

基于WIFI探针的数据分析系统

体系结构文档



2017-6-24

南京大学软件学院

Coding Fairy

目录

[1. 需求 4](#_Toc492195282)

[1.1. 需求定义 4](#_Toc492195283)

[1.2. 用例图 5](#_Toc492195284)

[2. 架构功能简述 5](#_Toc492195285)

[2.1. 主要模块 5](#_Toc492195286)

[2.1.1. statDisplay前端展示页面 5](#_Toc492195287)

[2.1.2. Server:web服务器 6](#_Toc492195288)

[2.1.3. Receiver: wifi 探针接收服务器 6](#_Toc492195289)

[2.1.4. mapreduce:计算与存储平台 6](#_Toc492195290)

[2.2. 开发模块目录 6](#_Toc492195291)

[2.3. 模块交互图 6](#_Toc492195292)

[2.4. 主要功能执行流程 7](#_Toc492195293)

[2.4.1. wifiprob数据接收流程描述 7](#_Toc492195294)

[- 探针发送数据 7](#_Toc492195295)

[- 负载均衡部件选择单个请求的探针接收服务器 7](#_Toc492195296)

[- 被选择的探针接收服务器接收数据 8](#_Toc492195297)

[2.4.2. 用户请求分析数据流程描述 8](#_Toc492195298)

[2.4.3. hadoop平台mapreduce任务 9](#_Toc492195299)

[2.5. 模块接口描述 9](#_Toc492195300)

[2.5.1. Hadoop(HDFS)提供的接口 9](#_Toc492195301)

[3. 详细设计 11](#_Toc492195302)

[3.1. MapReduce 11](#_Toc492195303)

[3.1.1. MapReduce算法描述 11](#_Toc492195304)

[- 运行前提 11](#_Toc492195305)

[- 执行过程 11](#_Toc492195306)

[- 算法运行示例 12](#_Toc492195307)

[3.1.2. mapreduce包设计 15](#_Toc492195308)

[- **common 公共依赖的代码** 16](#_Toc492195309)

[- config 配置类 16](#_Toc492195310)

[- TO 传输对象封装 16](#_Toc492195311)

[- tool 常用工具类 17](#_Toc492195312)

[- VO value object 17](#_Toc492195313)

[- **整体分析与计算的包** 17](#_Toc492195314)

[- classify 安装用户mac地址分类数据--第一个mapreduce 17](#_Toc492195315)

[- flow流式有序的扫描并分析数据--第二个mapreduce 17](#_Toc492195316)

[- logic 复杂逻辑实现 18](#_Toc492195317)

[**3.2.** Web 18](#_Toc492195318)

[3.2.1. 处理用户请求 18](#_Toc492195319)

[3.2.2. 主要实现分包 19](#_Toc492195320)

[- Bl：业务逻辑 19](#_Toc492195321)

[- data：数据访问层以及持久化对象 19](#_Toc492195322)

[包括数据库映射实体，数据库访问层，数据库配置对象以及协助类。 20](#_Toc492195323)

[- Exception: 所有的自定义异常封装。 20](#_Toc492195324)

[- Util: 工具类封装 20](#_Toc492195325)

[- Web: web controller 20](#_Toc492195326)

[**3.3.** Receiver 20](#_Toc492195327)

[3.3.1. 负载均衡 20](#_Toc492195328)

[3.3.2. Session一致性 21](#_Toc492195329)

[3.3.3. 数据存储 21](#_Toc492195330)

[- bl 21](#_Toc492195331)

[- web 23](#_Toc492195332)

[4. 部署视图 23](#_Toc492195333)

[4.1. 部署方式 23](#_Toc492195334)

[4.2. 节点交互 24](#_Toc492195335)

[4.2.1. 用户与web服务器 24](#_Toc492195336)

[4.2.2. wifi探针与接收服务器 24](#_Toc492195337)

[4.2.3. Web服务器与HDFS 24](#_Toc492195338)

