

GAEA 帮助文档

58 同城-技术中心-架构部

2012 年 2 月

目 录

前言.....	4
第 1 章 Gaea 简介	6
1.1 什么是 Gaea	6
1.2 Gaea 组成	6
1.3 特点.....	6
1.4 应用场景.....	7
1.5 架构.....	8
1.6 发展历史.....	10
1.7 总结.....	10
第 2 章 Gaea 容器安装与配置	11
2.1 在 windows 上安装 Gaea	11
2.1.1 检查 JDK 安装.....	11
2.1.2 下载 Gaea	11
2.1.3 本地安装.....	11
2.2 在基于 UNIX 的系统上安装 Gaea	12
2.3 Gaea 容器目录结构	12
2.4 小结.....	13
第 3 章 Gaea 使用入门	14
3.1 编写服务端.....	14
3.2 编写客户端.....	19
3.2.1 Java 客户端	19
3.2.2 .net 客户端	21
3.2.3 C++客户端.....	23
3.3 客户端配置.....	23
3.4 服务部署.....	24
3.5 服务运行.....	25
3.6 小结.....	27

第 4 章 Gaea 高级使用	28
4.1 服务端.....	28
4.1.1 流程图.....	28
4.1.2 参数说明.....	28
4.2 客户端.....	30
4.2.1 流程图.....	30
4.2.2 参数说明.....	30
4.3 协议.....	32
4.4 序列化.....	32
4.4.1 Object 序列化	32
4.4.2 Array/List/Map 序列化	33
4.4.3 基本类型.....	33
4.4.4 String.....	34
4.5 Gaea 服务监测	34

前言

随着 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，新产品不断地被研发，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 部分：客户端、协议、序列化、服务端，客户端现支持 c++、c#、java 等主流平台。

本书使用的约定

本书主要针对 Gaea 的使用进行讲解，包括 c++、c#、java 客户端，服务端由 java 开发。本文中提到的 Gaea 为 Gaea 本身，包括客户端、服务端、协议、序列化，Gaea 容器指可以部署服务的容器本身。

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

第1章 Gaea简介

1.1 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。

1.2 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。
- ☐ **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。

1.3 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 (High Availability)，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++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，不需要知道如何搭建一个服务集群以及怎么做 HA，这些框架都做了封装，需要做的只是定义一个接口，实现接口方法，标注注解。可以便捷的使一般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。

1.4 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

可以对各个系统间公有部分抽象成一个 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，未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。

1.5 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**: 该模块负责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自服务端的响应，该模块通过 SessionID 和等待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中断的热部署。
- **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，避免了反射调用对性能的影响。
- **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实时的监控，分析服务的运行情况，当服务出现问题时可以快速定位问题。

