**GAEA文档**

**58同城-技术中心-架构部**

**2012年2月**

目 录

[前言 4](#_Toc334538394)

[第1章 Gaea简介 6](#_Toc334538395)

[1.1 什么是Gaea 6](#_Toc334538396)

[1.2 Gaea组成 6](#_Toc334538397)

[1.3 特点 6](#_Toc334538398)

[1.4 应用场景 7](#_Toc334538399)

[1.5 架构 8](#_Toc334538400)

[1.6 发展历史 10](#_Toc334538401)

[1.7 总结 10](#_Toc334538402)

[第2章 Gaea安装与配置 11](#_Toc334538403)

[2.1 在windows上安装Gaea 11](#_Toc334538404)

[2.1.1 检查JDK安装 11](#_Toc334538405)

[2.1.2 下载Gaea 11](#_Toc334538406)

[2.1.3 本地安装 11](#_Toc334538407)

[2.2 在基于UNIX的系统上安装Gaea 12](#_Toc334538408)

[2.3 目录结构 12](#_Toc334538409)

[2.4 小结 13](#_Toc334538410)

[第3章 Gaea使用入门 14](#_Toc334538411)

[3.1 编写服务端 14](#_Toc334538412)

[3.2 编写客户端 19](#_Toc334538413)

[3.2.1 Java客户端 19](#_Toc334538414)

[3.2.2 .net客户端 21](#_Toc334538415)

[3.2.3 C++客户端 23](#_Toc334538416)

[3.2.4 客户端配置 23](#_Toc334538417)

[3.3 服务部署 24](#_Toc334538418)

[3.4 服务运行 25](#_Toc334538419)

[3.5 小结 27](#_Toc334538420)

[第4章 Gaea高级使用 28](#_Toc334538421)

[4.1 服务端 28](#_Toc334538422)

[4.1.1 流程图 28](#_Toc334538423)

[4.1.2 参数说明 28](#_Toc334538424)

[4.2 客户端 30](#_Toc334538425)

[4.2.1 流程图 30](#_Toc334538426)

[4.2.2 参数说明 30](#_Toc334538427)

[4.3 协议 32](#_Toc334538428)

[4.4 序列化 33](#_Toc334538429)

[4.4.1 Object序列化 33](#_Toc334538430)

[4.4.2 Array/List/Map序列化 33](#_Toc334538431)

[4.4.3 基本类型 34](#_Toc334538432)

[4.4.4 String 34](#_Toc334538433)

[4.5 Gaea服务监测 35](#_Toc334538434)

# 前言

随着[www.58.com](http://www.58.com)的知名度越来越高访问量越来越大，目前日访问量已经近3亿，日发帖量近200万，其中用户系统负载已达到5k QPS，信息系统更是达到了20k QPS，对系统的性能和可靠性要求越来越高。随着公司的发展，新产品不断的被研发，这也使得系统越来越多，系统间的相互调用、通讯越来越复杂。开发人员也从原来的几十人增加到了几百人,对开发人员的开发规范也提出了更高的要求。

如何让一般的程序员能够开发出优秀程序员一样高效，安全，稳定的系统？

如何实现即简单又高效的跨平台服务通讯？

如何统一web开发人员的开发规范，提高开发效率和代码重用、降低沟通成本？

如何保证各个系统的高可用性？

随着这些问题的出现，Gaea诞生了。Gaea主要包括4部门：客户端、协议、序列化、服务端，并支持c++、c#、java等主流客户端平台。

**本书使用的约定**

本书主要针对Gaea使用进行讲解，包括c++、c#、java平台。

**JAVA版依赖**

com.bj58.spat.gaea.client.1.0.0.jar

com.bj58.spat.gaea.protocol.1.0.0.jar

com.bj58.spat.gaea.serializer.1.0.0.jar

com.bj58.spat.gaea.server-1.0.0.jar

**C#依赖**

Com.Bj58.Spat.Gaea.client.dll、Com.Bj58.Spat.Gaea.Serializer.dll

**C++依赖**

gaea-c-client.a

**等宽字体**

用于示例代码和代码段、类、变量、方法名称、文本内使用的Java关键字、SQL命令、表名、列名及XML元素和标记等。

**等宽黑体**

某些代码示例中用于强调。

**如何联系我们**

邮件:code@58.com

地址:北京市朝阳区北苑路乙108号北美国际商务中心E座

# Gaea简介

## 什么是Gaea

Gaea是服务通讯框架(Service Communication Framework)支持跨平台具有高并发、高性能、高可靠性，并提供异步、多协议、事件驱动的中间层服务框架。

