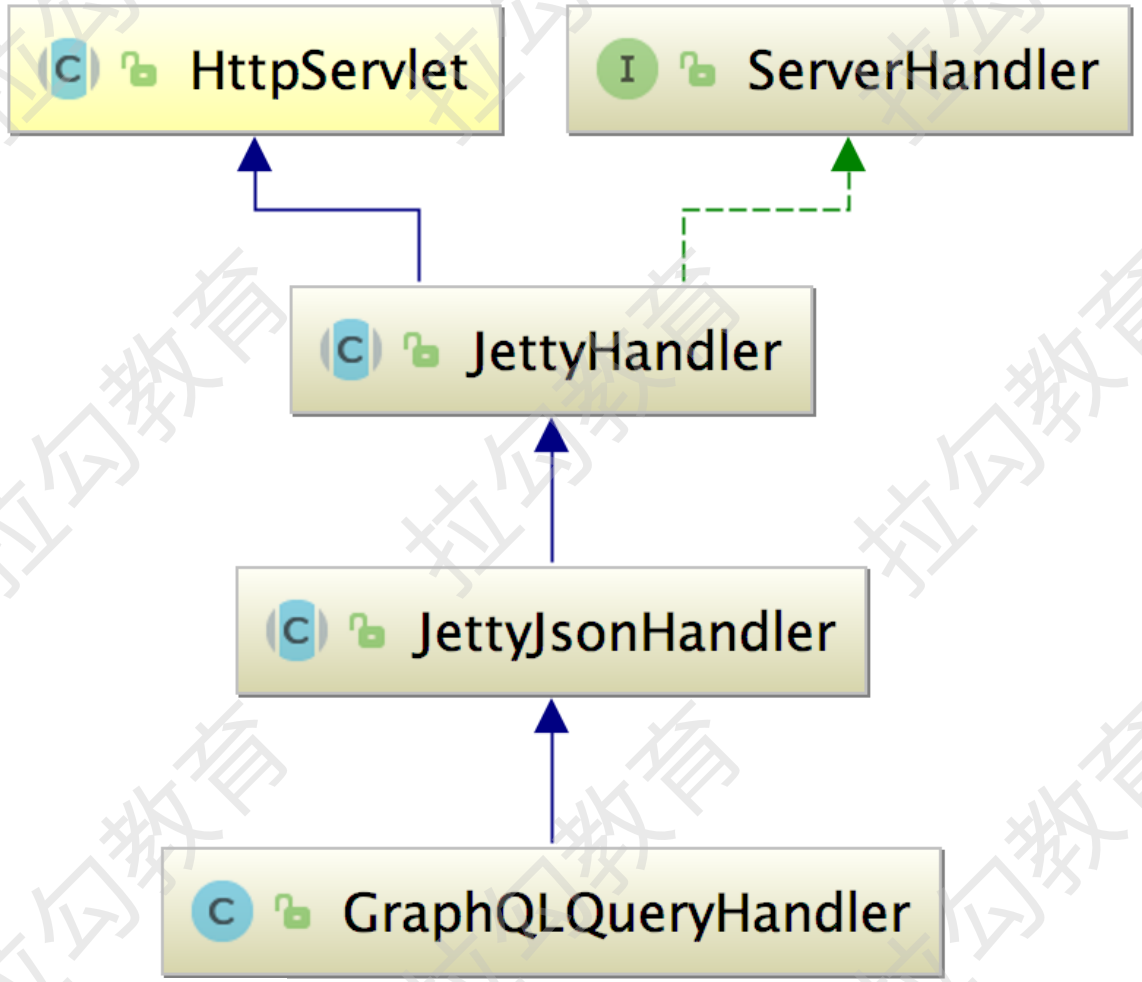
Server 核心实现



Server 核心实现



Server 核心实现



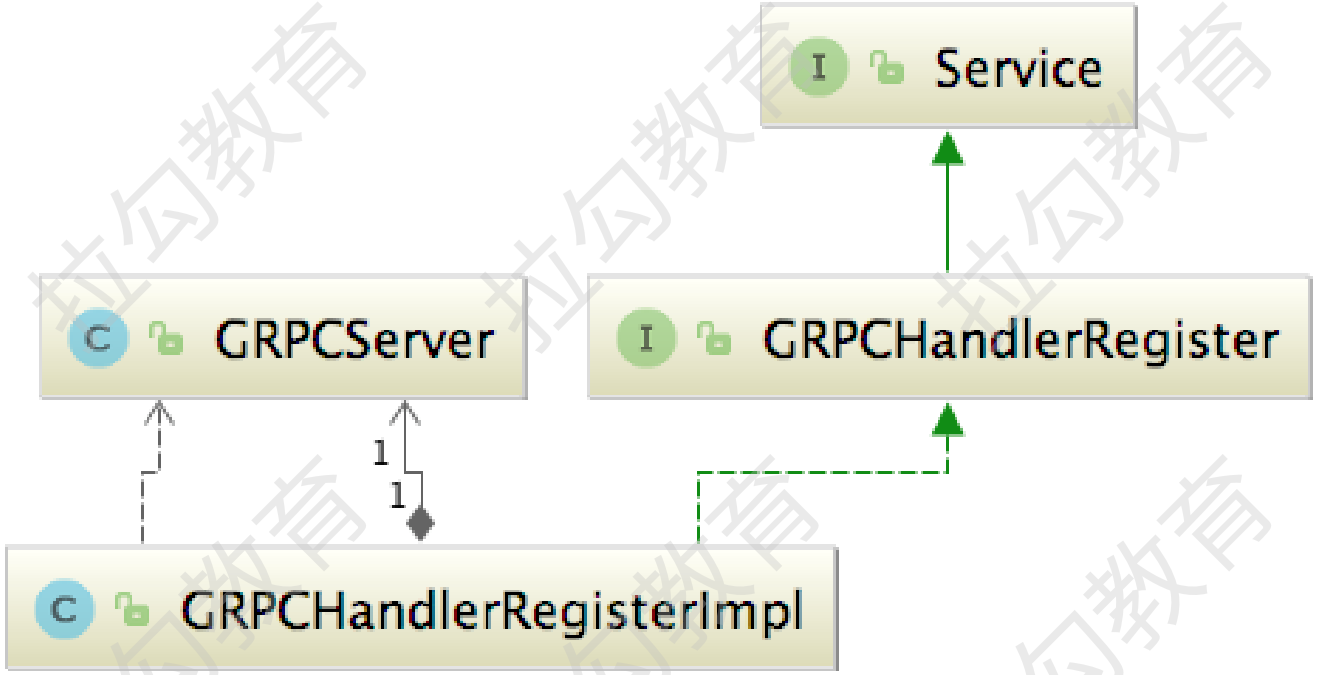
Server 核心实现

Server 核心实现

相关 Service 实现

OAP 其他模块在使用 **library-server 模块**提供的 Server 组件时

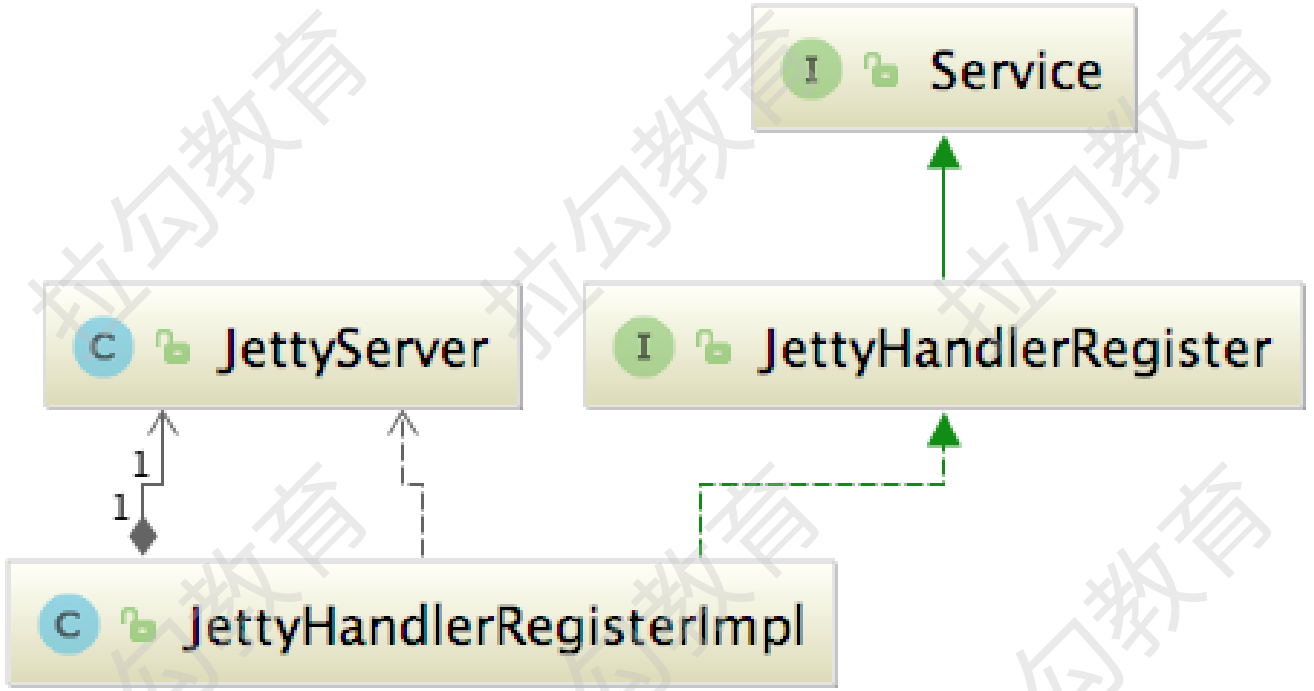
需要对其进行一层封装



相关 Service 实现

OAP 的 server-core 模块中定义了一个 **GRPCHandlerRegister** 接口

其实现中封装了一个 GRPCServer，并继承 Service 接口



Server 启动流程

org.apache.skywalking.oap.server.core.storage.StorageModule
org.apache.skywalking.oap.server.core.cluster.ClusterModule
org.apache.skywalking.oap.server.core.CoreModule
org.apache.skywalking.oap.server.core.query.QueryModule
