ALIBABA CLOUD

阿里云

企业级分布式应用服务 EDAS

历史文档

文档版本：20220630

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.历史文档说明 05
2.使用Ali-Tomcat开发应用（不推荐） 06
2.1. Ali-Tomcat概述 06
2.2. 安装及开发环境配置 06
2.3. 开发HSF应用（SDK） 09

1.历史文档说明

什么是历史文档？是EDAS早期支持过的但现已不再迭代维护的历史功能的使用文档。新用户或没有使用过相关功能的老用户已无法再使用相关功能。

历史功能的适用用户：

- 在EDAS下线某历史集群前，创建过该集群的应用并在持续使用，一直未完成迁移的用户。
- 在EDAS使用新的功能方案来替代了旧的功能方案前，使用了旧的功能方案并且一直未部署升级或未完成迁移的用户。

2.使用Ali-Tomcat开发应用（不推荐）

2.1. Ali-Tomcat概述

开发HSF应用需要依赖Ali-Tomcat和Pandora。

Ali-Tomcat是EDAS中的服务运行时可依赖的一个容器，它主要集成了服务的发布、订阅、调用链追踪等一系列的核心功能。无论是开发环境还是运行时，您均可将应用程序发布在该容器中。

Pandora是一个轻量级的隔离容器，也就是taobao-hsf.sar。它用来隔离应用和中间件的依赖，也用来隔离中间件之间的依赖。EDAS的Pandora中集成了服务发现、配置推送和调用链跟踪等各种中间件功能产品插件。您可以利用该插件对EDAS应用进行服务监控、治理、跟踪、分析等全方位运维管理。

> **注意** 在EDAS中，只有WAR包格式的HSF应用才需要使用Ali-Tomcat。

2.2. 安装及开发环境配置

本文介绍如何安装Ali-Tomcat和Pandora，以及如何配置Eclipse和IntelliJ IDEA的开发环境。

安装Ali-Tomcat和Pandora

Ali-Tomcat和Pandora为EDAS中的服务运行时所依赖的容器，集成了服务的发布、订阅、调用链追踪等一系列核心功能，应用程序须发布在该容器中运行。

> **注意** 请使用JDK 1.7及以上版本。

1. 下载Ali-Tomcat-8或Ali-Tomcat-7，保存并解压至相应的目录（例如*d:\work\tomcat*）。
2. 下载所需版本的Pandora容器，保存并解压至Ali-Tomcat的deploy目录（*d:\work\tomcat\deploy*）下。更多信息，请参见容器版本说明。
3. 查看Pandora容器的目录结构。
   - Linux系统中，在相应路径下执行**tree -L 2 deploy/** 命令查看目录结构。

```
d:\work\tomcat >  tree -L 2 deploy/
            deploy/
                └── taobao-hsf.sar
                    ├── META-INF
                    ├── lib
                    ├── log.properties
                    ├── plugins
                    ├── sharedlib
                    └── version.properties
```

   - Windows中，直接进入相应路径进行查看。

如果您在安装和使用Ali-Tomcat和Pandora过程中遇到问题，请参见Ali-Tomcat问题和Pandora问题。

配置Eclipse开发环境

配置Eclipse需要下载Tomcat4E插件，并存放在安装Ali-Tomcat和Pandora的保存路径中，完成配置后可以直接在Eclipse中发布、调试本地代码。

1. 下载Tomcat4E插件，压缩包内容如下图所示。

名称	修改日期	类型
features	2016/3/15 10:31	文件夹
plugins	2016/3/15 10:31	文件夹
artifacts.jar	2016/3/9 17:10	Executable Jar File
content.jar	2016/3/9 17:10	Executable Jar File

2. 打开Eclipse，在菜单栏中选择**Help > Install New Software**。
3. 在Install对话框中**Work with**区域右侧单击**Add**，且在弹出的**Add Repository**对话框中单击**Local**，并在弹出的对话框中选中已下载并解压的Tomcat4E插件的目录（*d:\work\tomcat4e*），单击**OK**。
4. 返回Install对话框，单击**Select All**，并单击**Next**。

   后续步骤，请按界面提示操作。安装完成后，请重启Eclipse，使Tomcant4E插件生效。
5. 重启Eclipse后，在Eclipse菜单中选择**Run As > Run Configurations**。
6. 选择左侧导航选项中的**AliTomcat Webapp**，单击上方的**New launch configuration**图标。
7. 在弹出的界面中，选择**AliTomcat**页签，并在**taobao-hsf.sar Location**区域单击**Browse**，选择本地的Pandora路径，如：*d:\work\tomcat\deploy\taobao-hsf.sar*。
8. 单击**Apply**或**Run**，完成设置。