## Gaea组成

Gaea主要由4部分组成

□ Gaea Server：服务容器，用于宿主开发人员所开发的Gaea服务，接收和处理来自客户端的请求。

□ Gaea Client：多种平台客户端，调用者就像在调用本地接口一样方便，同时还提供了负载均衡，容错等机制等。

□ Gaea Serializer：提供了跨平台的二进制序列化解决方案。

□ Gaea Protocol：分为传输协议和数据协议．传输协议用于进行数据传输，数据协议用于请求和响应结果的内容数据。

## 特点

* 高性能：客户端和服务端特定的通讯模型，序列化组件对序列化和反序列化性能以及结果字节数组大小的严格控制，通讯协议的针对性设计等使得Gaea比.net平台的WCF，Java平台的EJB、RMI，跨平台的WebService等性能都要好。在追求性能的同时支持客户端的HA，容错机制为服务提供了良好的可靠性保证。
* 跨平台：由于特定系统的需要，会使用特定的语言或平台来开发，这么一来会存在多个平台的情况，比如有java、.net、c++等，想要实现这些系统的简单高效的跨平台调用是一个非常麻烦的事情，Gaea通过为不同平台提供不同的客户端来实现跨平台，目前已有java、.net、c&c++等平台的客户端。服务开发人员在开发服务时无需关心这些细节，所开发出来的服务就能很好的支持跨平台调用。
* 便捷开发：这是Gaea设计的主要目标之一，开发人员不需要了解网络通讯，不需要关心如何跨平台，不需要知道如何塔建一个服务集群以及怎么做HA，这些框都做了封装，需要做只是定义一个接口，实现接口里边的方法，再打几个注解就成了.使得一般的程序员能够开发出优秀程序员一样高效，安全，稳定的服务．对于服务调用者也是非常简单的，所有服务开发人员不需要了解的东西调用者同样也不需要了解，对调用者来说调用远程服务的方法就像是调用本地方法一样。
* 高扩展性：当服务有压力时可以随时加服务器，增加服务节点来分担压力，客户端的负载均衡模块可以根据服务结点的压力情况来调整各结点的权重并平衡压力。
* 安全性：Gaea 可以对调用者进行权限授权，不同的调用者只能调用对他授权的方法，这对有些对外暴露的服务又想对访问进行授权将会非常有用。

## 应用场景

* 对并发，稳定性要求都比较高的中间层服务  
  58主要的服务都宿主在Gaea中，核心服务包括：信息系统，用户系统对每秒能支撑的并发数和服务的稳定性要求都相当的高．58电商平台的秒杀系统也是以Gaea服务的形式对外提供服务，该系统高峰期瞬间并发数能达到好几万。
* 跨平台的RPC,RMI调用

58帮帮后端使用的是c++做开发，帮帮后端由于业务的需求需要调用很多58的基础服务，这些基础服务都是以Gaea服务的形式对外提供服务，帮帮使用Gaea的c++客户端可以很方便的调用这些基础服务，而不需要另外再开发一套。

* SOA实施  
  随着公司的发展很多业务系统会越来越复杂，系统间的耦合性越来越高，要解决这种现状可以对各个系统间共性公有的部分抽象成一个Gaea服务，各个业务系统通过调用该服务对外提供的接口来完成相关的业务逻辑．58所有大大小小的Gaea服务加起来已经有上百个，Gaea是58整体架构的重要组成部分。
* 服务安全级别要求比较高，需要对服务调用者进行授权的场景

58支付平台的Gaea服务对调用者需要做严格的控制，末经授权是不允许调用的。Gaea对请求授权粒度可以控制到方法很好的满足了这种场景的需求。

## 架构



* Gaea Client 负载均衡：该模块提供了对服务集群的均衡调度，坏节点自动移除检测等功能。通过请求队列长度值来平衡服务结点的权重，以及动态计算的请求超时时间(队列越长超时时间越短)，避免了服务被压死或调用者被堵死的情况。
* Gaea Client 动态代理：该模块负载创建客户端的服务代理拦截调用请求，将请求转换成Gaea协议通过网络通讯模块发送出去，同时将响应结果转换成方法调用的返回值返回给调用者。
* Gaea Client 网络通讯：负责发送请求协议包和接收来着服务端的响应，该模块通过SessionID和等侍窗口的机制实现了服务的异步调用。
* Gaea Serializer(跨平台二进制序列化)：该组件的目标是高效，跨平台，序列化出来结果的字节少，使用简单。不同语言平台遵循该结构进行序列化和反序列化，进而实现跨平台
* Gaea Protocol(通讯协议)：分为传输协议和数据协议。传输协议用于进行数据传输，数据协议用于请求和响应结果的内容数据。
* Gaea Server网络通讯：负责监听和接收来自客户端的请求，将收到请求包交给工作线程去处理。
* Gaea Server 热部署：负责服务启动时jar和class的加载，当jar有变更时实现服务不终断的热部署。
* Gaea Server 服务代理：在服务启动时生成服务的静态代理，每次请求都是先调用服务代理，由代理去直接调用真实服务，避免了反射调用能性能的影响。
* Gaea Server 权限控制：Gaea提供了两种机制

1. 通过授权码的机制，每一个授权码都对应着一个或几个方法，只有拥有该授权码的调用者才能调用这些方法。
2. 通过IP黑白名单来控制只有哪些IP能访问或哪些IP不能访问。

* Gaea Server 服务管理：通过telnet到指定的端口，可以对服务运行情况进行实时监控，例如：time:方法每次的执行时间，参数，调用者IP，count:每秒并发数，control:对服务进行控制等。有了这些功能可以对服务进行时时的监控，分析服务的运行情况，当服务出现问题时可以快速的定位是哪里出了问题。

## 发展历史

* Gaea\_V1版

跨平台序列化(支持json格式)

手动生成代理

* Gaea\_V2版

序列化升级

自动生成代理

客户端重构

telnet监测

* Gaea\_V3版

http协议支持

热部署

服务“容器化”