org.apache.skywalking.oap.server.core.alarm.AlarmModule
org.apache.skywalking.oap.server.core.exporter.ExporterModule

Server 启动流程

```
public Class[] services() {
    List<Class> classes = new ArrayList<>();
    addServerInterface(classes);
    //省略向classes集合中添加其他Service接口的代码
    return classes.toArray(new Class[] {});
}

private void addServerInterface(List<Class> classes) {
    classes.add(GRPCHandlerRegister.class);
    classes.add(JettyHandlerRegister.class);
}
```

Server 启动流程


```
public void prepare() {
    grpcServer = new GRPCServer(...); //创建并初始化GRPCServer
    ... //省略设置GRPServer的代码
    grpcServer.initialize();
    //创建并初始化 JettyServer
    jettyServer = new JettyServer(...); //省略相关配置
    jettyServer.initialize();

    // GRPCServer封装成到GRPCHandlerRegisterImpl之中，然后注册成
    // GRPCHandlerRegister这个Service的实现
    this.registerServiceImplementation(GRPCHandlerRegister.class,
                        new GRPCHandlerRegisterImpl(grpcServer));
    // JettyServer封装成到JettyHandlerRegisterImpl之中，然后注册成
    // JettyHandlerRegister这个Service的实现
    this.registerServiceImplementation(JettyHandlerRegister.class,
                        new JettyHandlerRegisterImpl(jettyServer));
}
```