   一个工程只需配置一次，下次可直接启动。
9. 查看工程运行的打印信息，如果出现下图Pandora Container的相关信息，即说明Eclipse开发环境配置成功。

```
*******************************************************************************
**                                                                           **
**                                  Pandora Container                        **
**                                                                           **
**      Pandora Host:      [redacted]                                         **
**      Pandora Version:   2.1.4                                              **
**      SAR Version:       edas.sar.V3.3.4.dev                                **
**      Package Time:      2017-11-20 09:58:18                                **
**                                                                           **
**      Plug-in Modules: 11                                                  **
**                                                                           **
**         edas-assist ..................................... 1.6             **
**         pandora-qos-service ............................. edas215         **
**         spas-sdk-client ................................. 1.2.4-SNAPSHOT  **
**         eagleeye-core ................................... 1.6.0.2-SNAPSHOT **
**         vipserver-client ................................ 4.6.8-SNAPSHOT  **
**         diamond-client .................................. acm-3.8.5       **
**         spas-sdk-service ................................ 1.2.4.nodc-SNAPSHOT **
**         config-client ................................... 2.0.1-edas-SNAPSHOT **
**         unitrouter ...................................... 1.0.11          **
**         sentinel-plugin ................................. 2.12.2_edas     **
**         hsf ............................................. 2.2.4.2         **
**                                                                           **
**      [WARNING] All these plug-in modules will override maven pom.xml dependencies.  **
**      More: http://gitlab.alibaba-inc.com/middleware-container/pandora/wikis/home  **
**                                                                           **
*******************************************************************************
```

配置IntelliJ IDEA开发环境

> **注意** 目前仅支持IDEA商业版，社区版暂不支持。

1. 运行IntelliJ IDEA。
2. 在菜单栏中选择**Run > Edit Configurations**。
3. 在**Run/Debug Configurations**页面左侧的导航栏中选择**Defaults > Tomcat Server > Local**。
4. 配置AliTomcat。
   i. 在右侧页面单击**Server**页签，并在**Application Server**区域单击**Configure**。
   ii. 在**Application Server**页面右上角单击**+**，并在**Tomcat Server**对话框中设置**Tomcat Home**和**Tomcat base directory** 路径，且单击**OK**。

   将Tomcat Home的路径设置为本地解压后的Ali-Tomcat路径，Tomcat base directory可以自动使用该路径，无需再设置。
5. 在**Application Server**区域的下拉菜单中，选择刚刚配置好的Ali-Tomcat。
6. 在**VM Options**区域的文本框中，设置JVM启动参数指向Pandora的路径及设置本地轻量级配置及注册中心的vipserver端口。

   例如 `-Dpandora.location=d:\work\tomcat\deploy\taobao-hsf.sar -Dvipserver.server.port=8080`

   将*d:\work\tomcat\deploy\taobao-hsf.sar*替换为在本地安装Pandora的实际路径。 `-Dvipserver.server.port=8080` 参数用于指定应用连接本地轻量级配置及注册中心的vipserver端口，在本地开发环境使用新版本的轻量级配置及注册中心时，能有效提升本地应用启动速度。

   当应用部署到EDAS时，不需要添加上面这个JVM属性参数。
7. 单击**Apply**或**OK** 完成配置。

2.3. 开发HSF应用（SDK）

介绍如何使用EDAS-SDK快速开发HSF应用，完成服务注册与发现。

前提条件

在开发应用前，您已经完成以下工作：

- 配置EDASSAE的私服地址和轻量级配置及注册中心
- 启动轻量级配置及注册中心

下载Demo工程

您可以按照本文的步骤一步步搭建工程，也可以直接下载本文对应的示例工程，或者使用Git下载：**git clone https://github.com/aliyun/alibabacloud-microservice-demo.git**。

该项目包含了众多示例工程，本文对应的示例工程位于*alibabacloud-microservice-demo/microservice-doc-demo/hsf-ali-tomcat*，包含*itemcenter-api*、*itemcenter*和*detail*三个Maven工程文件夹。