权限控制

服务重启客户端优雅切换

服务异步化

## 总结

本章主要介绍了什么是Gaea以及Gaea组成、特点、架构和主要使用场景，并对Gaea的发展历史进行了简单的介绍，下章开始逐步介绍如何安装Gaea及如何使用。

# Gaea安装与配置

第1章介绍了Gaea是什么，以及为什么要使用Gaea及应用场景，我们将从本章开始时间接触Gaea。本章首先介绍如何在主流的操作系统下安装Gaea，并详细解释Gaea安装文件。

## 在windows上安装Gaea

### 检查JDK安装

在安装Gaea之前，首先要确认你已经正确安装了JDK。Gaea可以运行在JDK1.4及以上的版本。本书的所有样例都基于JDK5及以上版本。打开Windows的命令行，运行如下命令来检查JDK安装:

echo %JAVA\_HOME%

java –version



上述命令首先检查环境变量JAVA\_HOME是否指向了正确的JDK目录，接着尝试运行java命令。如果Windows无法执行java命令，或无法找到JAVA\_HOME环境变量，请检查jdk是否正确安装，或环境变量是否设置正确。

### 下载Gaea

请访问Gaea下载页面,其中包含针对不同平台的各种Gaea下载文件。

### 本地安装

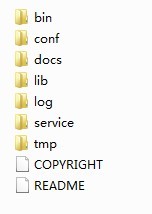
将安装文件解压到指定目录。

## 在基于UNIX的系统上安装Gaea

Gaea是跨平台的，它可以在任何一种主流的操作系统上运行。本书将介绍如何在基于UNIX的系统(包括linux、Mac OS以及FreeBSD)上安装Gaea。

首先，与Windos上安装Gaea一样，需要检查Java环境以及Java命令。接着下载Gaea文件，解压。

## 目录结构



* bin：该目录包含了Gaea的运行脚本，startup.sh/bat **服务名**启动服务, shutdown.sh/bat **服务名**关闭服务。

□ conf：该目录包含了全局配置文件gaea\_config.xml、gaea\_log4j.xml。gaea\_config.xml包含了Gaea框架一些全局默认配置，具体配置参数说明请查看第四章服务端参数说明，gaea\_log4j配置全局的日志输出文件。

□ docs：Gaea相关文档

□ lib：Gaea框架所需jar

□ log：日志文件

□ service：托管服务

------deploy 部署目录

------lib 所有服务公用组件(ex:jdbc驱动, memcache\_client等)

□ tmp：服务进程ID文件

□ COPYRIGHT：版权说明

□ README：说明

## 小结

本章详细介绍了在各种操作系统平台上安装Gaea，并对Gaea目录进行了深入的分析，下一章会创建一个Gaea项目，介绍如何创建和使用Gaea。

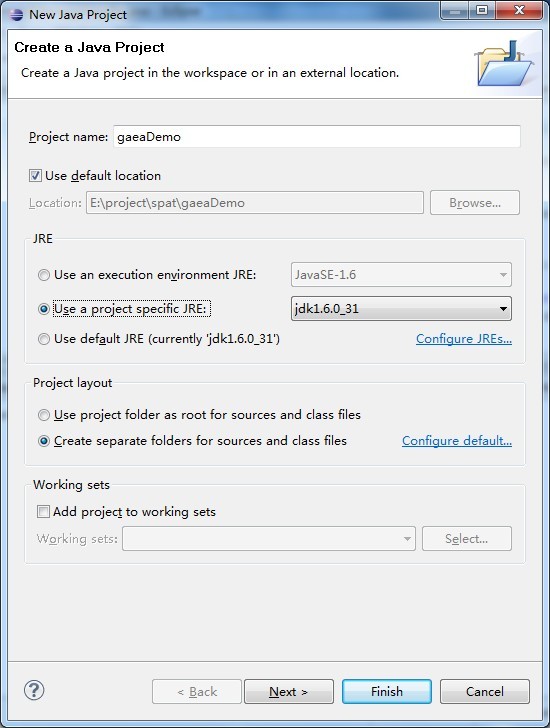
# Gaea使用入门

到目前为止，已经了解并安装好了Gaea，现在，我们开始创建一个简单的Gaea项目。

## 编写服务端

本书以Eclipse IDE为例进行代码编写。

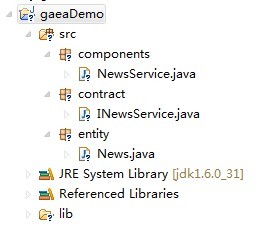
1. 创建Gaeademo java Project（File – New - Java Project）



点击Finish完成。

1. 编写主代码

在src目录下创建包components、contract、entity3个包。



contract包下创建对外提供服务接口实现类INewsService

components包下创建对外提供服务接口类NewsService

entity包下创建News实体类

InewsService类

**package** contract;

**import** java.util.List;

**import** com.bj58.spat.gaea.server.contract.annotation.OperationContract;

**import** com.bj58.spat.gaea.server.contract.annotation.ServiceContract;

**import** entity.News;

/\*\*

\* 对外提供服务接口类

\* **@ServiceContract** 标记该接口对外提供服务

\* **@OperationContract** 标记该方法对外暴露

\* **@author** **@author** Service Platform Architecture Team (spat@58.com)

\*/

@ServiceContract

**public** **interface** INewsService {

@OperationContract

**public** News getNewsByID(**int** newsID) **throws** Exception;

@OperationContract

**public** List<News> getNewsByCateID() **throws** Exception;

}

说明：

编写Gaea服务端，如果客户端需要调用该接口，则在定义接口类时必须实现注解@ServiceContract，该接口类中的接口方法需要实现注解@OperationContract