Server 启动流程


```
public void start() throws ModuleStartException {
    grpcServer.addHandler(new
RemoteServiceHandler(getManager()));
    grpcServer.addHandler(new HealthCheckServiceHandler());
}
```

Server 启动流程

- **HealthCheckServiceHandler**

  在 Cluster 模块中提供支持多种第三方服务发展组件的实现

  cluster-zookeeper-plugin 模块使用的 curator-x-discovery 扩展库底层

  是依赖 Zookeeper 的 Watcher 来监听一个 OAP 服务实例是否可用

  但有的第三方服务发现组件（例如 Consul）会依靠定期健康检查（Health Check）

  来检查一个 OAP 服务实例是否可用，此时 OAP 就需要保留一个接口来处理健康检查请求

  对 Consul 感兴趣的同学，可以参考下面两篇文档：

  - https://www.consul.io/docs/agent/checks.html
  - https://github.com/grpc/grpc/blob/master/doc/health-checking.md

- **RemoteServiceHandler**

  OAP 集群中各个 OAP 实例节点之间通信的接口

sharing-server-plugin 中的 Server 实例

SkyWalking OAP 需要接收外部请求的地方很多

例如：

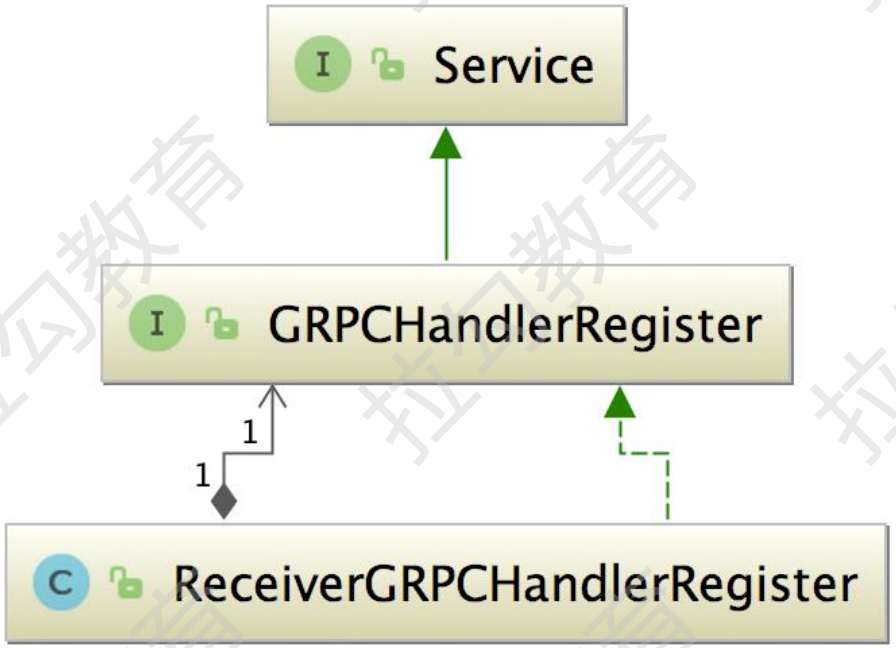
- Agent 上报的监控数据、
- SkyWalking Rocketbot 的查询请求
- OAP 集群中节点之间的相互通信
- 等等

