

EDAS 开发者指南

此文档提供关于阿里云中间件技术产品 EDAS (Enterprise Distributed Application Service) 的分布式应用的开发指南。

2016-05

EDAS 开发者指南	1
1. 开发工具准备	2
1.1 Ali-Tomcat 安装	2
1.1.1 Eclipse 安装 Tomcat4E	2
1.1.2 IntelliJ Idea 安装 Ali-Tomcat	5
1.2 HotCode 插件安装(可选)	7
1.2.1 Eclipse 安装 Hotcode 插件	8
1.2.2 IntelliJ Idea 安装 Hotcode 插件	9
2. 开发环境搭建	10
2.1 启动 EDAS 配置中心	10
2.2 使用 EDAS 配置中心	10
2.3 通过 EDAS 配置中心查询某个应用提供或调用的服务	11
3. HSF 开发	12
3.1 HSF 简介	12
3.3.1PC	12
3.3.2 ConfigServer(地址注册中心)	12
3.3.3 Diamond(配置中心)	12
3.3.4 Pandora	12
3.2 HSF 标签清单	12
3.3 HSF 基础开发	14
3.3.1 创建 Web 项目	15
3.3.2 添加 Maven 依赖	15
3.3.3 编写发布 Hsf 的服务	15
3.3.4 发布服务配置文件	16
3.3.5 消费者服务配置文件	18
3.3.6 Demo 下载地址	19
3.4 HSF 高级开发	19
4. FAQ	19
4.1 EDAS 日志说明	19
4.2 Tomcat FAQ	20
4.2.1 修改应用访问根路径	20
4.2.2 Tomcat jvm 内存配置	20
4.2.3 Tomcat 在启动过程中卡住	21
4.3 HSF FAQ	21
4.3.1 HSF-0001	21
4.3.2 HSF-0002	21

4.3.3 HSF-0003	22
4.3.4 HSF-0027	22
4.3.5 HSF-0031	22

声明:

使用 EDAS 产品之前, 我们默认您已经掌握基本的编程技巧, 包括但不限于熟练使用 JAVA 编程语言, 掌握相关的开发工具 (Eclipse、IntelliJ Idea), 熟悉相关开发规范 (Maven、Spring 等)

1. 开发工具准备

目前开发工具主要介绍 Eclipse 和 IntelliJ Idea 相关插件的安装配置。

1.1 Ali-Tomcat 安装

通过在 Eclipse 安装 Tomcat4e 插件, 或者在 IntelliJ Idea 安装配置 Ali-tomcat, 可以快速方便地启动并调试基于 EDAS 服务化框架 HSF 开发的应用。


1.1.1 Eclipse 安装 Tomcat4E

Eclipse Tomcat4E 安装

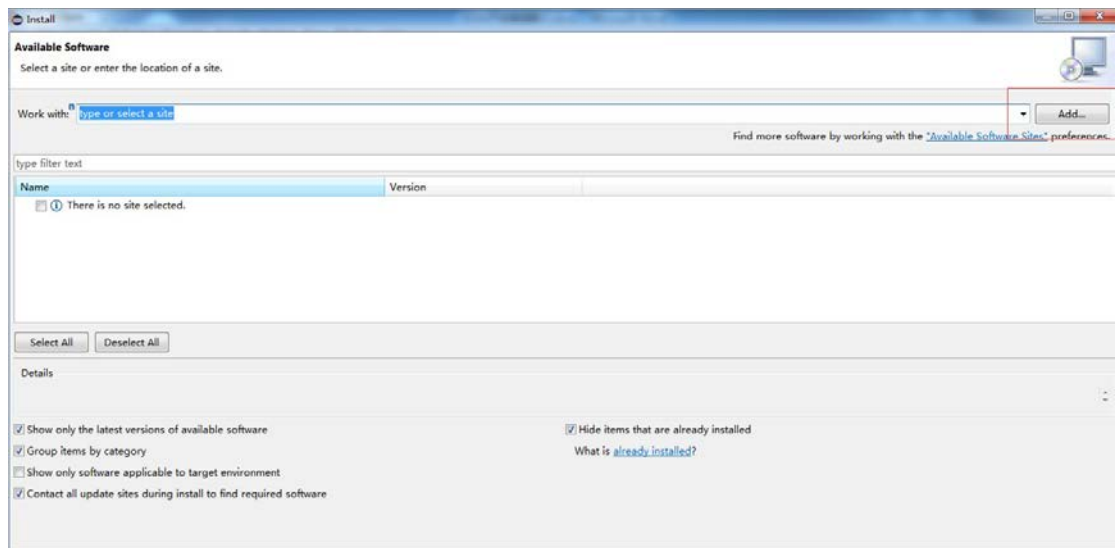
1 通过以下地址下载插件,

<http://edas-public.oss-cn-hangzhou.aliyuncs.com/tomcat4e.zip>

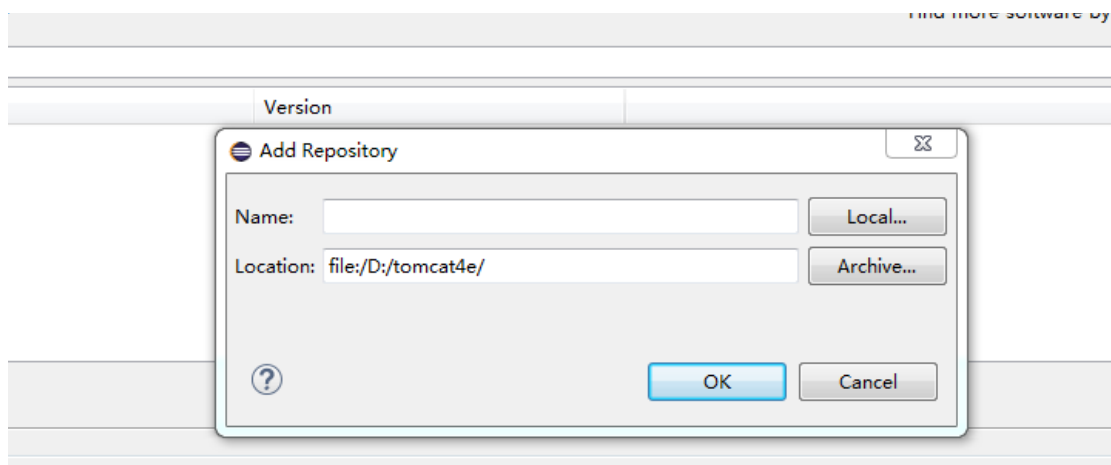
2. 解压压缩包里面内容如下:

						
C:\Users\shenyu.yxl\Downloads\tomcat4e (2).zip\						
名称	大小	压缩后大小	修改时间	创建时间	访问时间	
features	2 192	1 271	2016-03-15 10:31	2016-03-15 1...	2016-03-15 1...	
plugins	11 454 015	11 441 691	2016-03-15 10:31	2016-03-15 1...	2016-03-15 1...	
artifacts.jar	722	694	2016-03-09 17:10	2016-03-15 1...	2016-03-15 1...	
content.jar	1 314	1 319	2016-03-09 17:10	2016-03-15 1...	2016-03-15 1...	