1.6 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

1.7 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。

第2章 Gaea容器安装与配置

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。


2.1 在windows上安装Gaea

2.1.1 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
```



```
C:\Users\Administrator>echo %JAVA_HOME%  
C:\Program Files\Java\jdk1.6.0_31  
  
C:\Users\Administrator>java -version  
java version "1.6.0_31"  
Java(TM) SE Runtime Environment (build 1.6.0_31-b05)  
Java HotSpot(TM) 64-Bit Server VM (build 20.6-b01, mixed mode)
```

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。

2.1.2 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 容器下载文件。Windows 下为 gaea.zip，Linux/UNIX 下为 gaea.tar.gz。

2.1.3 本地安装

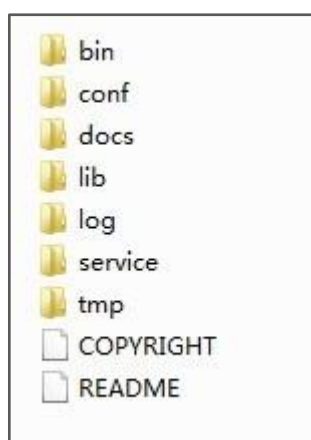
将安装文件解压到指定目录。

2.2 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。

与 Windows 上安装 Gaea 一样,检查 Java 环境以及 Java 命令。然后下载 Gaea 文件,并解压到目标目录。