NewsService类

**package** components;

**import** entity.News;

**import** java.util.List;

**import** java.util.ArrayList;

**import** contract.INewsService;

**import** com.bj58.spat.gaea.server.contract.annotation.ServiceBehavior;

/\*\*

\* 对外提供服务接口实现类

\* **@ServiceBehavior** 标记该类对外提供服务，服务契约为INewsService

\* **@author** **@author** Service Platform Architecture Team (spat@58.com)

\*/

@ServiceBehavior

**public** **class** NewsService **implements** INewsService {

@Override

**public** News getNewsByID(**int** newsID) **throws** Exception {

**return** NewsService.*getNews*();

}

@Override

**public** List<News> getNewsByCateID() **throws** Exception {

List<News> list = **new** ArrayList<News>();

list.add(NewsService.*getNews*());

**return** list;

}

**private** **static** News getNews() {

News news = **new** News();

news.setNewsID(58);

news.setTitle("58同城一个神奇的网站");

**return** news;

}

}

说明：

编写Gaea服务端，如果客户端需要调用该接口实现类，则在定义接口实现类时必须实现注解@ServiceBehavior

News实体类

**package** entity;

**import** com.bj58.spat.gaea.serializer.component.annotation.GaeaMember;

**import** com.bj58.spat.gaea.serializer.component.annotation.GaeaSerializable;

/\*\*

\* 实体类

\* **@GaeaSerializable** 标记当前类为需要序列化的类

\* **@GaeaMember** 标记该字段为需要序列化字段

\* **@author** **@author** Service Platform Architecture Team (spat@58.com)

\*/

@GaeaSerializable

**public** **class** News {

@GaeaMember

**private** **int** newsID;

@GaeaMember

**private** String title;

**public** **int** getNewsID() {

**return** newsID;

}

**public** **void** setNewsID(**int** newsID) {

**this**.newsID = newsID;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

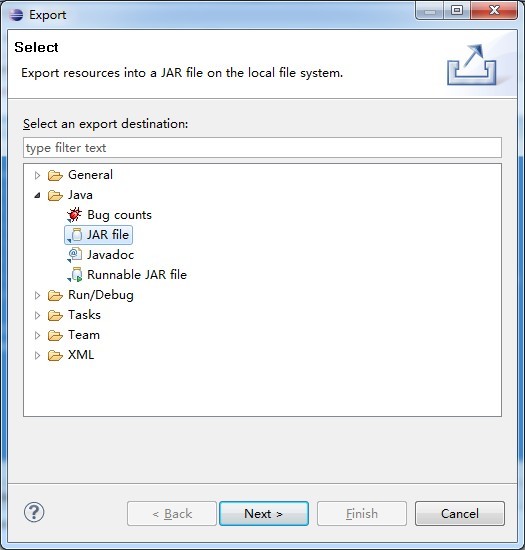
}

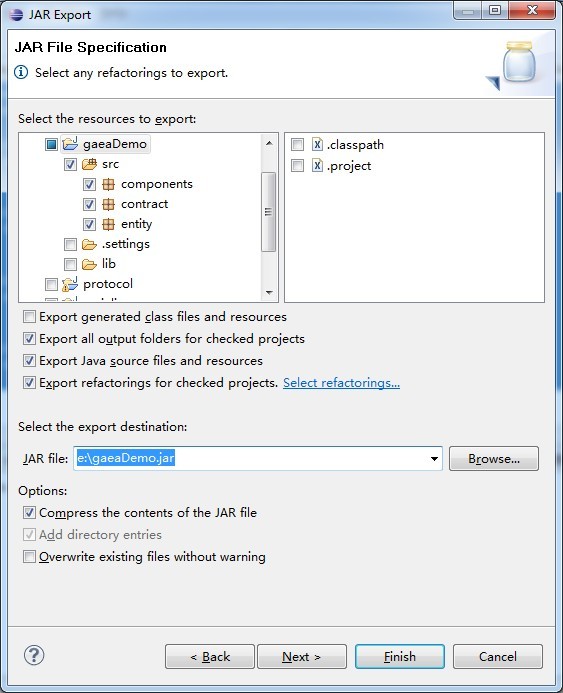
说明：

客户端调用服务端方法需要进行传输的实体类需要实现注解@GaeaSerializable，实体类内字段需要实现注解@GaeaMember

1. 导出jar

右键项目 – Export





## 编写客户端

### Java客户端

1、创建GaeaClientDemo 项目,如下图:



备注:

客户端需要依赖Gaea客户端、协议、序列化jar和服务端导出jar(gaeaDemo.jar)

com.bj58.spat.gaea.client-0.0.1-SNAPSHOT.jar

com.bj58.spat.gaea.protocol-0.0.1-SNAPSHOT.jar

com.bj58.spat.gaea.serializer-0.0.1-SNAPSHOT.jar

gaeaDemo.jar

2、编写主代码

GaeaClientTest类

**package** gaeaclientdemo;

**import** entity.News;

**import** java.util.List;

**import** contract.INewsService;

**import** com.bj58.spat.gaea.client.GaeaInit;

**import** com.bj58.spat.gaea.client.proxy.builder.ProxyFactory;

**public** **class** GaeaClientTest {

**public** **static** **void** main(String[] args) **throws** Exception {

// 加载配置文件

GaeaInit.*init*("e:/gaea.config");