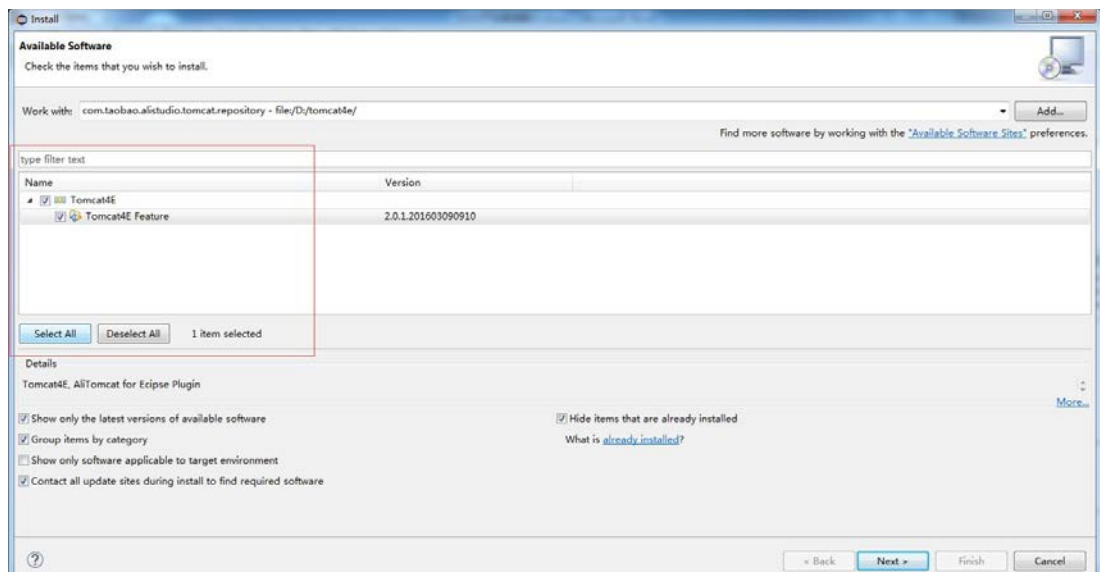
2. 打开 eclipse, 点击工具栏菜单 Help 按钮, 选择 Install New SoftWare, 弹出如下页面:



3. 点击 Add 按钮，选择 local 按钮，找到本地解压的 tomcat4e，点击 ok:



4. 点击 Select All 选择列表 tomcat4e 插件，点击 next 完成安装:

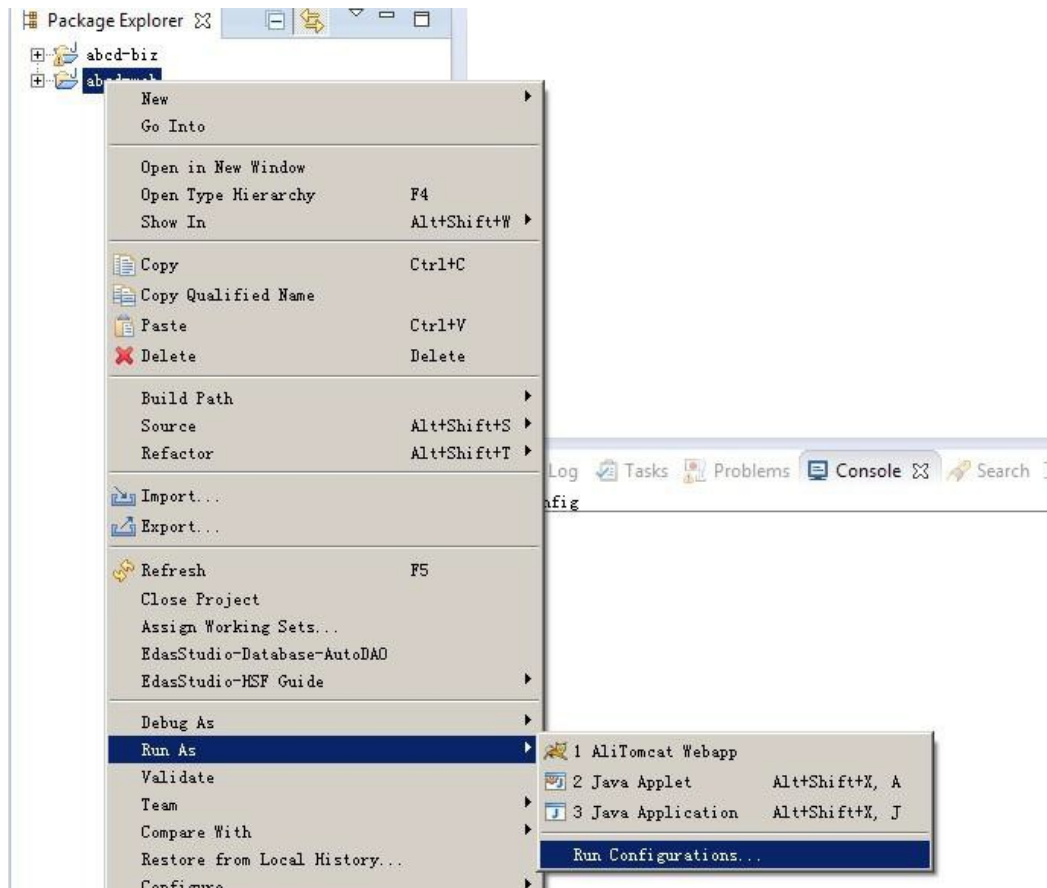


Eclipse tomcat4e 配置

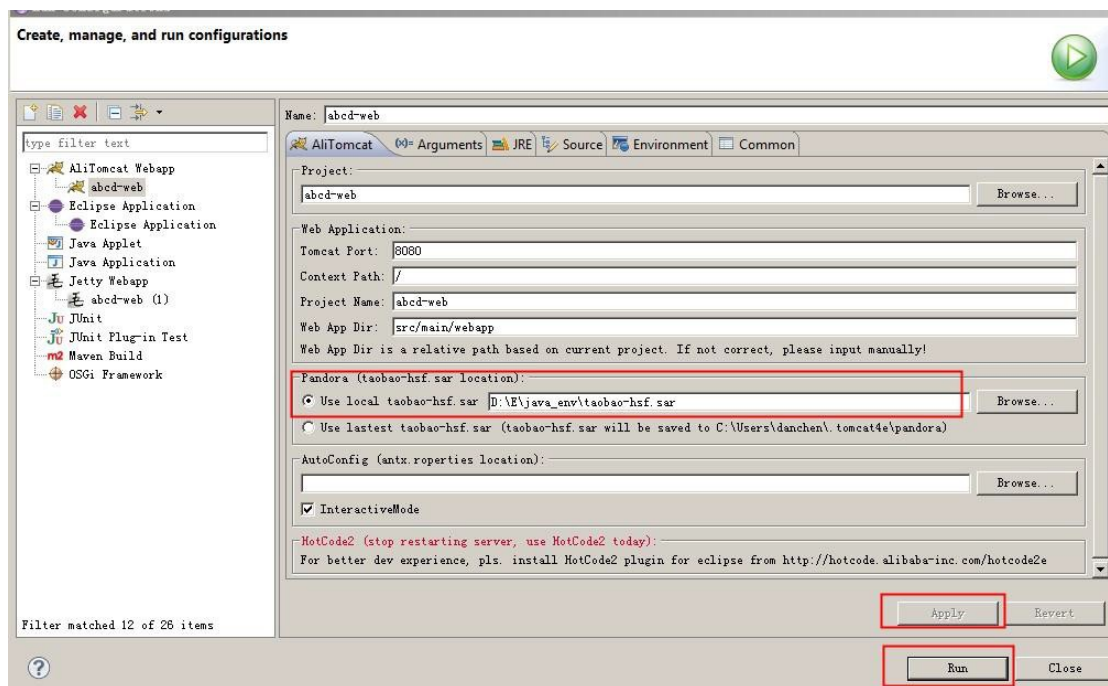
1. 从以下地址下载 Pandora (taobao-hsf.sar):

http://edas-public.oss-cn-hangzhou.aliyuncs.com/install_package/pandora/unauth/taobao-hsf.tgz

1. 在 Eclipse 里创建了 web 工程 (已创建 web 工程), 点击 run configuration 如下图:



2. 在新建的应用配置 pandora 路径, 在图中红色位置配置 pandora 包路径:



4. 点击 Apply 和 Run。一个工程只需要配置一次,下次可以直接启动了。

1.1.2 IntelliJ Idea 安装 Ali-Tomcat

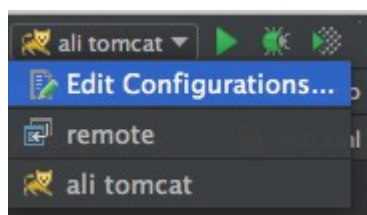
注意:

目前通过 IDEA 使用 Ali-Tomcat 的方式, 仅 IDEA 的商业版才支持。 1. 下

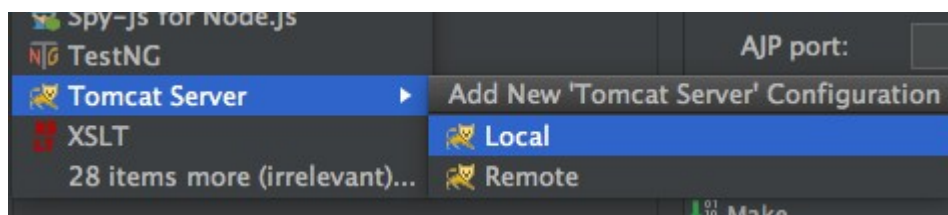
载 Ali Tomcat:

http://edas-public.oss-cn-hangzhou.aliyuncs.com/install_package/tomcat/taobao-tomcat-7.0.59.tgz

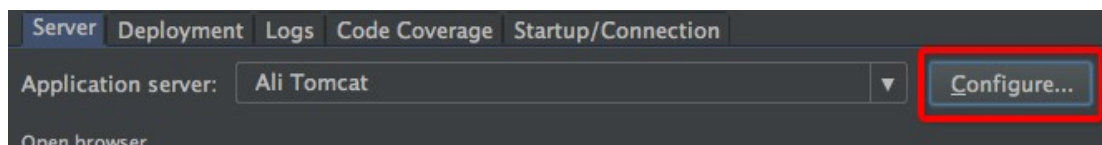
2. 从菜单中或者工具栏中点击 Run 选择 Edit Configuration:



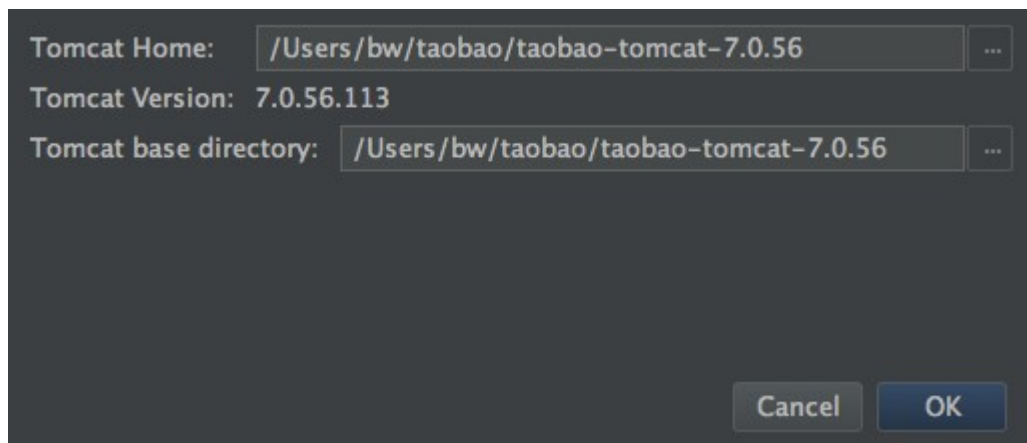
3. 点击界面左上方的+号增加一个 Tomcat 的本地实例:



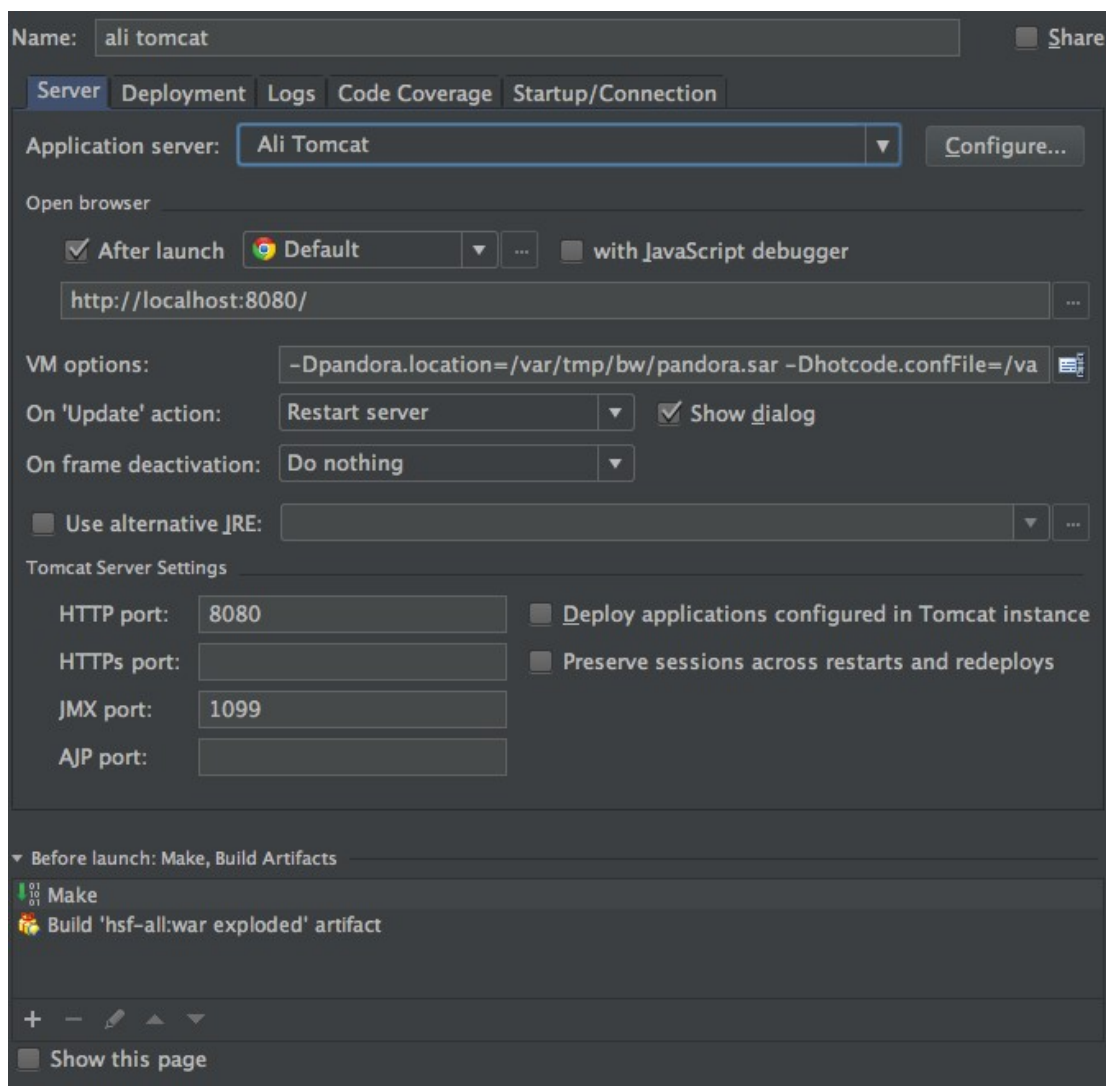
4. 通过 Application Server -> Configuration... 指定 Tomcat 的位置:



5. 在弹出的 Application Servers 对话框里单击+, 并选择 Ali Tomcat 7.0.59 的安装目录



6. 配置好的 Run/Debug Configuration 中 Application Server 指向刚刚配置好的 Ali Tomcat 实例



7. 配置 Pandora (taobao-hsf. sar) 下载

地址：

http://edas-public.oss-cn-hangzhou.aliyuncs.com/install_package/pandora/unauth/taobao-hsf.tgz

说明：

如果把 sar 包放到 tomcat 的安装目录(\$CATALINA_HOME)的 deploy 目录下，那么不需要指定 `-Dpandora.location` 参数。