2.3 Gaea容器目录结构



- ☐ **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.xml 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

2.4 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。

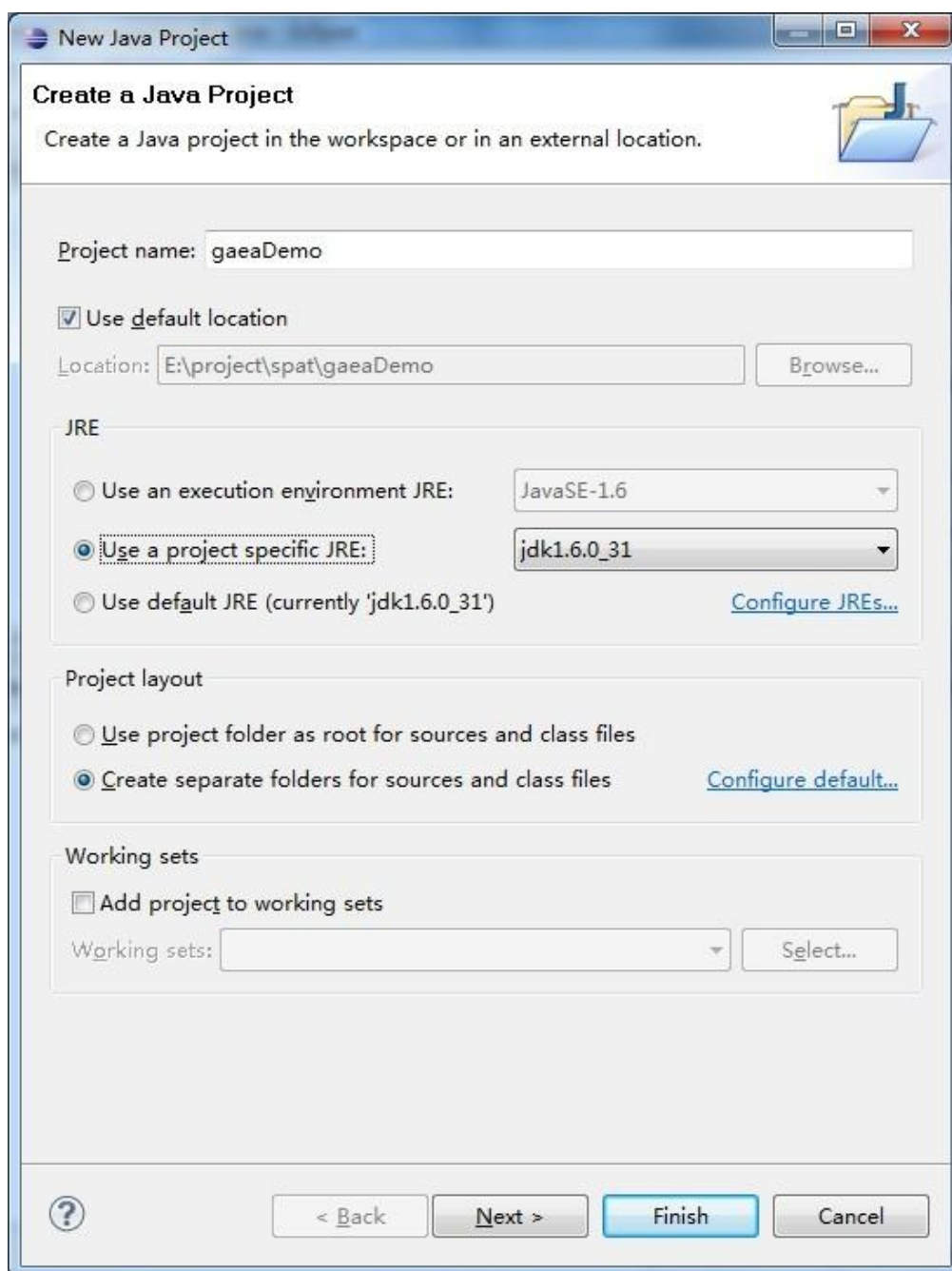
第3章 Gaea使用入门

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。

3.1 编写服务端

本书以 Eclipse IDE 为例进行代码编写。

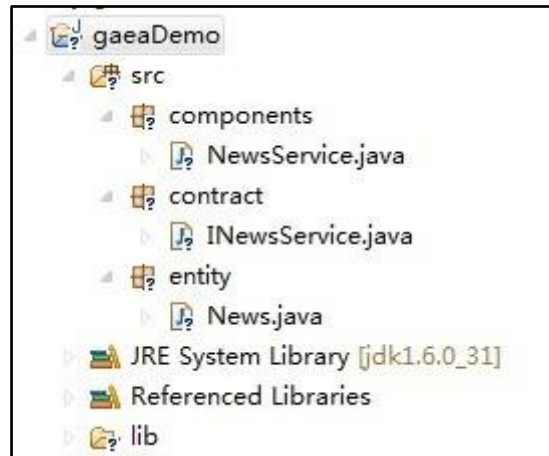
1、创建 gaeaDemo java Project (File - New - Java Project)



点击 Finish 完成。

2、编写主代码

在 src 目录下创建包 components、contract、entity



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 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:

Gaea 服务端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 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 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;
    }

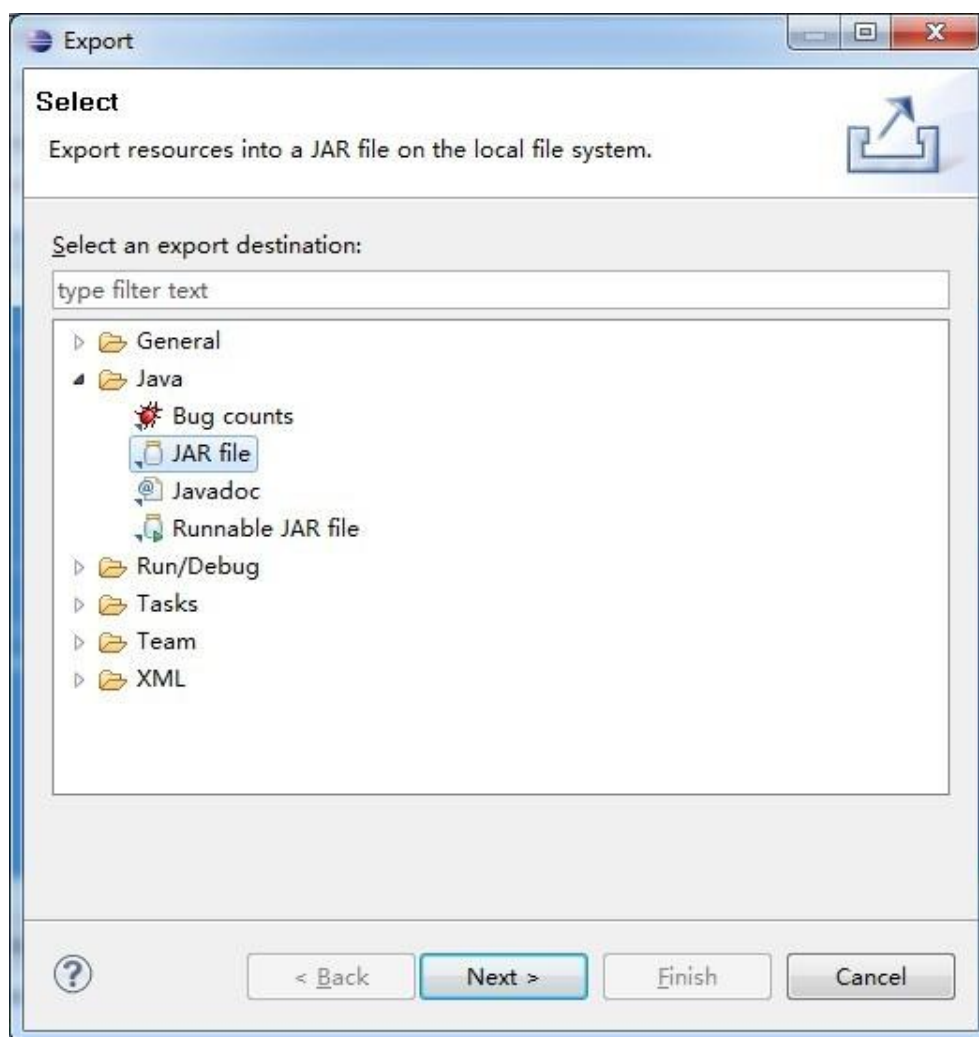
    public void setTitle(String title) {
        this.title = title;
    }
}
```