/\*\*

\* 调用URL 格式:tcp://服务名//接口实现类

\* 备注:

\* 服务名：需要与gaea.config中的服务名一一对应

\* 接口实现类：具体调用接口的接口实现类

\*/

**final** String url = "tcp://demo/NewsService";

INewsService newsService = ProxyFactory.*create*(INewsService.**class**, url);

List<News> list = newsService.getNewsByCateID();

**for** (News news : list) {

System.*out*.println("ID is " + news.getNewsID() + " title is "

+ news.getTitle());

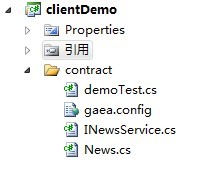
}

}

}

### .net客户端

1. 创建clientDemo项目



备注：

.net项目中需要引用

Com.Bj58.Spat.Gaea.client.dll、Com.Bj58.Spat.Gaea.Serializer.dll

1. 编写主代码

INewsService类

[ServiceContract]

public interface INewsService

{

News getNewsByID(int newsID);

List<News> getNewsByCateID();

}

备注：

需要远程方法调用的接口要实现[ServiceContract]特征

News类

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Com.Bj58.Spat.Gaea.Serializer.Component.Attributes;

namespace clientDemo.contract

{

[GaeaSerializable]

public class News

{

int newsID\_;

String title\_;

[GaeaMember]

public int newsID

{

get { return newsID\_; }

set { newsID\_ = value; }

}

[GaeaMember]

public string title

{

get{return title\_;}

set{title\_ = value;}

}

}

}

备注：

1. 需要传输的实体类要实现[GaeaSerializable]特征
2. 需要传输的实体类get/set方法要实现[GaeaMember]特征

demoTest类

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace clientDemo

{

class demoTest

{

static void Main(string[] args)

{

string url = "tcp://demo/NewsService";

clientDemo.contract.INewsService newsService = Com.Bj58.Spat.Gaea.Client.

Proxy.Builder.ProxyFactory.Create<clientDemo.contract.INewsService>(url);

var list = newsService.getNewsByCateID();

Console.WriteLine(list.Count);

Console.WriteLine(list.Count > 0 ? "" : list[0].title);

Console.ReadLine();

}

}

}

备注：

gaea.config需要放在跟demoTest类同一目录

### C++客户端

请查看c++客户端类:

test/GaeaClient.cpp

struct/struct.h

备注：

1. 运行时需要配置参数

-f/项目路径/src/protocol/SdpStruct.h /项目路径/src/struct/struct.h

struct.h需要序列化实体类对象

SdpStruct.h实现序列化操作类

### 客户端配置

gaea.config配置文件

<?xml version="1.0" encoding="utf-8" ?>

<GAEA>

<Service name="demo" id="1" maxThreadCount="50">

<Commmunication >

<SocketPool bufferSize="4096" minPoolSize="2" maxPoolSize="5" nagle="true" autoShrink="00:00:20" sendTimeout="00:00:10" receiveTimeout="00:00:09" waitTimeout="00:00:01" maxPakageSize="102400" protected="true"/>

<Protocol serialize="gaea" encoder="UTF-8" compressType="UnCompress" />

</Commmunication>

<Loadbalance>

<Server deadTimeout="00:00:10">

<!—客户端访问服务端机器负载-->

<add name="imc1" host="127.0.0.1" port="16001" maxCurrentUser="50" />

</Server>

</Loadbalance>

</Service>

</GAEA>

gaea.config 配置参数说明请查看后面4.2节内容。

## 服务部署

在下载的Gaea服务中，进入Gaea/service/deploy文件夹，创建gaeaDemo文件夹，进入gaeaDemo文件夹，拷贝gaeaDemo.jar到当前目录,并创建gaea\_config.xml、gaea\_log.xml配置文件

gaea\_config.xml

<?xml version="1.0"?>

<configuration>

<!-- service name -->

<property>

<name>gaea.service.name</name>

<value>gaeaDemo</value>

</property>

<!-- socket server listent ip -->

<property>

<name>gaea.server.tcp.listenIP</name>

<value>127.0.0.1</value>

</property>

<!-- socket server listent port -->

<property>

<name>gaea.server.tcp.listenPort</name>

<value>16001</value>

</property>

<!-- telnet server listent port -->

<property>

<name>gaea.server.telnet.listenIP</name>

<value>127.0.0.1</value>

</property>

<!-- telnet server listent port -->

<property>

<name>gaea.server.telnet.listenPort</name>

<value>26001</value>

</property>

</configuration>

gaea\_log4j.xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration xmlns:log4j='http://jakarta.apache.org/log4j/'>

<appender name="myConsole" class="org.apache.log4j.ConsoleAppender">

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="[%d{MM-dd HH:mm:ss,SSS\} %-5p] [%t] %c{2\} - %m%n" />

</layout>

<filter class="org.apache.log4j.varia.LevelRangeFilter">

<param name="levelMin" value="debug" />

<param name="levelMax" value="off" />

<param name="AcceptOnMatch" value="true" />

</filter>

</appender>

<appender name="activexAppender" class="org.apache.log4j.DailyRollingFileAppender">

<param name="File" value="/gaea/log/gaea.log" />

<param name="DatePattern" value="'.'yyyy-MM-dd'.log'" />

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="[%d{MM-dd HH:mm:ss SSS\} %-5p] [%t] %c{3\} - %m%n" />