如果把 sar 包放到其他地方，请指定 JVM 参数 `-Dpandora.location` 来配置 Pandora (taobao-hsf. sar) 的位置。

1.2 HotCode 插件安装(可选)

HotCode 插件的作用是, 当你启动应用后, 修改应用代码, 保存代码后, 将会立刻生效, 而不需要重启整个应用。 这一功能在一些大型应用的开发中, 对于提高调试效率会很有帮助。

1.2.1 Eclipse 安装 Hotcode 插件

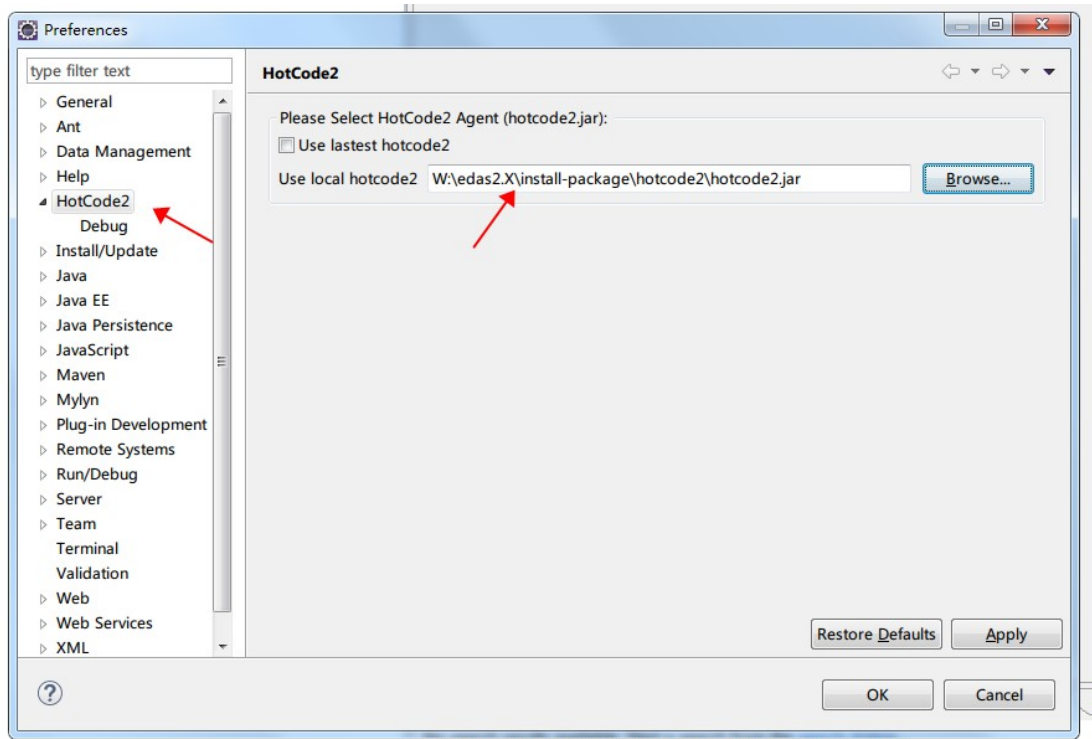
1. 通过以下地址下载插件，

<http://edas-public.oss-cn-hangzhou.aliyuncs.com/hotcode2e.zip>

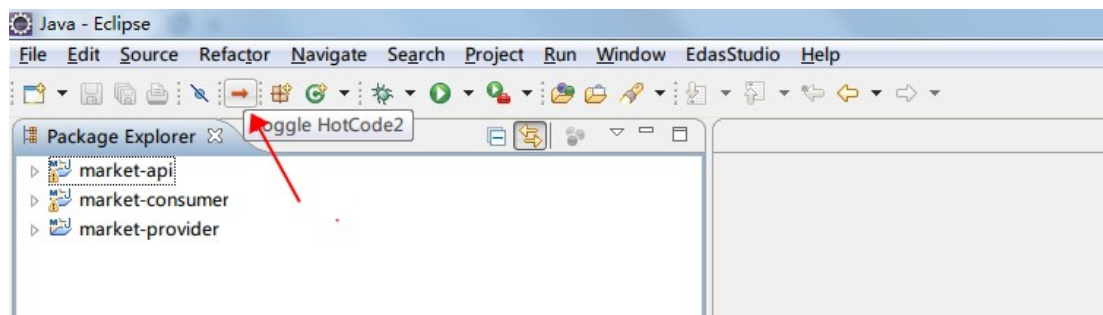
同 Tomcat4E 插件一样进行本地安装。

2. 下载 http://edas-public.oss-cn-hangzhou.aliyuncs.com/install_package/hotcode2/hotcode2.jar

与 1 中安装 hotcode 插件中配置关联，如下图：



3. 开启 Hotcode，点击工具栏中的 hotcode 图标，如下图：



启动应用之后，就可以在控制台的日志中末尾看到如下日志输出：

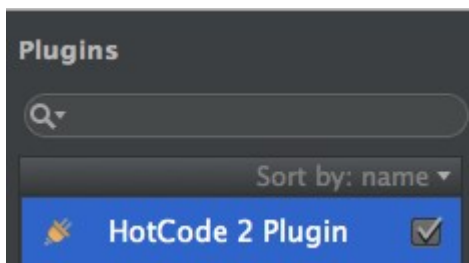

```
market-provider [1] [WinHttpWebApp] C:\java_win_1_6_0_29\bin\javaw.exe (2015年07月17日 上午9:52:42)
2015-8-17 9:52:41 org.springframework.context.support.AbstractApplicationContext prepareRefresh
信息: Refreshing org.springframework.web.context.support.XmlWebApplicationContext@bc36ff: display name [Root WebApplicationContext]; startup
2015-8-17 9:52:42 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
信息: Loading XML bean definitions from class path resource [hsf-provider-beans.xml]
2015-8-17 9:52:42 org.springframework.context.support.AbstractApplicationContext obtainFreshBeanFactory
信息: Bean factory for application context [org.springframework.web.context.support.XmlWebApplicationContext@bc36ff]: org.springframework.be
2015-8-17 9:52:42 org.springframework.beans.factory.support.DefaultListableBeanFactory preInstantiateSingletons
信息: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@266b44: defining beans [item,samp
JM.Log:INFO Init JM logger with Slf4jLoggerFactory
JM.Log:INFO Init JM logger with Slf4jLoggerFactory
JM.Log:INFO Log root path: C:\Users\yinshi.nc\logs\
JM.Log:INFO Set diamond-client log path: C:\Users\yinshi.nc\logs\diamond-client
2015-8-17 9:52:42 com.alibaba.alimonitor.jmonitor.client.JmonitorClient$CheckTask run
严重: Connection refused: connect
JM.Log:INFO Log root path: C:\Users\yinshi.nc\logs\
JM.Log:INFO Set hsf log path: C:\Users\yinshi.nc\logs\hsf
JM.Log:INFO Log root path: C:\Users\yinshi.nc\logs\
JM.Log:INFO Init JM logger with Slf4jLoggerFactory
JM.Log:INFO Set configclient log path: C:\Users\yinshi.nc\logs\configclient
2015-8-17 9:52:43 org.springframework.web.context.ContextLoader initWebApplicationContext
信息: Root WebApplicationContext: initialization completed in 1893 ms
2015-8-17 9:52:43 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["http-bio-8080"]
===== HotCode2 =====
Hello, HotCode2 (Ver: 2.0.1.20150731-1533) !!!
Start JVM with HotCode2 on Java 1.6.0_29-b11 @ Tomcat-Windows 7-6.1
HotCode2 Path: /W:/edas2.X/install-package/hotcode2/hotcode2.jar
Web Container: Tomcat
Monitored Resource Paths (Mapping class & non-xml resource) :
WEB-INF\classes -----> W:\flower-shop\market-provider\target\classes
webroot -----> W:\flower-shop\market-provider\src\main\webapp
Enabled Plugins: [ibatis_plugin, spring_plugin, webx3_plugin, webx2_plugin]
===== HotCode2 =====
```