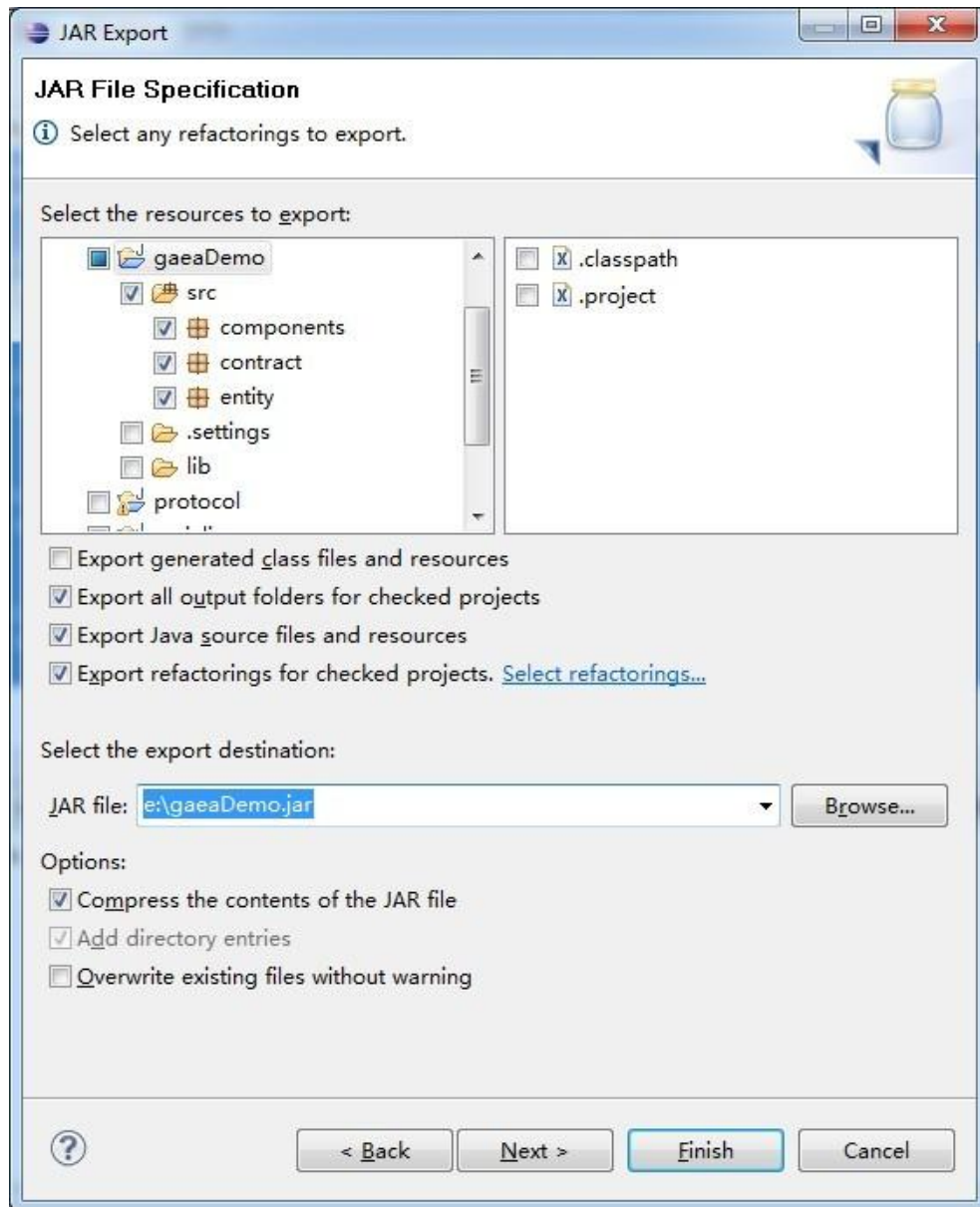
说明:

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

3、导出 jar

右键项目 - Export

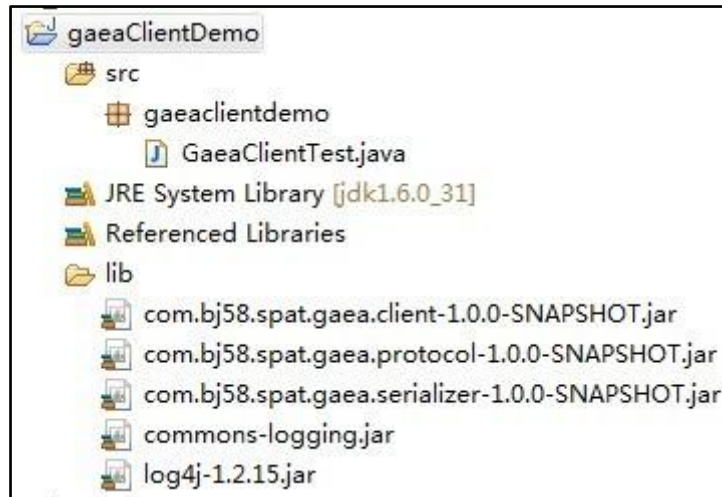




3.2 编写客户端

3.2.1 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-1.0.0.jar

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

com.bj58.spat.gaea.serializer-1.0.0.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.Gaealnit;
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
        Gaealnit.init("/路径/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
         */
    }
}
```

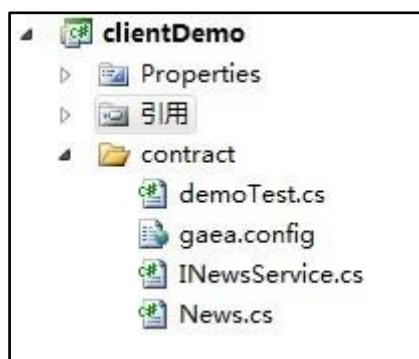
```

    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());
    }
}
}

```

3.2.2 .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

2、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.getNewsByCatelD();
        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

3.2.3 C++客户端

请查看 c++客户端类:

test/GaeaClient.cpp

struct/struct.h

备注:

运行时需配置参数

-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

3.3 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">
```

```
<Communication >
```

```

    <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" />
```

```
</Communication>
```

```

    <Loadbalance>
        <Server deadTimeout="00:00:10">
            <add name="demo1" 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。

3.4 服务部署

在下载 Gaea 容器中，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}\n - %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}\n - %m%n" />
    </layout>
  </appender>

  <root>
    <priority value="debug" />
    <appender-ref ref="myConsole" />
    <appender-ref ref="activexAppender" />
  </root>
</log4j:configuration>

```

3.5 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 shutdown.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, 进入 gaea/bin 目录, 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

```
D:\gaea\bin>startup.bat gaeaDemo
```

```
[07-30 15:58:10,979 INFO ] [main] bootstrap.Main - ++++++ server  
start success!!! ++++++
```