- *itemcenter-api*：提供接口定义
- *itemcenter*：服务提供者
- *detail*：消费者服务

> **说明** 请使用JDK 1.7及以上版本。

定义服务接口

HSF服务基于接口实现，当接口定义好之后，生产者将使用该接口实现具体的服务，消费者也基于此接口去订阅服务。

在Demo的*itemcenter-api*工程中，定义了一个服务接口 `com.alibaba.edas.carshop.itemcenter.ItemService` 。

```
public interface ItemService {
    public Item getItemById(long id);
    public Item getItemByName(String name);
}
```

该服务接口将提供两个方法：getItemById与getItemByName。

开发服务提供者

服务提供者将实现服务接口以提供具体服务。同时，如果使用了Spring框架，还需要在xml文件中配置服务属性。

> **说明** Demo工程中的itemcenter文件夹为服务提供者的示例代码。

1. 实现服务接口。

   请参考*ItemServiceImpl.java*文件中的示例代码构建服务接口。

```
public class ItemServiceImpl implements ItemService {
    @Override
    public Item getItemById( long id ) {
        Item car = new Item();
        car.setItemId( 1l );
        car.setItemName( "Mercedes Benz" );
        return car;
    }
    @Override
    public Item getItemByName( String name ) {
        Item car = new Item();
        car.setItemId( 1l );
        car.setItemName( "Mercedes Benz" );
        return car;
    }
}
```

2. 服务提供者配置。

   实现服务接口中实现了com.alibaba.edas.carshop.itemcenter.ItemService，并在两个方法中返回了Item对象。代码开发完成之后，除了在web.xml中进行必要的常规配置，您还需要增加相应的Maven依赖，同时在Spring配置文件使用 `<hsf />` 标签注册并发布该服务。

   i. 在*pom.xml*中添加Maven依赖。

```
<dependencies>
    <!-- 添加servlet的依赖 -->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>servlet-api</artifactId>
        <version>2.5</version>
        <scope>provided</scope>
    </dependency>
    <!-- 添加服务接口的依赖 -->
    <dependency>
        <groupId>com.alibaba.edas.carshop</groupId>
        <artifactId>itemcenter-api</artifactId>
        <version>1.0.0-SNAPSHOT</version>
    </dependency>
    <!-- 添加Spring的依赖 -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>2.5.6(及其以上版本)</version>
    </dependency>
    <!-- 添加edas-sdk的依赖 -->
    <dependency>
          <groupId>com.alibaba.edas</groupId>
           <artifactId>edas-sdk</artifactId>
          <version>1.8.1</version>
    </dependency>
</dependencies>
```

   ii. 在*hsf-provider-beans.xml*文件中增加Spring关于HSF服务的配置。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:hsf="http://www.taobao.com/hsf"
    xmlns="http://www.springframework.org/schema/beans"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.taobao.com/hsf
    http://www.taobao.com/hsf/hsf.xsd" default-autowire="byName">
    <!-- 定义该服务的具体实现 -->
    <bean id="itemService" class="com.alibaba.edas.carshop.itemcenter.ItemService
Impl" />
    <!-- 用hsf:provider标签表明提供一个服务生产者 -->
    <hsf:provider id="itemServiceProvider"
        <!-- 用interface属性说明该服务为此类的一个实现 -->
        interface="com.alibaba.edas.carshop.itemcenter.ItemService"
        <!-- 此服务具体实现的Spring对象 -->
        ref="itemService"
        <!-- 发布该服务的版本号，可任意指定，默认为1.0.0 -->
        version="1.0.0"
    </hsf:provider>
</beans>
```

上面的示例为基本配置，您也可以根据您的实际需求，参考下面的生产者服务属性列表，增加其它配置。

属性	描述
interface	必须配置，类型为 [String]，为服务对外提供的接口。
version	可选配置，类型为 [String]，含义为服务的版本，默认为1.0.0。
clientTimeout	该配置对接口中的所有方法生效，但是如果客户端通过methodSpecials属性对某方法配置了超时时间，则该方法的超时时间以客户端配置为准。其他方法不受影响，还是以服务端配置为准。
serializeType	可选配置，类型为 `[String(hessian|java)]`，含义为序列化类型，默认为hessian。
corePoolSize	单独针对这个服务设置核心线程池，从公用线程池中划分出来。
maxPoolSize	单独针对这个服务设置线程池，从公用线程池中划分出来。
enableTXC	开启分布式事务GTS。
ref	必须配置，类型为 [ref]，为需要发布为HSF服务的Spring Bean ID。

属性	描述
methodSpecials	可选配置，用于为方法单独配置超时时间（单位ms），这样接口中的方法可以采用不同的超时时间。该配置优先级高于上面的clientTimeout的超时配置，低于客户端的methodSpecials配置。

服务创建及发布存在以下限制：

名称	示例	限制大小	是否可调整
{服务名}:{版本号}	com.alibaba.edas.testcase.api.TestCase:1.0.0	最大192字节	否
组名	HSF	最大32字节	否
单个Pandora应用实例发布的服务数	N/A	最大800个	可在应用基本信息页面单击**应用设置**部分右侧的设置，在下拉列表中选择**JVM**，在弹出的**应用设置**对话框中进入**自定义 > 自定义参数**，`-DCC.pubCountMax=1200` 属性参数（该参数值可根据应用实际发布的服务数调整）。

服务提供者属性配置示例：