1.2.2 IntelliJ Idea 安装 Hotcode 插件

1. 在以下地址下载

http://edas-public.oss-cn-hangzhou.aliyuncs.com/install_package%2Fhotcode2%2Fhotcode2-idea-plugin.jar

2. 通过 **Install plugin from disk**, 选择前面本地保存的 hotcode 目录, 完成 Hotcode 的安装

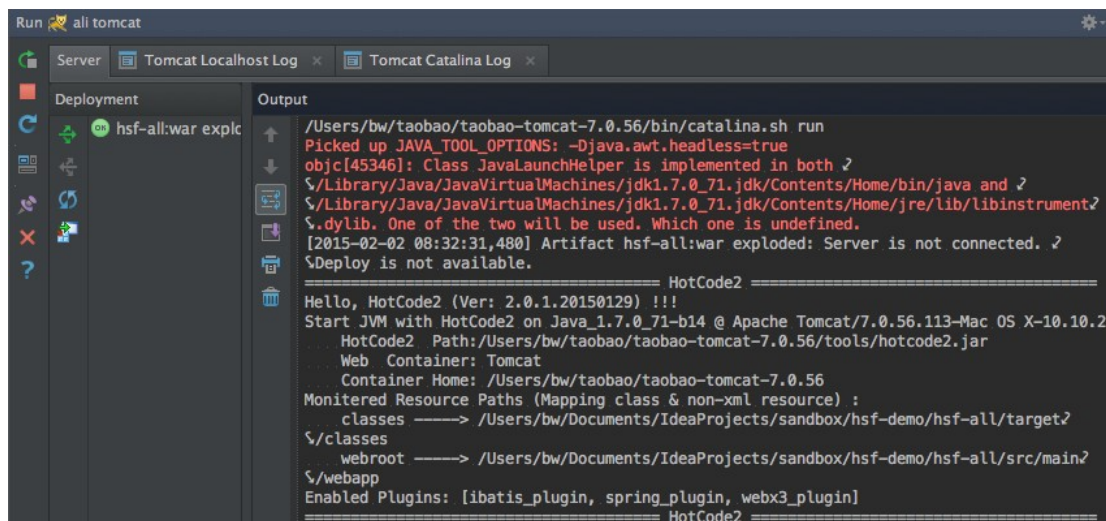


3. 安装完成之后, 在工具栏点击 Hotcode 插件的图标即可针对当前工程激活 Hotcode。

说明: 在安装插件的情况下, 应用不再需要配置 **-Dhotcode.base**, 也就是说, 在 hotcode 插件激活的时候, 唯一需要指定的是 **taobao-hsf.sar** 的位置 (**-Dpandora.location**)。



4. 确认 Hotcode 和 Pandora 启动, 应用部署成功。



2. 开发环境搭建

完成以上工具准备后，您可以通过安装配置中心（地址服务器、Diamond、ConfigServer）来快速搭建 EDAS 本地开发环境。

2.1 启动 EDAS 配置中心

请按照以下步骤启动 EDAS 配置中心：

1. 确认环境是否达到要求：

- 1) 正确配置环境变量 `JAVA_HOME`，指向一个 1.6 或 1.6 以上版本的 JDK。
- 2) 确认 8080 和 9600 端口未被使用。

注意事项：

由于启动 EDAS 配置中心将会占用此台机器的 8080 和 9600 端口，因此推荐您找一台专门的机器启动 EDAS 配置中心，比如某台测试机器。

2. 启动 EDAS 配置中心：

- 1) 从以下地址下载 EDAS 配置中心安装包并解压：

[edas-config-center.zip](#)

- 2) 进入 `edas-config-center` 目录启动配置中心：

- Windows 系列操作系统请双击 `startup.bat`。
- Unix 系列操作系统请在当前目录下执行下面的命令。

```
sh startup.sh
```

2.2 使用 EDAS 配置中心

在本地 DNS（hosts 文件）中，将 `jmenv.tbsite.net` 指向启动了 EDAS 配置中心的机器 IP。

hosts 文件的路径如下：

- Windows 系列操作系统：C:\Windows\System32\drivers\etc\hosts

- Unix 系列操作系统: /etc/hosts

示例:

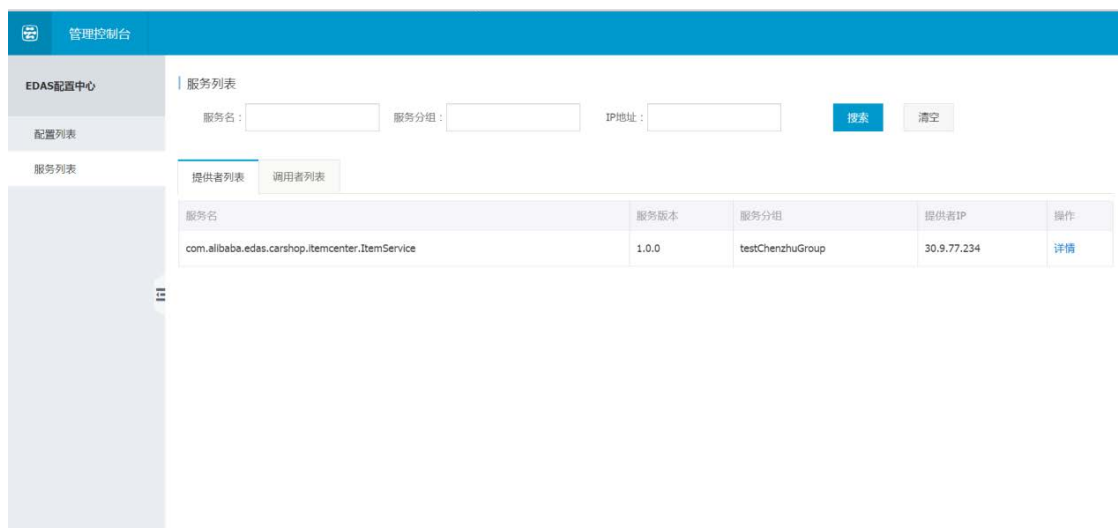
如果您在 IP 为 192.168.1.100 的机器上面启动了 EDAS 配置中心, 则所有开发者只需要在机器的 hosts 文件里加入如下一行即可:

```
192.168.1.100 jmenv.tbsite.net
```

2.3 通过 EDAS 配置中心查询某个应用提供或调用的服务

以下过程假设您在一台 IP 为 192.168.1.101 的机器上启动了 EDAS 配置中心。

1. 进入 `http://192.168.1.101:8080/`
2. 在左侧菜单栏点击**服务列表**, 会显示以下页面。您可以根据服务名、服务组名或者 IP 地址进行模糊搜索, 用来查询服务提供者以及服务调用者。



常见查询案例: 在提供者列表页:

- 在搜索条件里输入 IP 地址, 点击“搜索”即可查询该 IP 地址的机器提供了哪些服务。
- 在搜索条件里输入服务名或服务分组, 即可查询哪些 IP 地址提供了这个服务。

在调用者列表页:

- 在搜索条件里输入 IP 地址, 点击“搜索”即可查询该 IP 地址的机器调用了哪些服务。
- 在搜索条件里输入服务名或服务分组, 即可查询哪些 IP 地址调用了这个服务。

3. HSF 开发

3.1 HSF 简介

HSF(High Speed Service Framework)，高速服务框架，是阿里-主要采用的服务框架，其目的是作为桥梁联通不同的业务系统，解耦系统之间的实现依赖。

3.3.1RPC

远程过程调用(Remote Procedure Call)是一种通过网络从远程计算机程序上请求服务的协议，它的特点在于不需要了解底层网络技术。在 OSI 网络通信模型中，RPC 跨越了传输层和应用层。RPC 使得开发分布式应用更加容易。