2、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.basepackage 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-1.0.0.jar

scanByURLClassLoader:/E:/Object/GaeaOpen/GaeaClientDemo/lib/com.bj58.spat.gaea.protocol-1.0.0.jar

scanByURLClassLoader:/E:/Object/GaeaOpen/GaeaClientDemo/lib/com.bj58.spat.gaea.serializer-1.0.0.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

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

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

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

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

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

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

scanning

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

scanning com.bj58.spat.Gaea.serializer.component.com.bj58.spat.Gaea.serializer.component.ClassScanner

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

scanning com.bj58.spat.Gaea.serializer.serializer.com.bj58.spat.Gaea.serializer.serializer.ObjectSerializer

scanning 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#

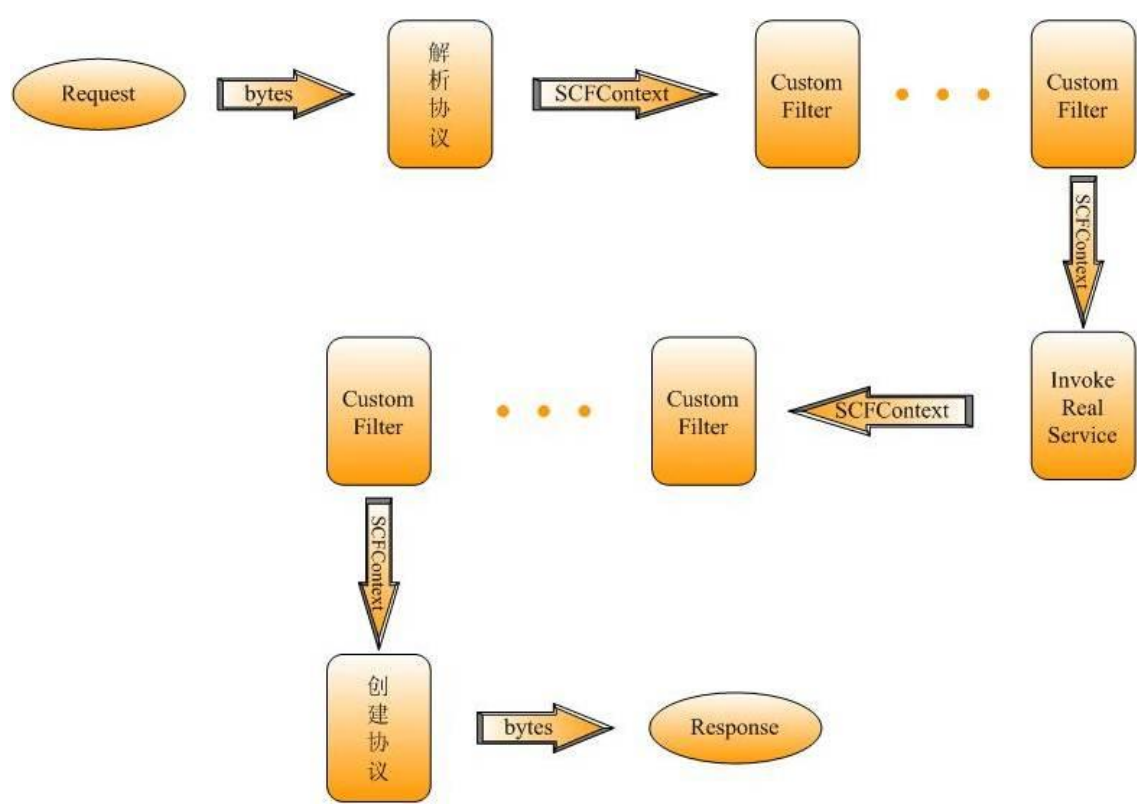
3.6 小结