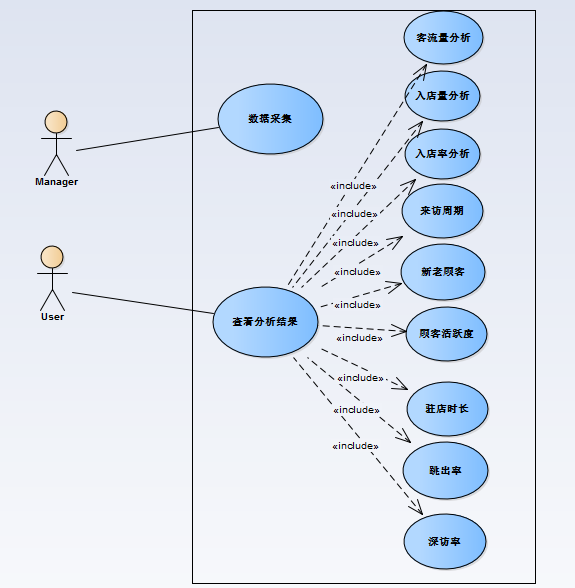
[4.2.4. 接收服务器Receiver与HDFS 24](#_Toc492195339)

1. 需求
   1. 需求定义

具体需求描述见需求文档。

|  |  |  |
| --- | --- | --- |
| 需求 | 需求描述 | 需求优先级 |
| WIFI探针 | 硬件设备 | 高 |
| 数据采集 | 收集用户数据 | 高 |
| 客流量分析 | 收集数据后处理 | 中 |
| 入店量分析 | 收集数据后处理 | 中 |
| 入店率分析 | 收集数据后处理 | 中 |
| 来访周期分析 | 收集数据后处理 | 中 |
| 新老顾客分析 | 收集数据后处理 | 中 |
| 顾客活跃度 | 收集数据后处理 | 中 |
| 驻店时长分析 | 收集数据后处理 | 中 |
| 跳出率分析 | 收集数据后处理 | 中 |
| 深访率分析 | 收集数据后处理 | 中 |
| 安全性 | 防XSS，防注入，防Ddos | 高 |
| 可维护性 | 项目架构与代码风格 | 中 |
| 易用性 | 前端交互 | 中 |
| 可靠性 | 扛并发能力，数据吞量 | 中 |
|  |  |  |