3.3.2 ConfigServer(地址注册中心)

HSF 是一个 RPC 框架，服务端需要将地址发送到注册中心让客户端能够进行服务发现，客户端需要通过注册中心订阅一个服务的地址。服务与地址的对应关系是多对多的关系，一个服务可以由多个地址提供，一个地址可以提供多种服务。当一个服务有了新地址（机器）或者减少了地址（机器）时，注册中心会通知这个服务的订阅方将地址增加或者减少，这个注册中心就是 Configserver，它负责存储地址信息以及地址变更的推送。具体信息请参考本文第二章[开发环境搭建](#)。

3.3.3 Diamond(配置中心)

HSF 持久化的配置中心是 Diamond。HSF 提供软负载服务，其中的路由规则是在 Diamond 上进行配置，然后推送到客户端进行解析，第二章节中开发环境搭建已经包含 Diamond。

3.3.4 Pandora

Pandora 是 HSF 生存的容器，由 pandora 来管理整个 HSF 的生命周期和二方包的隔离。

3.2HSF 标签清单

3.3.3.1 Consumer 配置

属性	描述
interface	interface 必须配置[String]，为需要调用的服务的接口。
version	version 为可选配置[String]，含义为需要调用的服务的

	版本，默认为 1.0.0。
group	group 为可选配置[String]，含义为需要调用的服务所在的组，默认为 HSF，建议配置。
methodSpecials	methodSpecials 为可选配置，含义为为方法单独配置超时(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 hsf.runmode=0 的开发环境中，在运行环境下，此属性将无效，而是采用配置中心推送回来的目标服务地址信息。
connectionNum	connectionNum 为可选配置，含义为支持设置连接到 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)，超时设置优先级由高到低是：客户端 MethodSpecial，客户端接口级别，服务端 MethodSpecial，服务端接口级别。
asyncallMethods	asyncallMethods 为可选配置[List]，含义为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	配置这个，在服务订阅的时候，会在指定的时间内，等待地址推送过来，避免地址为过来后，调用该服务出现找不到地址的情况，超过指定时间未等到地址，将继续初始化完成。

标签配置示例：

```
<hsf:consumer id="service" interface="com.taobao.edas.service.SimpleService"
  version="1.1.0" group="test1" clientTimeout="3000"
  target="10.1.6.57:12200?_TIMEOUT=1000" maxWaitTimeForCsAddress="5000">
  <hsf:methodSpecials>
    <hsf:methodSpecial name="sum" timeout="2000"/>
  </hsf:methodSpecials>
</hsf:consumer>
```

3.3.3.2 Provider 配置

属性	描述
interface	interface 必须配置[String]，为服务对外提供的接口
version	version 为可选配置[String]，含义为服务的版本，默认为 1.0.0
group	serviceGroup 为可选配置[String]，含义为服务所属的组别，以便按组别来管理服务的配置，默认为 HSF
clientTimeout	该配置对接口中的所有方法生效，但是如果客户端通过 MethodSpecial 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	serializeType 为可选配置[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，是从公用线程池这个大蛋糕里切一块下来
maxPoolSize	单独针对这个服务设置线程池，是从公用线程池这个大蛋糕里切一块下来
enableTXC	开启分布式事务 Txc
ref	ref 必须配置[ref]，为需要发布为 HSF 服务的 spring bean id
methodSpecials	methodSpecials 为可选配置，用于为方法单独配置超时(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

标签配置示例：

```
<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" group="test1" clientTimeout="3000" enableTXC="true"
    serializeType="hessian">
    <hsf:methodSpecials>
      <hsf:methodSpecial name="sum" timeout="2000" />
    </hsf:methodSpecials>
  </hsf:provider>
```

3.3.3.3 HSF jvm 参数配置

-Dhsf.server.port

指定 HSF 的启动服务绑定端口，默认为 12200

-Dhsf.serializer

指定 HSF 的序列化方式，默认值为 hessian

-DdefaultHsfClientTimeout

指定 HSF 的客户端超时时间 ms，默认为 3000

-Dhsf.server.max.poolsize

指定 HSF 的服务端最大线程池大小，默认值为 600

-Dhsf.server.min.poolsize

指定 HSF 的服务端最小线程池大小。默认值为 50

3.3 HSF 基础开发

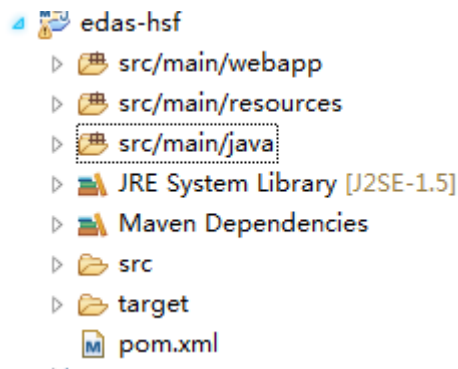
参考下面的 demo 编写示例，来快速的开发一个基础版的 HSF 应用

3.3.1 创建 Web 项目

通过上面第一、二章节我们已经在开发工具中配置好对应的插件，并且搭建了本地服务注册配置中心，下面我们以一个 demo 展开介绍 Hsf 开发的相关细节。

以 eclipse 为例创建一个 maven web 项目。

请点击 *File -> New -> Project -> Maven Project -> maven-archetype-webapp*，在弹出的界面输入 groupId、artifactId，然后连续点击 Next，完成项目创建。项目目录结构如图：



3.3.2 添加 Maven 依赖

在项目 *pom.xml* 中添加如下依赖：

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>3.1.1.RELEASE</version>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.0.1</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>hsf.schema</artifactId>
    <version>edas1.0.1</version>
</dependency>
```

3.3.3 编写发布 HSF 的服务

创建需要发布的服务接口：*com.alibaba.edas.SampleService*

编写实现类: `com.alibaba.edas.impl.SampleServiceImpl`

```
public interface SampleService {
    String echo(String str);
}

public class SampleServiceImpl implements SampleService {
    @Override public String echo(String str)
    { return str;
    }
}
```

3.3.4 发布服务配置文件

请按照以下步骤配置发布者的服务配置文件。

1. 在 `web.xml` 中配置 `spring` 的监听器:

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:config/applicationContext.xml</param-value>
</context-param>
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-
class>
</listener>
```

2. 在 `resources` 目录下面添加 `spring` 配置文件: `config/applicationContext.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    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">
    <import resource="classpath:config/providers-spring.xml"/>
</beans>
```

3. 在 `resources` 目录下面添加发布者的配置文件: `config/providers-spring.xml`

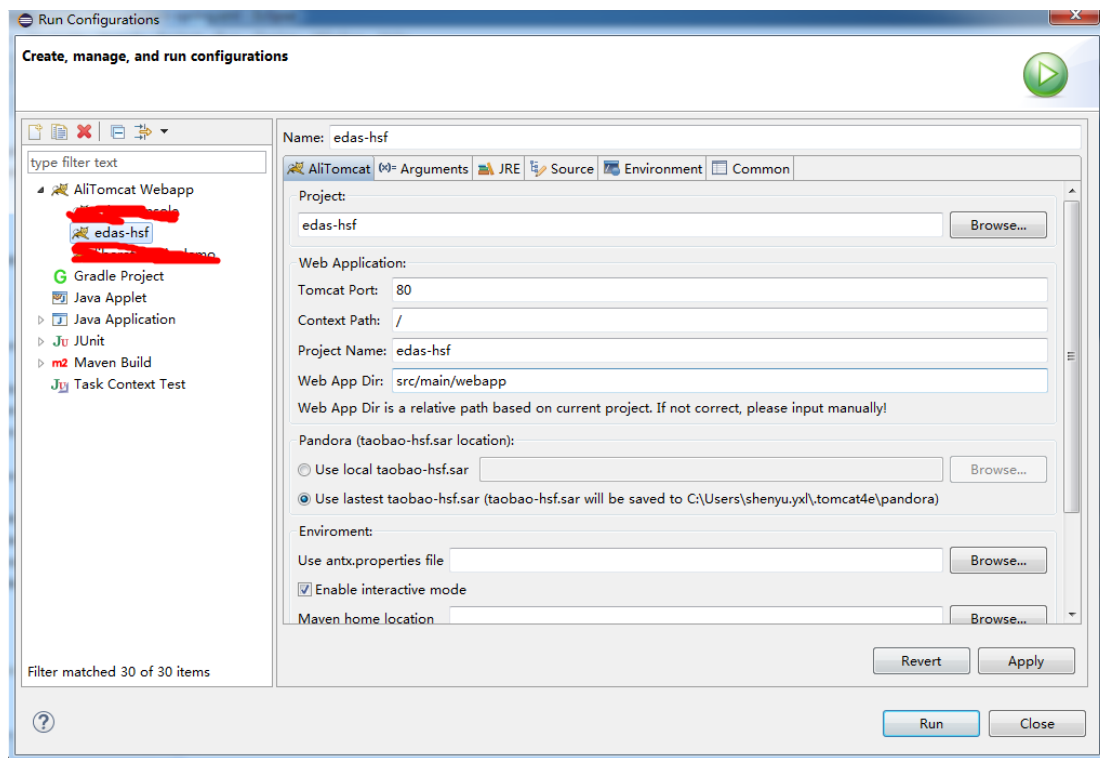
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:hsf="http://www.taobao.com/hsf"
    xmlns="http://www.springframework.org/schema/beans"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.taobao.com/hsf
http://www.taobao.com/hsf/hsf.xsd" default-autowire="byName">
    <bean id="target" class="com.alibaba.edas.impl.SampleServiceImpl"/>
<hsf:provider id="sampleServiceProvider" interface="com.alibaba.edas.SampleService"
```