</layout>

</appender>

<root>

<priority value="debug" />

<appender-ref ref="myConsole" />

<appender-ref ref="activexAppender" />

</root>

</log4j:configuration>

## 服务运行

进入Gaea/bin/目录,执行startup.sh/bat **服务名**启动服务，shutdown.sh/bat **服务名**停止服务。

UNIX系统下第一次使用startup.sh和shutdowm.sh脚本时需要赋予权限。

chmod +x startup.sh

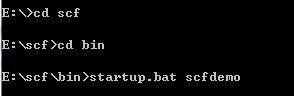
chmod +x shutdown.sh

1、Gaea服务启动:

windows下进入cmd执行startup.bat gaeaDemo

UNIX下进入Gaea/bin目录,执行./startup.sh gaeaDemo

启动过程中没有出现任何异常，并输出”server start success”表明服务已正常启动



bat.jpg

1. Gaea客户端调用

运行 GaeaClientTest类，日志输出如下:

Scan jar files begin!

开始扫描全部引用jar包，如果扫描过程过长请在启动vm参数中设置Gaea.serializer.basepakage或者设置Gaea.serializer.scantype=asyn使用异步模式扫描。

scanByURLClassLoader:/E:/Object/GaeaOpen/GaeaClientDemo/bin/

scanByURLClassLoader:/E:/Object/GaeaOpen/GaeaClientDemo/lib/com.bj58.spat.Gaea.client-1.4.4-SNAPSHOT.jar

scanByURLClassLoader:/E:/Object/GaeaOpen/GaeaClientDemo/lib/com.bj58.spat.Gaea.protocol-1.4.1-SNAPSHOT.jar

scanByURLClassLoader:/E:/Object/GaeaOpen/GaeaClientDemo/lib/com.bj58.spat.Gaea.serializer-1.4.4-SNAPSHOT.jar

scanByURLClassLoader:/E:/Object/GaeaOpen/GaeaClientDemo/lib/Gaeademo.jar

scanByURLClassLoader:/E:/Object/GaeaOpen/GaeaClientDemo/lib/log4j-1.2.15.jar

scanByURLClassLoader:/E:/Object/GaeaOpen/GaeaClientDemo/lib/commons-logging.jar

scaning com.bj58.spat.Gaea.protocol.sdp.com.bj58.spat.Gaea.protocol.sdp.ExceptionProtocol

scaning com.bj58.spat.Gaea.protocol.sdp.com.bj58.spat.Gaea.protocol.sdp.HandclaspProtocol

scaning com.bj58.spat.Gaea.protocol.sdp.com.bj58.spat.Gaea.protocol.sdp.RequestProtocol

scaning com.bj58.spat.Gaea.protocol.sdp.com.bj58.spat.Gaea.protocol.sdp.ResetProtocol

scaning com.bj58.spat.Gaea.protocol.sdp.com.bj58.spat.Gaea.protocol.sdp.ResponseProtocol

scaning com.bj58.spat.Gaea.protocol.utility.com.bj58.spat.Gaea.protocol.utility.KeyValuePair

scaning com.bj58.spat.Gaea.serializer.component.annotation.com.bj58.spat.Gaea.serializer.component.annotation.GAEASerializable

scaning com.bj58.spat.Gaea.serializer.component.com.bj58.spat.Gaea.serializer.component.ClassScaner

scaning com.bj58.spat.Gaea.serializer.component.com.bj58.spat.Gaea.serializer.component.TypeMap

scaning com.bj58.spat.Gaea.serializer.serializer.com.bj58.spat.Gaea.serializer.serializer.ObjectSerializer

scaning entity.entity.News

Scan jar files completed!