* 1. 用例图



1. 架构功能简述
   1. 主要模块

系统分为四部分：

* + 1. statDisplay前端展示页面

**-** 调用web服务器，展示实时数据与分析结论

* + 1. Server:web服务器

**-** 提供RESTful的接口，可获取实时数据与分析结果

* + 1. Receiver: wifi 探针接收服务器

**-** 提供多服务器的、可靠的、高并发的、大存储量的wifi探针数据接收接口。

* + 1. mapreduce:计算与存储平台

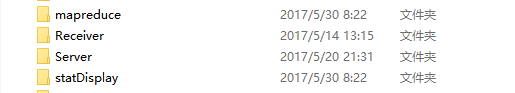
- hadoop计算集群 对Receiver收集的数据进行全局的统计与分析

- hdfs文件系统 按块存储所有的数据

- 关系型数据库 存储高频度使用的数据，主要是对探针数据的统计结果

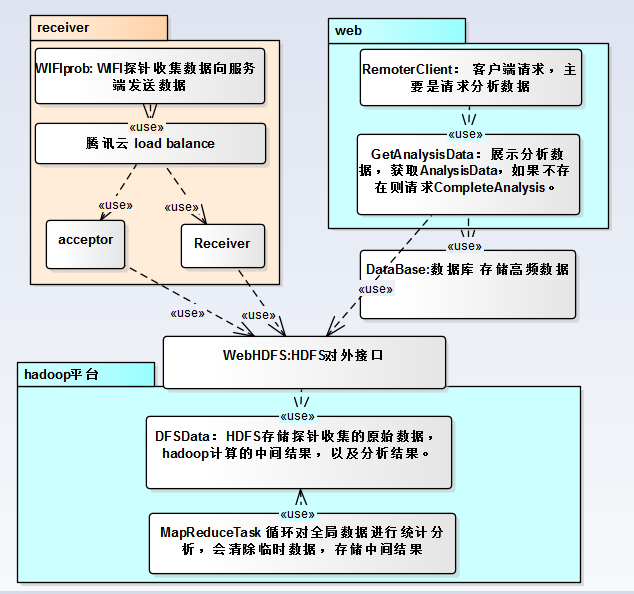
* 1. 开发模块目录

子项目目录图如下：



* 1. 模块交互图

下图刻画了各个子项目之间的逻辑关系与交互方式。



* 1. 主要功能执行流程
     1. wifiprob数据接收流程描述
        + 探针发送数据

我们选用的探针是选购的设备，可配置网络，可配置接收服务器以及发送频率等等。探针上传方式是HTTP post方式。

* + - * 负载均衡部件选择单个请求的探针接收服务器

在采购腾讯的负载均衡前，我们使用NGINX作为单节点转发，转发策略是轮询方式（因为发送数据频率与数据大小在概率上开始均衡的）。

目前采用了腾讯的负载均衡，探针设备设置的服务器为负载均衡的浮动IP，处理会被，转发至任意一台receiver服务器。

具体的抗压能力详见*测试文档。*

* + - * 被选择的探针接收服务器接收数据

探针接收服务器会对每条数据进行初步处理，这里会做一些统计分析，尤其是比较好进行增量分析的、不需要所有数据即可分析的统计。（比如总客户统计是很好进行增量分析的，但是回访量分析却没有很好的算法进行增量分析）处理后会将数据存入队列，这个缓冲队列初次采用了redis的内置队列实现，但是考虑到队列功能上的单一性，我们后来直接采用了java内存实现这个多线程的队列。

队列中数据达到一定量时会异步的写入HDFS文件系统（这里是为了尽可能的减少与HDFS的交互次数，也是为了较少HDFS的小文件数量，这个大小被设置为127M，如果离上一次提交时间超过20分钟，不论缓冲大小，都会提交。这样理论上仍有可能会造成HDFS小文件，但是这会在HADOOP循环任务中处理。）

* + 1. 用户请求分析数据流程描述

1. 用户发起请求
2. 界面将请求转发的web服务器
3. web服务器收到请求
4. web 服务器查询本地缓存是否有用户所需数据，有则返回
5. web 服务器查询数据库是否有用户所需数据，有则返回
6. web 服务器读取HDFS文件系统中的数据，并根据需要存入数据库与本地缓存中，都到后即返回。这里HDFS通过WEBHDFS以及Kerberos授权的方式提供远程调用。
   * 1. hadoop平台mapreduce任务

hadoop集群会循环执行以下任务

1. 合并hdfs中的小文件
2. 执行mapreduce任务
3. 清除临时文件
4. 清除过期统计数据
   1. 模块接口描述
      1. Hadoop(HDFS)提供的接口

|  |  |  |
| --- | --- | --- |
| Hdfs(WebHDFS)接口，使用时要求   1. WebHDFS已经启动 2. 客户端已经配置hosts文件，如master指向对应的IP 3. webHDFS配置授权 | | |
| 1 | 接口规范 | void uploadFiles(  InputStream in,String outputFile) |
| 接口描述 | 上传数据至HDFS的指定目录 |
| 前置条件 | OutputFile父目录已存在，且指向的文件不存在 |
| 2 | 接口规范 | DataInputStream readFromHdfs(  String fileName) |
| 接口描述 | 读取HDFS中对应的文件至本地 |
| 前置条件 | Filename对应的文件必须存在且必须是文件 |
| 3 | 接口规范 | void deleteFromHdfs(String fileName) |
| 接口描述 | 删除HDFS中对应的文件 |
| 前置条件 | Filename对应的文件存在且必须是文件 |
| 4 | 接口规范 | void emptyDirectory(String directory) |
| 接口描述 | 清空某一个目录下面的所有文件 |
| 前置条件 | 对应的directory存在且必须是目录 |
| 5 | 接口规范 | List<String> getDirectoryFromHdfs(  String directory) |
| 接口描述 | 读取某一个目录下面的所有文件名 |
| 前置条件 | 对应的directory存在且必须是目录 |
| 6 | 接口规范 | void renameFile(  String origin, String newName) |
| 接口描述 | 重命名某一个文件 |
| 前置条件 | 对应origin的文件必须存在，newName对应的文件不存在，且origin与newName在同一个目录下 |
| 7 | 接口规范 | void concat(String dir) |
| 接口描述 | 合并对应目录下面的所有文件为一个大文件 |
| 前置条件 | 对应dir的目录存在且必须是目录 |
| 9 |  |  |