```
<bean id="impl" class="com.taobao.edas.service.impl.SimpleServiceImpl" />
<hsf:provider
        id="simpleService"
        interface="com.taobao.edas.service.SimpleService"
        ref="impl"
        version="1.0.1"
        clientTimeout="3000"
        enableTXC="true"
        serializeType="hessian">
        <hsf:methodSpecials>
            <hsf:methodSpecial name="sum" timeout="2000" />
        </hsf:methodSpecials>
</hsf:provider>
```

开发服务消费者

消费者订阅服务从代码编写的角度分为两个部分。

- Spring的配置文件使用标签 `<hsf:consumer/>` 定义好一个Bean。
- 在使用的时候从Spring的context中将Bean取出来。

> **说明** Demo工程中的detail文件夹为消费者服务的示例代码。

与生产者相同，消费者的服务属性配置分为Maven依赖配置与Spring的配置。

1. 配置服务属性。
   i. 在*pom.xml*文件中添加Maven依赖。
   ii. 在*hsf-consumer-beans.xml*文件中添加Spring关于HSF服务的配置。

   增加消费者的定义，HSF框架将根据该配置文件去服务中心订阅所需的服务。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:hsf="http://www.taobao.com/hsf"
        xmlns="http://www.springframework.org/schema/beans"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
        http://www.taobao.com/hsf
        http://www.taobao.com/hsf/hsf.xsd" default-autowire="byName">
        <!-- 消费一个服务示例 -->
        <hsf:consumer
            <!-- Bean ID，在代码中可根据此ID进行注入从而获取consumer对象  -->
            id="item"
            <!-- 服务名，与服务提供者的相应配置对应，HSF将根据interface + version查询并订
阅所需服务  -->
            interface="com.alibaba.edas.carshop.itemcenter.ItemService"
            <!-- 版本号，与服务提供者的相应配置对应，HSF将根据interface + version查询并订
阅所需服务  -->
            version="1.0.0">
        </hsf:consumer>
</beans>
```

2. 服务消费者配置。

   请参考*StartListener.java*文件中的示例进行。

```
public class StartListener implements ServletContextListener{
    @Override
    public void contextInitialized( ServletContextEvent sce ) {
        ApplicationContext ctx = WebApplicationContextUtils.getWebApplicationContext( s
ce.getServletContext() );
        // 根据Spring配置中的Bean ID“item” 获取订阅到的服务final ItemService itemService =
( ItemService ) ctx.getBean( "item" );
        ……
        // 调用服务ItemService的getItemById方法System.out.println( itemService.getItemByI
d( 1111 ) );
        // 调用服务ItemService的getItemByName方法System.out.println( itemService.getItemB
yName( "myname is le" ) );
        ……
    }
}
```