ID is 58 title is 58同城一个神奇的网站

备注：

蓝色部分为客户端调用服务端返回内容

备注：

服务端为Java版、客户端支持Java、C++、C#

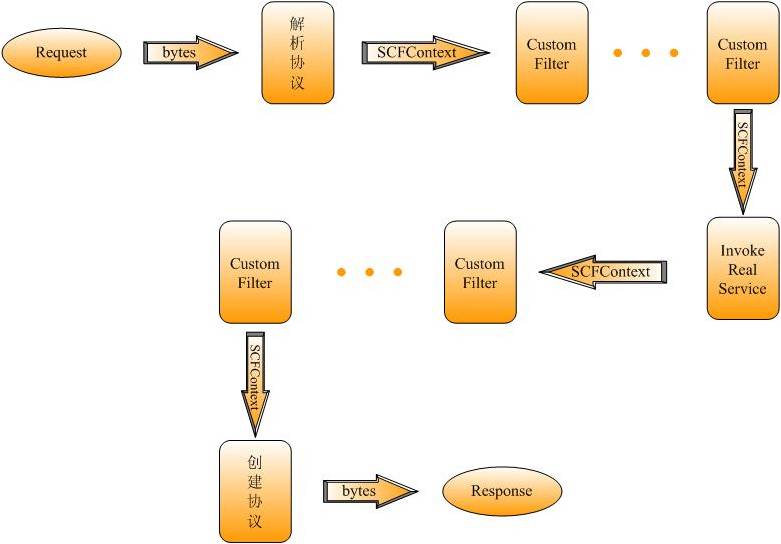
## 小结

本章主要介绍了如何编写Gaea项目，针对Gaea服务端、客户端的代码编写和服务如何部署进行了介绍。下一章将介绍Gaea的一些高级使用和流程图。

# Gaea高级使用

## 服务端

### 流程图



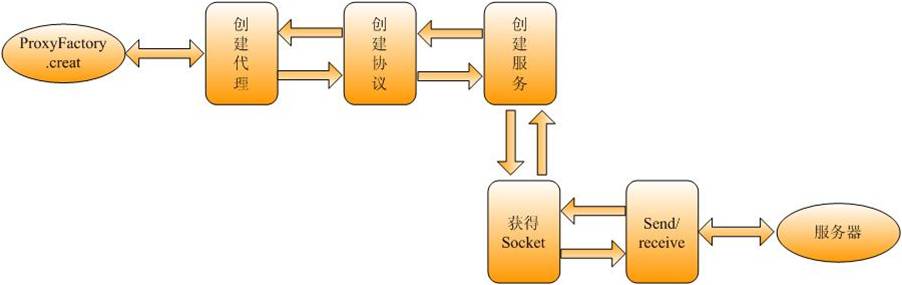
### 参数说明

服务端gaea\_config.xml各参数说明

|  |  |
| --- | --- |
| **字段** | **说明** |
| gaea.service.name | 服务名 |
| gaea.encoding | 编码格式(默认utf-8) |
| gaea.hotdeploy | 是否启动热部署 |
| gaea.filter.global.request | request全局过滤 |
| gaea.filter.global.response | response全局过滤 |
| gaea.filter.connection | 连接过滤 |
| gaea.iptable.allow.iplist | 允许访问IP |
| gaea.iptable.forbid.iplist | 禁止访问IP |
| gaea.init | 初始化类 |
| gaea.log.udpserver.ip | UDP服务IP |
| gaea.log.udpserver.port | UDP监听端口 |
| gaea.log.exectime.limit | UDP日志输出间隔 |
| gaea.proxy.invoker.implement | 服务端异步执行类 |
| gaea.async.worker.count | 服务端异步执行线程数 |
| gaea.servers | 启动服务集合 |
| gaea.server.tcp.enable | 是否启动TCP服务 |
| gaea.server.tcp.listenPort | TCP监听端口 |
| gaea.server.tcp.listenIP | TCP服务IP |
| gaea.server.tcp.receiveBufferSize | 返回字节大小(默认1024 \* 64) |
| gaea.server.tcp.sendBufferSize | 发送字节大小(默认1024 \* 64) |
| gaea.server.tcp.frameMaxLength | TCP服务最大支持长度(默认1024 \* 512) |
| gaea.server.tcp.workerCount | TCP服务工作线程数 |
| gaea.server.tcp.implement | TCP 启动服务类 |
| gaea.server.http.enable | 是否启动HTTP服务 |
| gaea.server.http.listenPor | HTTP监听端口 |
| gaea.server.http.listenIP | HTTP服务IP |
| gaea.server.http.receiveBufferSize | 返回字节大小(默认1024 \* 64) |
| gaea.server.http.sendBufferSize | 发送字节大小(默认1024 \* 64) |
| gaea.server.http.frameMaxLength | HTTP服务最大支持长度(默认1024 \* 512) |
| gaea.server.http.workerCount | HTTP服务工作线程数 |
| gaea.server.http.implement | HTTP启动服务类 |
| gaea.server.telnet.enable | 是否启动telnet监测服务 |
| gaea.server.telnet.listenPort | telnet监听端口 |
| gaea.server.telnet.listenIP | telnet服务IP |
| gaea.server.telnet.receiveBufferSize | 返回字节大小(默认1024 \* 64) |
| gaea.server.telnet.sendBufferSize | 发送字节大小(默认1024 \* 64) |
| gaea.server.telnet.frameMaxLength | telnet服务最大支持长度(默认1024 \* 512) |
| gaea.server.telnet.workerCount | telnet服务工作线程数 |
| gaea.server.telnet.implement | telnet启动服务类 |
| gaea.secure | 是否启动权限认证 |

## 客户端

### 流程图



### 参数说明

**Service**

|  |  |
| --- | --- |
| name | 远程服务名称 |
| id | 服务ID |
| maxThreadCount | 最大线程连接数 |

**SocketPool**

|  |  |
| --- | --- |
| bufferSize | 网络receive传输流缓存大小(最小10KB) |
| minPoolSize | 连接池中最小连接数量 |
| maxPoolSize | 连接池中最大连接数量 |
| nagl | 是否启用Nagle算法 |
| autoShrink | 是否启动连接自动回收 |
| sendTimeout | 发送超时时间 |
| receiveTimeout | 接收超时时间 |
| waitTimeout | 当连接池中没有连接时需要等待释放连接的时间 |
| maxPakageSize | 网络send传输流缓存大小(默认50KB) |
| reconnectTime | 超时重新连接次数 |
| protected | 是否根据请求数量自动调节超时时间.默认为true |

**Protocol**

|  |  |
| --- | --- |
| serialize | 序列化方式(gaea、json) |
| encoder | 序列化采用的编码格式 |
| compressType | 是否压缩(当前版本不支持压缩，默认为UnCompress) |

