ALBus架构设计

|  |  |
| --- | --- |
| 版本 | 1.0.0 |
| 发布时间 | 2014/9/16 |
| 作者 | 张欣 |

# 目录

[ALBus架构设计 1](#_Toc406426753)

[目录 2](#_Toc406426754)

[简单开发 3](#_Toc406426755)

[添加依赖 3](#_Toc406426756)

[接口定义 3](#_Toc406426757)

[业务单元定义 3](#_Toc406426758)

[配置手册 4](#_Toc406426759)

[配置文件样例 4](#_Toc406426760)

[配置项 4](#_Toc406426761)

[调用配置 4](#_Toc406426762)

[部署和启动 6](#_Toc406426763)

[本地调用 6](#_Toc406426764)

[客户端启动 6](#_Toc406426765)

[服务端 6](#_Toc406426766)

# 简单开发

## 添加依赖

|  |
| --- |
| <dependency>  <groupId>net.butfly.bus</groupId>  <artifactId>albus-plus</artifactId>  <version>2.0.1-SNAPSHOT</version>  </dependency> |

## 接口定义

最简单的albus接口由一个带@TX标注的接口方法定义，该接口应该继承Facade接口：

|  |
| --- |
| **public** **interface** UserFacade **extends** Facade {  @TX("Login")  Map<String, ?> doApplicantLogin(String func, ApplicantBasic applicantBasic, ApplicantExtend applicantExtend) **throws** Exception; |

## 业务单元定义

Albus需要接受一个业务单元（bean）组合。一般有两种模式：

1. BeanFactory模式：自行创建所有业务单元实例。
   1. 创建一个AbstractBeanFactoryInvoker的继承类，在继承类中实现getBeanList()方法。

|  |
| --- |
| **public** **class** BeanFactoryInvoker **extends** AbstractBeanFactoryInvoker {  @Override  **public** Object[] getBeanList() {  **return** **new** Object[] { **new** CometFacadeImpl() };  }  } |

* 1. 将此Invoker配置到albus中

|  |
| --- |
| <invoker type=*"net.butfly.bus.demo.invoker.BeanFactoryInvoker"* id=*"albus-test-comet"* tx=*"TST\_CMT-\*"* /> |

1. Spring容器模式：由Spring托管所有业务单元实例。

|  |
| --- |
| <invoker id=*"albus-test-comet"* tx=*"API-PERSON\_\*"* type=*"net.butfly.bus.invoker.SpringInvoker"*>  <files>com/hzcominfo/appstore/services/person/spring/beans.xml</files>  <lazy>false</lazy>  </invoker> |

# 配置手册

Albus从配置文件中读取配置。默认配置文件为classpath下的bus.xml。建议分离客户端和服务端配置为bus-client.xml/bus-server.xml。

## 配置文件样例

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <bus id=*"bus-person"* side=*"server"*>  <filter title=*"async-filter"* class=*"net.butfly.bus.filter.AsyncFilter"*  enabled=*"false"* />  <filter title=*"logger-filter"* class=*"net.butfly.bus.filter.LoggerFilter"*  enabled=*"true"* />  <filter title=*"exception-filter"* class=*"net.butfly.bus.filter.ExceptionHandleFilter"*  enabled=*"true"* />  <router type=*"net.butfly.bus.policy.SimpleRouter"* />  <invoker id=*"albus-test-comet"* tx=*"API-PERSON\_\*"* type=*"net.butfly.bus.invoker.SpringInvoker"*>  <files>com/hzcominfo/appstore/services/person/spring/beans.xml</files>  <lazy>false</lazy>  </invoker>  </bus> |

## 配置项

|  |  |  |  |
| --- | --- | --- | --- |
| 配置节点 | 配置节点说明 | 参数 | 参数说明 |
| bus | 配置文件根节点 | id | 必须唯一 |
| side | 可选，server/client |
| filter | 过滤器 | title | 必须唯一 |
| class | 过滤器实现类 |
| enable | 过滤器是否启用 |
| invoker | 总线调用定义 | id | 必须唯一 |
| class | 调用实现类 |
| tx | 由该调用器处理的TX码，逗号分隔，接受通配符（\*） |

## 调用配置

|  |  |  |  |
| --- | --- | --- | --- |
| 调用实现 | 调用实现说明 | 参数配置 | 参数说明 |
| BeanFactoryInvoker | 简单对象集合调用 | 无需参数配置 | |
| SpringInvoker | Spring容器调用 | file | Spring容器定义文件（基于classpath） |
| lazy | 是否延迟加载spring上下文 |
| WebServiceInvoker | WEB服务调用 | path | WEB服务入口 |

# 部署和启动

## 本地调用

当albus客户端（接口调用者）配置中直接使用本地调用器（如BeanFactoryInvoker/SpringInvoker），albus实现本地接口调用模式，亦即，接口客户端和接口服务端在同一应用中。一般该模式用于开发调试或接口抽象。

## 客户端启动

当albus客户端（接口调用者）配置中使用远程调用器（如WebServiceInvoker），albus实现C/S调用模式。调用被发送到指定的服务端（接口实现者）。

## 服务端

服务端（接口实现者）可以在配置中使用本地调用器或远程调用器，前者在服务端实现了具体业务接口，后者实现了业务调用转发。

服务端可以使用标准Servlet容器（如Tomcat、Jetty、WebLogic等）。在容器中定义WebServiceServlet以开放接口。WebServiceServlet应带有参数config-file，该参数指定了albus配置文件入口（基于classpath）。

服务端也可以使用albus提供的嵌入容器，直接从命令行启动albus：

1. 启动类（main函数所在类）：net.butfly.bus.deploy.JettyStarter
2. 参数：albus配置文件
3. 选项：

|  |  |
| --- | --- |
| 选项 | 说明 |
| -h, --heko | 打印帮助 |
| -s, --secure | 使用https（当前无效） |
| -k, --fork | 是否作为服务线程（若添加了该参数，服务提供线程作为当前线程的子线程运行，也就是当前应用结束时服务结束） |

1. 系统变量：

|  |  |  |
| --- | --- | --- |
| 系统变量名 | 说明 | 默认值 |
| bus.port | 服务端口 | 19080 |
| bus.port.secure | 是否使用SSL(当前无效) | 19443 |
| bus.threadpool.size | 线程池大小 | -1（不使用线程池） |
| bus.server.context | 服务URL上下文（WEB应用名） | /bus/\* |
| bus.jndi | JNDI资源定义文件（格式同tomcat的context.xml） | 无jndi资源 |
| bus.server.base | WEB静态文件根目录 | 无静态WEB资源 |
| bus.server.class | Albus实现类 | net.butfly.bus.Bus |
| bus.servlet.class | Albus接口Servlet类 | net.butfly.bus.deploy.WebServiceServlet |