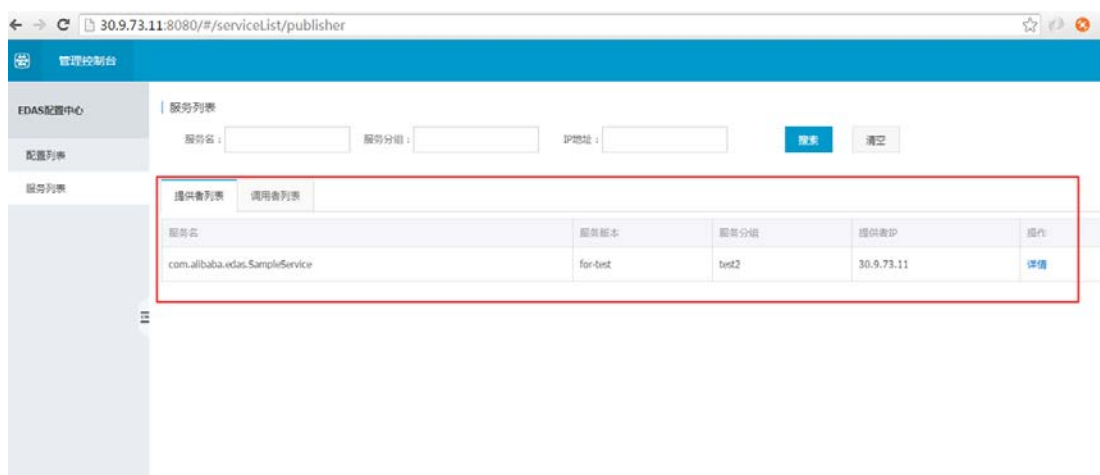
```
ref="target" version="for-test" group="your-namespace">
</hsf:provider>
</beans>
```

到此发布者就编写好了。

4. 右键点击创建的项目，点击 **Run As**，选择 **Run configuration**，在弹出页面选择 **AliTomcat Webapp**，右键 **new** 新建，如下图：



5. 填写好相关配置，点击 **Run**，启动应用，此时通过第二章节建的配置中心可以看到发布的服务，如下图：



3.3.5 消费服务配置文件

1. 在配置文件 `config/applicationContext.xml` 添加消费者配置:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       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">
    <import resource="classpath:config/providers-spring.xml"/>
    <import resource="classpath:config/consumers-spring.xml"/>
</beans>
```

2. 在 `resources` 目录下面添加消费者的配置文件 `consumers-spring.xml` 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.taobao.com/hsf
http://www.taobao.com/hsf/hsf.xsd" default-autowire="byName">
    <hsf:consumer id="sampleService" interface="com.alibaba.edas.SampleService"
                  version="for-test" group="your-namespace">
    </hsf:consumer>
</beans>
```

3. 已经完成了消费者的定义，下面创建 `servlet` 来调用测试代码进行测试:

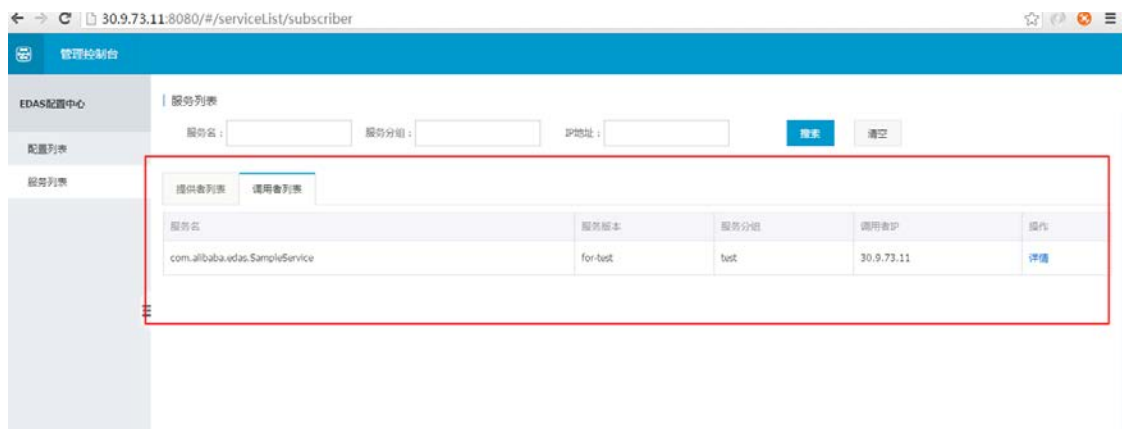
```
public class HsfServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException{
        ApplicationContext
        ctx=WebApplicationContextUtils.getWebApplicationContext(req.getServletContext());
        SampleService sampleService = (SampleService)ctx.getBean("sampleService");
        resp.getWriter().println(Long.toString(System.currentTimeMillis()));
    }
}
```

4. 在 `web.xml` 中添加如下内容:

```
<servlet>
    <servlet-name>hsf</servlet-name>
    <servlet-class>com.alibaba.edas.HsfServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>hsf</servlet-name>
```

```
<url-pattern>/hsf.htm</url-pattern>
</servlet-mapping>
```

5. 启动应用，通过配置中心可以查看消费服务列表，如下图：



通过前面 3.3.5 编写的消费 servlet，可以调用到服务提供者的服务。

3.3.6 Demo 下载地址

通过如下地址下载：

http://edas-public.oss-cn-hangzhou.aliyuncs.com/install_package%2Fedas-app-demo%2Fedas-app-demo.zip

将下载下来的压缩包解开后，可以看到 itemcenter-api，itemcenter 和 detail 三个 Maven 工程。其中 itemcenter-api 工程提供接口定义，detail 工程是消费者应用，Itemcenter 工程是服务提供者应用。

3.4 HSF 高级开发

敬请期待

4. FAQ

4.1 EDAS 日志说明

开发中遇到的很多问题，我们都可以通过查看相关日记，找到问题所在，进而解决问题。下面表格是 EDAS 相关日记路径汇总。

日志文件名	日志文件含义
/home/admin/taobao-tomcat-producti on-xxxx/logs/catalina.out	最重要的日志, 可以看到应用和 tomcat 应用服务器的异常, 这是开发最应该关注的日志
/home/admin/taobao-tomcat-producti	如果通过 catalina.out 看到的错误很模糊或者没

on-xxxx/logs/localhost.log.xxx	有错误，可以结合 localhost 来一起看，保证应用正常启动，再看下面的 log 继续排查问题
/home/admin/configclient/logs/config.client.log	可以通过此日志查看服务发布订阅是否成功，Register-ok Publish-ok 等关键字
/home/admin/logs/hsf/hsf.log	HSF 服务的日志，有 HSF 服务调用过程的一些详细信息，如果 HSF 调用中出现异常，可以看看这个日志