1. 详细设计
   1. MapReduce
      1. MapReduce算法描述

系统使用map-reduce算法统计当前为止的客户流量，新老客户，回访周期，入店时长。算法使用两趟map-reduce来得出统计结果。

* + - * 运行前提

Hadoop平台已经搭建好，HDFS文件系统可以访问。

* + - * 执行过程

**(a)** 运行map-reduce程序前会清空HDFS临时文件夹。

**(b)** 第一次map-reduce，我们会把HDFS文件系统的所有探针数据按照用户MAC地址分类，并按照时间排序，存入HDFS临时文件夹中。

所有的用户行踪均根据探测到的指针来判断。

**(c)** 第二次map-reduce，我们会对每个用户(mac)已排序好的探针数据，统计探针数据在时间轴上的出现密度，由此得出用用户在店内的时间段，继而得出根据单个用户数据统计出来的客户流量，新老客户，回访周期，入店时长这几个统计数据。

**(d)** 统计单个用户的数据后，然后需要进行归并。

1. 对于客户流量与新老客户，是分时间段归并的。
2. 对于回访周期与入店时长，是分时间长短归并的。

从语言形式上与实际处理上来说，这两者的归并方式都差不多。

* + - * 算法运行示例

**(对应下表第二个map的结果)**对单个用户数据统计结果如下：

*用户A: 9：30点入店1次，作为新客户入店1次，作为老客户入店0次*

*11：20点入店1次，**作为新客户入店0次，作为老客户入店1次*

*11：30点入店1次，作为新客户入店0次，作为老客户入店1次*

*用户B: 9:40点入店1次，作为新客户入店1次，作为老客户入店0次*

*11:40点入店1次，作为新客户入店0次，作为老客户入店1次*

那么

**(对应第二个combiner的结果)**对多个用户数据在单台（mapper）机器上统计结果如下：

*9-10点入店2次，作为新客户入店2次，作为老客户入店1次*

*10-11点入店3次，作为新客户入店0次，作为老客户入店3次*

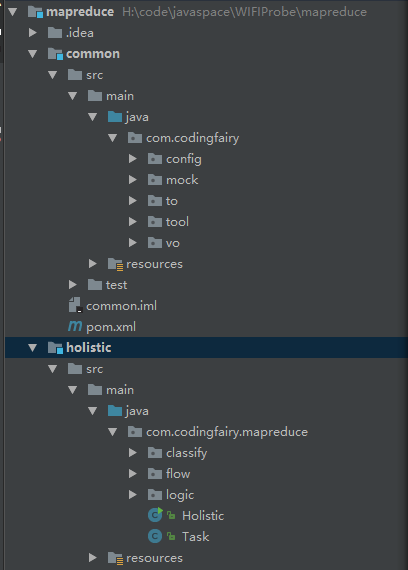
**(对应第二个reduce)**当用户A,B的数据被不同的mapper读取时，以上combiner的工作应当由reducer做。