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。

第4章 Gaea高级使用

4.1 服务端

4.1.1 流程图



4.1.2 参数说明

服务端 gaea_config.xml 各参数说明

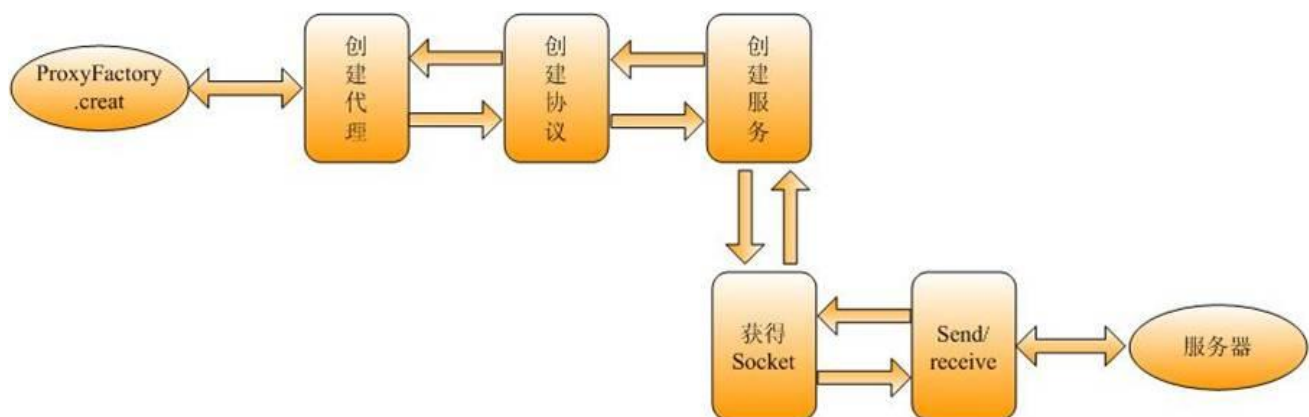
字段	说明
gaea.service.name	服务名
gaea.encoding	编码格式(默认 utf-8)
gaea.hotdeploy	是否启动热部署
gaea.filter.global.request	request 全局过滤
gaea.filter.global.response	response 全局过滤

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.udpservice.ip	UDP 服务 IP
gaea.log.udpservice.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

4.2 客户端

4.2.1 流程图



4.2.2 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)

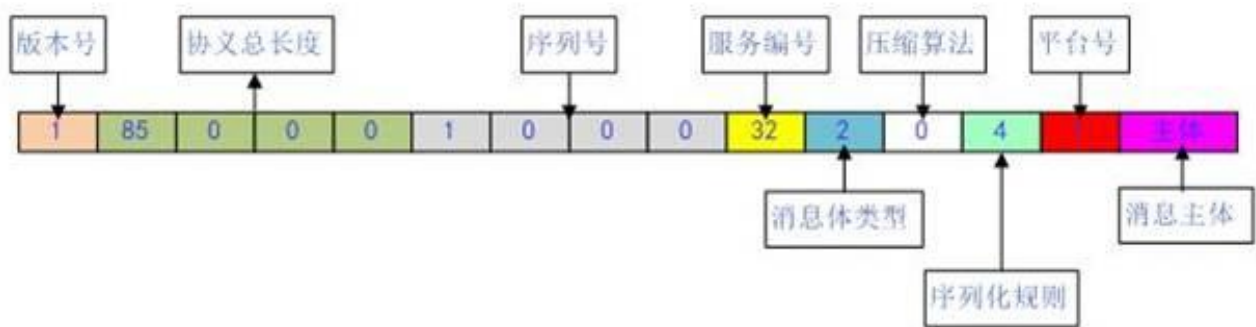
Server

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
------	--------

4.3 协议



尾分隔符 09, 11, 13, 17, 18

版本号：默认为 1

序列号：SessionId 每次自增, 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);

4.4 序列化

4.4.1 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						

4.4.2 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

4.4.3 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

4.4.4 String

String			
1	4	4	N
是否引用	Hashcode	总长度	数据

4.5 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:protocol 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