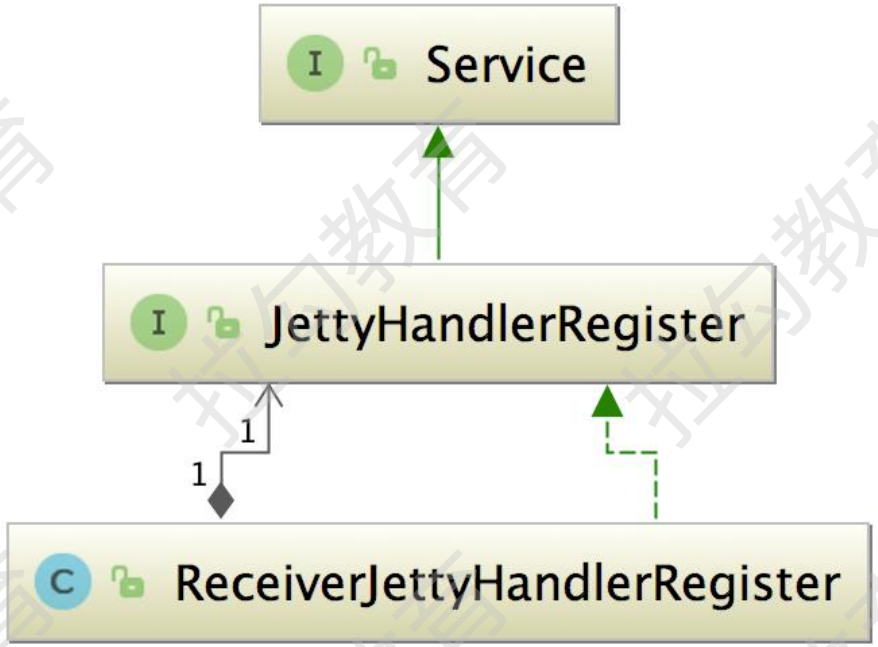
sharing-server-plugin 中的 Server 实例

拉勾教育
互联网人实战大学

```
public void prepare() {
    if (config.getGRPCPort() != 0) { //配置了独立端口，则启动独立GRPCServer
        grpcServer = new GRPCServer(...);
        grpcServer.initialize();
        //GRPCServer封装成到GRPCHandlerRegisterImpl之中，然后注册成
        //GRPCHandlerRegister这个Service的实现，与CoreModuleProvider相同
        this.registerServiceImplementation(GRPCHandlerRegister.class,
                                           new GRPCHandlerRegisterImpl(grpcServer));
    } else {
        //未指定独立GRPCServer的端口，则与CoreModule共用一个GRPCServer实例
        this.receiverGRPCHandlerRegister =
            new ReceiverGRPCHandlerRegister();
        this.registerServiceImplementation(GRPCHandlerRegister.class,
                                           receiverGRPCHandlerRegister);
    }
    //对JettyServer的处理相同，不再展开
}
```

sharing-server-plugin 中的 Server 实例





Server 的相关配置


```
core:
  default:
    restHost: ${SW_CORE_REST_HOST:0.0.0.0}
    restPort: ${SW_CORE_REST_PORT:12800}
    restContextPath: ${SW_CORE_REST_CONTEXT_PATH:/}
    gRPCHost: ${SW_CORE_GRPC_HOST:0.0.0.0}
    gRPCPort: ${SW_CORE_GRPC_PORT:11800}
```

```
public class CoreModuleConfig extends ModuleConfig {
    @Setter private String restHost;
    @Setter private int restPort;
    @Setter private String restContextPath;
    @Setter private String gRPCHost;
    @Setter private int gRPCPort;
    @Setter private int maxConcurrentCallsPerConnection;
    @Setter private int maxMessageSize;
```

Server 的相关配置


```
receiver-sharing-server:
  default:
      restHost: ${SW_CORE_REST_HOST:0.0.0.0}
      restPort: ${SW_CORE_REST_PORT:14800}
      restContextPath: ${SW_CORE_REST_CONTEXT_PATH:/xxx}
      gRPCHost: ${SW_CORE_GRPC_HOST:0.0.0.0}
      gRPCPort: ${SW_CORE_GRPC_PORT:13800}
```

```
public class SharingServerConfig extends ModuleConfig {
    private String restHost;
    private int restPort;
    private String restContextPath;
    private String gRPCHost;
    private int gRPCPort;
    private int maxConcurrentCallsPerConnection;
    private int maxMessageSize;
}
```

Next：加餐3《SkyWalking OAP 存储体系剖析》

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

L / A / G / O / U