Mapreduce算法可以如下表所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | key | value | 备注 |
|  | | | | |
| 第一次读入数据 | | 行号 | 探针数据包 |  |
| 第一个map输出 | | 用户的mac地址 | 探针数据包 | 探针数据包中含有mac地址,我们把这个mac地址拿出来作为key，数据包作为value |
| combiner输出 | | 用户的mac地址 | 探针数据包 |  |
| 第一个reducer输出 | | 用户的mac地址 | 探针数据包 | 这里把每个用户的发送的数据包都归一后按照数据包的时间戳排序后存储，存储格式为：*用户mac地址【空格】用户所有数据包json列表* |
| 第二次读入数据 | | 行号 | *用户mac地址【空格】用户所有数据包* | 这里的所有数据包都是时间上排过序的 |
| 第二个map输出 | 客户流量 | “Flow”+Hour | (Hour,Hour+1)期间入店数，出店数，深访率，跳出率 | 都是统计每一个时间区间内的响应统计项的数量。如：*2016.5.30日4到5点的入店数为12，出点数为45，深访率0.34，跳出率0.12，新客户7，老客户5。* |
| 新老客户 | “newOld”+Hour | (Hour,Hour+1)期间新客户数，老客户数 |
| 回访周期 | “cycle”+时间区间（t,t+1） | 回访周期时间长度居于（t,t+1）区间的客户数 | 都是统计落在某个区间的的客户数。  比如：*回访周期为（6-12小时）的客户有122个，入店时长为（3-5分钟）的客户有1241个* |
| 入店时长 | “inStore”+时间区间（t,t+1） | 入店时长时间长度居于（t,t+1）区间的客户数 |
| combiner输出 | 客户流量 | 同map输出 | 同map输出 | 对来自同一个map同一个KEY的数据累加,比如在1:05跟1:45的分别有一人入店，那么输入为{(1,2)=>1},{(1,2)=>1}（表示两个人都是在1-2点进入了店铺，但是没有合并），那么combiner会合并为{(1,2)=>2} |
| 新老客户 |
| 回访周期 |
| 入店时长 |
| 第二个reducer输出 | 客户流量 | 同map输出 | 内容同map输出  但会转换成json格式的Text | 同上，对来自不同的map输出同一个KEY的数据累加 |
| 新老客户 |
| 回访周期 |
| 入店时长 |
|  | |  |  |  |

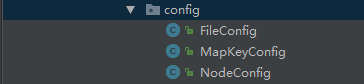
Map reduce的统计结果会被存入HDFS文件系统中。

* + 1. mapreduce包设计



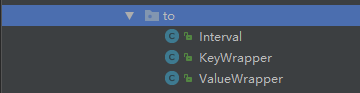
* + - * **common 公共依赖的代码**
      * config 配置类

HDFS各个文件夹路径配置，字符串常量，服务器节点配置



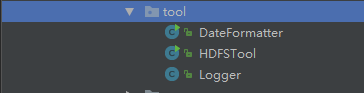
* + - * TO 传输对象封装

to 实现了Writable与cloneable的value包装类，实现了ComparableWritable的key包装类。方便的自定义包装各种可作为KEYVALUE使用的类。



* + - * tool 常用工具类

如文件读写，日志打印，日期格式转换等。

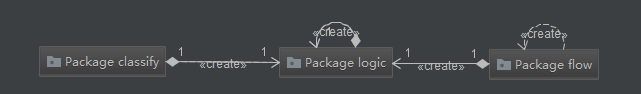


* + - * VO value object

主要是对应wifi探针上传的数据格式对象，以及分析结果的数据对象。

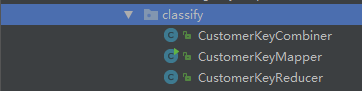
* + - * **整体分析与计算的包**

**主要有三个包，依赖关系如下**



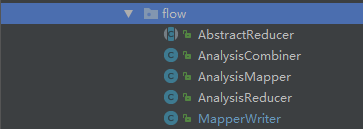
* + - * classify 安装用户mac地址分类数据--第一个mapreduce