4.2 Tomcat FAQ

作为应用的容器，简单的排查问题的思路就是：

遇到问题先查看/home/admin/taobao-tomcat-production-xxxx/logs/catalina.out，再查看/home/admin/taobao-tomcat-production-xxxx/logs/localhost.log.xxx，找到 tomcat 相关的第一个错误。先解决第一个错误，重启观察，看是否还有其他错误。

4.2.1 修改应用访问根路径

问题：通过修改 server.xml 来配置路径不生效，导致启动出现 Class not Def

解决方案：

Tomcat 启动应用时默认使用 war 包名字作为 ContextPath。访问应用时必须带上 ContextPath 路径才能访问，ali-tomcat 兼容 jboss jboss-web.xml 配置，可通过 jboss-web.xml 修改 ContextPath。添加 jboss-web.xml 修改应用 ContextPath 为 /，即可通过根路径访问应用（不需要带应用名）。

在应用 WEB-INF 目录下添加 jboss-web.xml 文件，文件完整内容如下：

```
<jboss-web>

    <context-root>/</context-root>

</jboss-web>
```

4.2.2 Tomcat jvm 内存配置

检查内存相关的配置如-Xmx, MaxDirectMemory, MaxPermSize 这些加在一起如果接近或者超过虚拟机的内存(可通过 cat /proc/meminfo | grep MemTotal 进行查看)，就很有可能发生 oom-killer。

4.2.3 Tomcat 在启动过程中卡住

- 1) 如果进程存在，那么说明 Tomcat 在部署过程中卡在某个地方了，首先通过 jstack 打印线程堆栈并保存到某个文件中。
- 2) `sudo -u admin /opt/taobao/java/bin/jstack $PID > /var/tmp/$PID.jstack`
- 3) `vim /var/tmp/$PID.jstack`

然后打开该文件，搜索 `localhost-startStop-1` 查看该线程的堆栈，通常会看到该线程 (Tomcat 的部署线程) 在处于等待状态，或者是出现死锁，根据情况对应用进行修改。

4.3 HSF FAQ

4.3.1 HSF-0001

问题描述

客户端报错：在 `/home/admin/logs/hsf` 查看 `hsf.log`

```
04 2016-09-18 14:54:36.694 WARN [http-bio-8080-exec-3:it.hsf] [] [] RPC协议调用服务[com.alibaba.edas.test.EdasDemoService:1.0.0]的[echo]方法时出现错误:
所调用的服务目标地址为: [null]
TraceId=0a650d9014635544756041007d39d1,
, RpcId=0.1
com.taobao.hsf.exception.HSFServiceAddressNotFoundException: HSFServiceAddressNotFoundException-
    at com.taobao.hsf.remoting.service.RPCProtocolTemplateComponent.invoke(RPCProtocolTemplateComponent.java:281) [hsf.app.spring-edas.1.0.4.jar:na]
    at com.taobao.hsf.remoting.service.RPCProtocolTemplateComponent.invokeWithMethodObject(RPCProtocolTemplateComponent.java:196) [hsf.app.spring-edas.1.0.4.jar:na]
    at com.taobao.hsf.process.component.ProcessComponent$HSFServiceProxy.invoke(ProcessComponent.java:189) [hsf.app.spring-edas.1.0.4.jar:na]
    at com.taobao.hsf.process.component.ProcessComponent$HSFServiceProxy.invoke(ProcessComponent.java:183) [hsf.app.spring-edas.1.0.4.jar:na]
    at com.taobao.csp.sentinel.entrypoint.proxy.TraceInvocationHandler.invoke(TraceInvocationHandler.java:38) [sentinel-2.0.6.4.2-pandora.jar:na]
    at $Proxy36.echo(Unknown Source) [na:na]
    at com.alibaba.edas.test.portal.InvokeEcho.doGet(InvokeEcho.java:30) [InvokeEcho.class:na]
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:620) [servlet-api.jar:na]
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:727) [servlet-api.jar:na]
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:231) [catalina.jar:5.0.50-163]
```

HSFServiceAddressNotFoundException-[HSF-Consumer] 未找到需要调用的服务目标地址 描述信息：需要调用的目标服务为：xxxx 组别为：xxxx

解决方案

1、服务不存在：

在 edas 控制台-应用-服务列表查看对应的服务是否已经发布成功 2、

配置不对：

检查发布者和消费者配置的服务名字、版本号、组别，这 3 个要一致（注意大小写也要一样，同时要注意前后不能有空格）

4.3.2 HSF-0002

问题描述 消费端报超时

(HSFTimeoutException)

解决方案 目前出现这个错误的很大一部分原因都是因为服务提供方接口内部逻辑报错，导致消费端 报超时错误，可以先去检查一下服务提供方的业务 log 是否报错。 1、服务端跑的慢，在服务端的 `hsf.log` 里找业务执行超时的日志。

2、两边可能有 GC 。 检查服务端和客户端 gc 日志，耗时很长的 gc，会导致超时。

- 3、服务端有序列化错误。看一下服务端的 hsf.log，里面一般有序列化错误。
- 4、上面三个都排除了的话，可以让 PE（运维人员）看一下机器的网络是否健康。

4.3.3 HSF-0003

问题描述

[HSF-Provider] 未找到需要调用的方法

解决方案

1.服务端确实没有提供这个方法，在 edas 控制台-应用-服务列表查看对应的服务, 查看服务详情。2.服务端应用新增的服务，但有两台机器，一台部署了新服务，加了方法。一台还是老的应用，调用走到了老的应用机器上，联系服务方排查。 3.服务方和消费方接口没统一，比如服务方的参数列表里是 `java.lang.Double`，而消费方调用时直接传了基本类型 `double`，也会造成这个错误。

4.3.4 HSF-0027

问题描述

hsf.log 日志出现：[HSF-Provider] HSF thread pool isfull.

解决方案 分配给服务端业务执行的线程池已经达到最大值，并且没有空闲线程，拒绝执行了一些任务。出现这个异常时，HSF 默认会 dump 出一个文件：HSF_JStack.log（这个日志文件一般在服务提供方的 hsf.log 同级目录下），查看这个文件可以看到此时的线程堆栈信息，其中名称为 HSFBizProcessor-xxx 的是服务端的业务线程，分析大部分线程此刻正在等待什么资源，从而分析出哪个地方是瓶颈。

HSF 默认的最大线程数是 600，如果确实要增大这个数，则使用 `-Dhsf.server.max.poolsize=xxx` 参数进行修改。

4.3.5 HSF-0031

问题描述

[HSF-Provider] 执行 HSF 服务 xxx 的方法 xxx 耗时 xxxms，接近超时时间

解决方案 服务端：

服务端会在实际耗时大于超时时间与 100ms 之差时，打印这个日志，所以 1、如果超时时间本来就很短，比如小于 100ms，则每次调用都会打印这个日志，这时可忽略这个日志。 2、如果超时时间已经比较大了，还打印这个日志，则说明业务执行慢，需要分析业务执行的性能瓶颈。

客户端：

超时一般报在客户端 `HSFTimeoutException`，因为 HSF 的整个服务调用过程是点对点的，如果客户端发现超时，一般是服务端执行慢了。

异常里会打服务端的 ip，第一步应该是去看服务端的 `hsf.log` 日志，80% 的原因都是服务端跑的慢。

服务端执行非常快，但当客户端 Load 很高，负载压力很大的时候，会因为客户端请求发不出去、响应卡在 tcp buffer 等问题，造成超时。

```
#####timeout,so give up send response to client
```