上面的示例中为基本配置，您也可以根据您的实际需求，参考下面的服务属性列表，增加其它配置。

属性	描述
interface	必须配置，类型为 [String]，为需要调用的服务的接口。

属性	描述
version	可选配置，类型为 [String]，为需要调用的服务的版本，默认为1.0.0。
methodSpecials	可选配置，为方法单独配置超时时间（单位ms）。这样接口中的方法可以采用不同的超时时间，该配置优先级高于服务端的超时配置。
target	主要用于单元测试环境和开发环境中，手动地指定服务提供端的地址。如果不想通过此方式，而是通过配置中心推送的目标服务地址信息来指定服务端地址，可以在消费者端指定 -Dhsf.run.mode=0。
connectionNum	可选配置，为支持设置连接到server连接数，默认为1。在小数据传输，要求低延迟的情况下设置多一些，会提升TPS。
clientTimeout	客户端统一设置接口中所有方法的超时时间（单位ms）。超时时间设置优先级由高到低是：客户端methodSpecials，客户端接口级别，服务端methodSpecials，服务端接口级别 。
asyncallMethods	可选配置，类型为 [List]，设置调用此服务时需要采用异步调用的方法名列表以及异步调用的方式。默认为空集合，即所有方法都采用同步调用。
maxWaitTimeForCsAddress	配置该参数，目的是当服务进行订阅时，会在该参数指定时间内，阻塞线程等待地址推送，避免调用该服务时因为地址为空而出现地址找不到的情况。若超过该参数指定时间，地址还是没有推送，线程将不再等待，继续初始化后续内容。注意，在应用初始化时，需要调用某个服务时才使用该参数。如果不需要调用其它服务，请勿使用该参数，会延长启动时间。

消费者服务属性配置示例

```
<hsf:consumer
      id="service"
      interface="com.taobao.edas.service.SimpleService"
      version="1.1.0"
      clientTimeout="3000"
      target="10.1.6.57:12200?_TIMEOUT=1000"
      maxWaitTimeForCsAddress="5000">
      <hsf:methodSpecials>
           <hsf:methodSpecial name="sum" timeout="2000" ></hsf:methodSpecial>
      </hsf:methodSpecials>
</hsf:consumer>
```

本地运行服务

完成代码、接口开发和服务配置后，在Eclipse或IDEA中，可直接以Ali-Tomcat运行该服务（具体请参见安装及开发环境配置）。

在开发环境配置时，有一些额外JVM启动参数来改变HSF的行为，具体如下：

属性	描述
-Dhsf.server.port	指定HSF的启动服务绑定端口，默认值为12200。
-Dhsf.serializer	指定HSF的序列化方式，默认值为hessian。
-Dhsf.server.max.poolsize	指定HSF的服务端最大线程池大小，默认值为720。
-Dhsf.server.min.poolsize	指定HSF的服务端最小线程池大小。默认值为50。
-DHSF_SERVER_PUB_HOST	指定对外暴露的IP，如果不配置，使用 -Dhsf.server.ip的值。
-DHSF_SERVER_PUB_PORT	指定对外暴露的端口，该端口必须在本机被监听，并对外开放了访问授权，默认使用 -Dhsf.server.port的配置，如果 -Dhsf.server.port没有配置，默认使用12200。

本地查询HSF服务

在开发调试的过程中，如果您的服务是通过轻量级注册配置中心进行服务注册与发现，就可以通过EDAS控制台查询某个应用提供或调用的服务。

假设您在一台IP为192.168.1.100的机器上启动了EDAS配置中心。

- 进入http://192.168.1.100:8080/
- 在左侧菜单栏单击**服务列表**，输入服务名、服务组名或者IP地址进行搜索，查看对应的服务提供者以及服务调用者。

> **说明** 配置中心启动之后默认选择第一块网卡地址做为服务发现的地址，如果开发者所在的机器有多块网卡的情况，可设置启动脚本中的SERVER_IP变量进行显式的地址绑定。

常见查询案例

- 提供者列表页
  - 在搜索框中输入IP地址，单击**搜索**，即可查询该IP地址的物理机所提供的服务。
  - 在搜索框中输入服务名或服务分组，即可查询提供该服务的IP地址。
- 调用者列表页
  - 在搜索框中输入IP地址，单击**搜索**，即可查询该IP地址的物理机所调用的服务。
  - 在搜索框中输入服务名或服务分组，即可查询调用该服务的IP地址。

部署到EDAS

本地使用轻量级配置及注册中心的应用可以直接部署到EDAS中，无需做任何修改，注册中心会被自动替换为EDAS上的注册中心。

正常打包出可供EDAS-Container运行的WAR包，需要添加如下的Maven打包插件

1. 在*pom.xml*文件中添加以下打包插件的配置。

```
<build>
    <finalName>itemcenter</finalName>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.1</version>
        </plugin>
    </plugins>
</build>
```

2. 执行**mvn clean package**将本地的程序打成WAR包。

应用运行时环境需要选择EDAS-Container。

具体部署操作详情请参见在ECS集群中部署应用和在K8s集群中使用镜像部署Java微服务应用。