也是第一个mapreduce的主要代码，用于根据mac地址分类的map-reduce程序，其中包括了Mapper、Reducer、Combiner三个类。

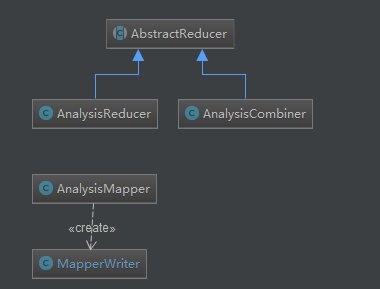


* + - * flow流式有序的扫描并分析数据--第二个mapreduce

也就是第二个mapreduce的主要代码，用于统计主要分析项。包括了Mapper、 Reducer、Combiner三个部分以及辅助类MapperWriter

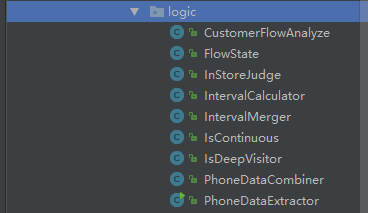


其依赖关系如下：



* + - * logic 复杂逻辑实现

主要是实现一些复杂的逻辑，是flow的辅助包



* 1. Web
     1. 处理用户请求

- 用户发起请求

- 界面将请求转发的web服务器

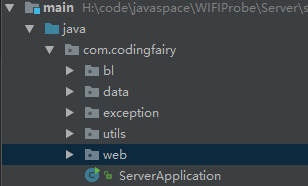
- web服务器收到请求

- web 服务器查询本地缓存是否有用户所需数据，有则返回

- web 服务器查询数据库是否有用户所需数据，有则返回

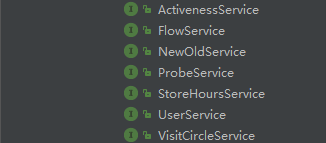
- web 服务器读取HDFS文件系统中的数据，并根据需要存入数据库与本地缓存中，都到后即返回。这里HDFS通过WEBHDFS以及Kerberos授权的方式提供远程调用。

* + 1. 主要实现分包



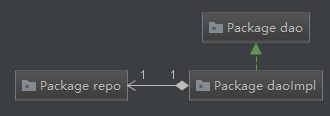
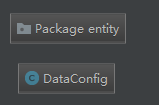
* + - * Bl：业务逻辑