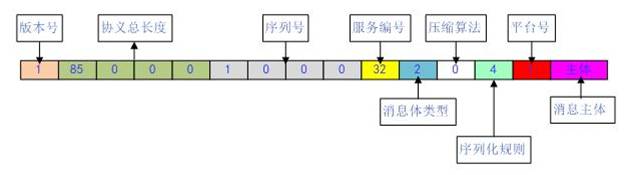
**Server**

|  |  |
| --- | --- |
| deadTimeout | 服务器心跳检测间隔时间 |
| name | 服务名称 |
| host | 服务IP |
| port | 服务端口 |
| maxCurrentUser | 最大当前用户数，超过此数将服务器设置为busy状态 |

**Secure**

|  |  |
| --- | --- |
| info | Key内容 |

## 协议



尾分隔符09,11,13,17,18

|  |  |
| --- | --- |
| **编码** | **说明** |
| 009 | 水平制表符 |
| 011 | 垂直制表符 |
| 013 | 回车键 |
| 017 | 键盘控制码 |
| 018 | 键盘控制码 |

版本号：默认为1

序列号：SessionId 每次自加,当数值大于1024 \* 1024 \* 1024时重新赋值为1

服务编号：客户端配置文件中serviceid

消息体类型：Response(1),Request(2),Exception(3),Config(4),Handclasp(5),Reset(6);

压缩算法：不压缩UnCompress(0),SevenZip(1), DES(2);

序列化规则：JSON(1),JAVABinary(2),XML(3),GAEABinary(4);

平台号：Dotnet(0),Java(1),C(2);

## 序列化

### Object序列化

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Object** | | | | | | |
| 4 | 1 | 4 | 1 | 4N(N>0) | 属性类型Id | 数据 |
| 属性类型Id | 数据 |
| 类型Id | 是否引用 | Hashcode | 泛型参数数量 | 泛型参数类型Id | 数据 | |
| 备注:如果第5个字节是1那么只写入该对象的Hashcode, 如果类型是泛型时才输出灰色部分 | | | | | | |

### Array/List/Map序列化

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Array/List/Map** | | | | | |
| 4 | 1 | 4 | 4 | 属性类型Id | 数据 |
| 属性类型Id | 数据 |
| 类型编码 | 是否引用 | Hashcode | 数组长度 | 数据 | |

**ArraySerializer**

**WriteObject**

1、写入4byte的类型编码

2、写入1byte判断是否为引用类型，1为引用类型(如果输入值为null,写入1,0返回,获取当前输入值的hashcode ,如果包含当前hashcode则写入1，hashcode并返回,如果不包含写入0，hashcode)

3、根据类型编码获取class,判断属于什么类型,根据不同类型分别调用不同类型的序列化方式

4、写入数据长度

5、写入数据内容

**ReadObject**

1、读取4byte获得类型编码，如果类型编码为0返回null

2、读取1byte判断判断是否为引用类型，如果获取值大于0为则引用类型,返回当前hashcode对应object

3、读取4byte获得hashcode

4、根据类型编码获取class,判断属于什么类型，根据不同类型分别调用不同类型的序列化方式

6、获取数据长度

7、获取数据内容

### 基本类型

|  |  |  |  |
| --- | --- | --- | --- |
| **基本类型** | | | |
| Byte | 1 | Int16 | 2 |
| Int32 | 4 | Int64 | 8 |
| Float | 8 | Double | 8 |
| Char | 2 | DateTime | 8 |
| Boolean | 1**(**0:false 1:true**)** | | |

**EnumSerializer**

**WriteObject**

1、写入4byte类型编码

2、调用String序列化方式进行序列化

**ReadObject**

1、读取4byte类型编码

2、根据类型编码获得class

3、根据class类型,String发序列化方式读取

4、Enum.valueOf(classType, value)返回

### String

|  |  |  |  |
| --- | --- | --- | --- |
| **String** | | | |
| 1 | 4 | 4 | N |
| 是否引用 | Hashcode | 总长度 | 数据 |

## Gaea服务监测

Gaea服务端提供了良好的监测功能，可以用来监测具体方法执行时间，并发数等。

1. **Gaea分析功能**

count[|second num|method methodName]

\* show method call times in num seconds

\* second : in num seconds statistics once (num default 1)

\* method : for statistics method

\* example : count

\* example : count|second 3

\* example : count|second 3|method getInfo

time|grep abc[|group num|column -tkda]

\* show method execute time

\* grep : condition

\* group : method called num times show statistics once

\* column : show column a->all t->time k->key d->description

\* example: time|grep getInfo

\* example: time|grep getInfo|group 10|column -tk

exec|top

|netstat -na

\* exec command (at present only allow:top or netstat)

\* example: exec|top

control \* use for control Gaea-server

help \* show help

quit \* quit monitor

**使用示例**

使用方式以gaeaDemo为例，gaeaDemo对外提供的监控端口为26001，该端口可以通过查看 Gaea/service/deploy/服务名/gaea\_config.xml

<!-- telnet server listent port -->

<property>

<name>gaea.server.telnet.listenPort</name>

<value>26001</value>

</property>

1、telnet 127.0.0.1 26001

2、查看getInfo方法的执行情况：

time|grep getInfo回车

3、查看所有方法的执行情况：

time|grep \_回车

输出结果如下：

time:2ms--key:InvokeRealService\_NewsService.getNewsByCateID--description:protoco

l version:1

fromIP:127.0.0.1

lookUP:NewsService

methodName:getNewsByCateID

params:

4、查看服务并发数：  
count回车

5、退出quit

6、要想保存结果可以在telnet的时候把输出重定向到一个文件中

例：telnet 127.0.0.1 26001 > tmp