这些业务逻辑基本都对应着前面提到的需求：如活跃度，用户量，新老客户，入店时长，访问周期等等。

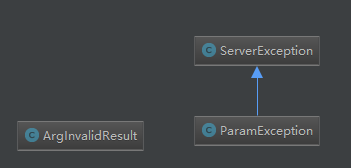


* + - * data：数据访问层以及持久化对象

包括数据库映射实体，数据库访问层，数据库配置对象以及协助类。

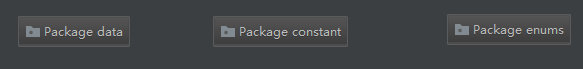
 

* + - * Exception: 所有的自定义异常封装。



* + - * Util: 工具类封装

公共工具类，常量，枚举类



* + - * Web: web controller

主要是路由控制，安全处理等。

* 1. Receiver

探针接收服务器以分布式部署，HTTP post的方式接收数据。

* + 1. 负载均衡

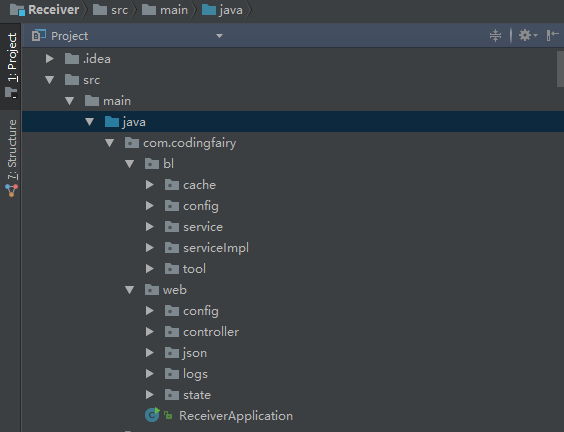
使用腾讯云负载均衡，每个探针想服务器发送的数据都会均匀的负载到各台服务器上。

* + 1. Session一致性

搭建分布式redis集群存储session信息，由于receiver服务器目前只接受wifi探针发送的数据，所以对session的一致性要求比较低。

* + 1. 数据存储

由于HDFS不适合频繁写、小文件存储，所以每个节点都会对本地数据缓存，达到一定规模后才会写入HDFS文件。

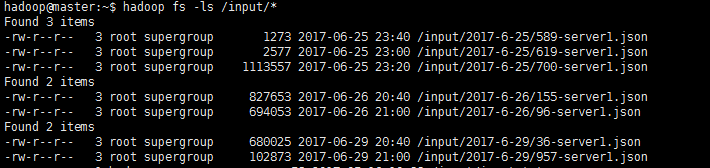


* + - * bl

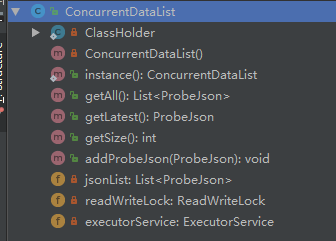
主要的业务逻辑，包括

Cache: 对每个wifi探针提交到该服务器上的数据都会缓存到一个线程安全的队列中。当这个队列足够大或者离上一次提交的时间超过20分钟时，则会被提交到Config包配置的HDFS文件系统中。目录与文件格式为 *日期/时间-receiver服务器名.json*

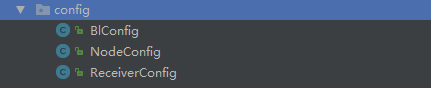
*如图：*



队列主要接口：

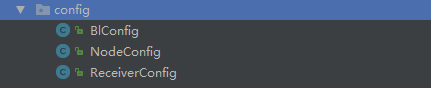


Config：依赖的其他模块配置，主要是hdfs WebHTTP接口地址。



Service: bl模块提供的接口，主要是对wifi探针提交到该服务器上的数据的具体处理，以及提交本地缓存数据至HDFS的接口。

ServiceImpl: service的具体实现。



* + - * web

http post数据的接收逻辑：

Config: 字符串常量的配置。

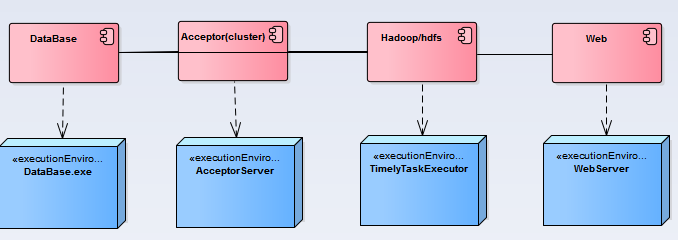
Controller：http访问路径的配置管理与转发，这里主要是wifi探针的提交路径，以及z 请求转发到业务逻辑中。

Json：java value object。主要是接收wifi探针数据字段。

Logs：日志工具类，对业务逻辑的主要方法切入log

State：服务器的心跳包发送，监控服务器健康

1. 部署视图



* 1. 部署方式

DataBase:mysql数据库单台

Receiver（Acceptor）:wifi探针数据接收服务器多台

Hadoop:多节点集群

WebServer：单台

* 1. 节点交互
     1. 用户与web服务器

标准的HTTP通信协议:对用户提供查看与监控的接口

* + 1. wifi探针与接收服务器

标准的HTTP通信协议：对wifi探针提供上传数据的接口

* + 1. Web服务器与HDFS

基于WEB-HDFS的HTTP：对web服务器提供读文件接口

* + 1. 接收服务器Receiver与HDFS

基于WEB-HDFS的HTTP：对接收服务器提供读、写文件接口