# 软件类设计文档

面向智慧工厂的准实时监管系统

项目组组号：NO.1

项目组负责人：周汉辰

联系电话：18621966896

电子邮箱：1341634255@qq.com

2016 年 11 月

# 文档信息

| 标题 | 面向智慧工厂的准实时监管系统软件类设计文档 |
| --- | --- |
| 作者 | 周汉辰、曹雨婷、张鹤腾 |
| 创建日期 | 2016/11/29 |
| 上次更新日期 | 2016/12/25 |
| 版本 | V2.1 |
| 组号 | NO.1 |

# 修改历史

| 日期 | 版本 | 说明 | 作者 |
| --- | --- | --- | --- |
| 2016/11/29 | V1.0 | 初稿 | 周汉辰、曹雨婷、张鹤腾 |
| 2016/12/21 | V1.1 | 修改类图 | 曹雨婷 |
| 2016/12/23 | V1.2 | 修改类间关系说明 | 张鹤腾 |
| 2016/12/24 | V2.0 | 增加配置管理模块，修改数据组装模块 | 曹雨婷 |
| 2016/12/25 | V2.1 | 修改 DataCore 模块 | 周汉辰 |

# 目录

# 一、 引言

## 1. 编制目的

本报告详细完成对面向智慧工厂的准实时监管系统的类设计，达到指导开发人员后续软件构造的目的，同时实现和测试人员的沟通。

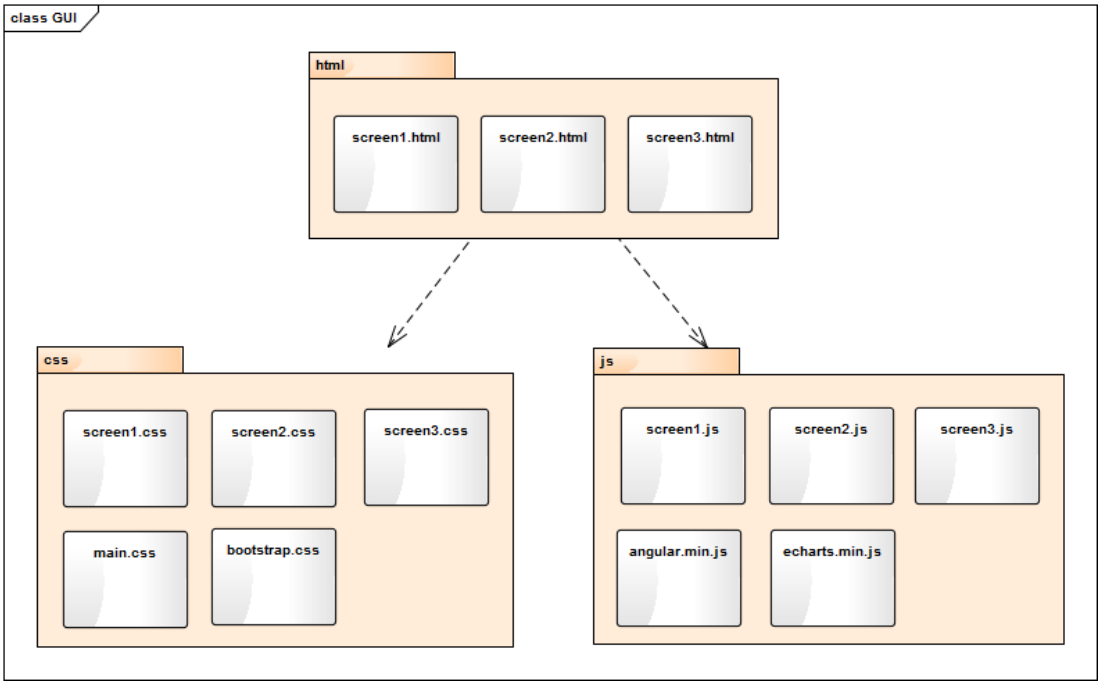
## 2. 产品概述

参考面向智慧工厂的准实时监管系统软件需求归约文档和软件架构文档说明中对产品的概括描述。

# 二、 类设计

## 1. 可视化模块

### 1.1 类图



## 1.2 类说明

| 包.类 | 职责 |
| --- | --- |
| html.screen1.html | 第一屏的界面呈现，显示订单数据和客服数据 |
| html.screen2.html | 第二屏的界面呈现，显示智能吊挂系统、自动裁床系统以及其他设备的监控数据 |
| html.screen3.html | 第三屏的界面呈现，显示智能排程系统、智能生产控制系统和供应链的监控数据 |
| js.screen1.js | 第一屏的控制器，负责向后台发送 Http 请求，处理后台返回的数据格式，以供界面显示 |
| js.screen2.js | 第二屏的控制器，负责向后台发送 Http 请求，处理后台返回的数据格式，以供界面显示 |
| js.screen3.js | 第三屏的控制器，负责向后台发送 Http 请求，处理后台返回的数据格式，以供界面显示 |
| js.angular.min.js | angularJS 库文件 |
| js.echarts.min.js | Echarts 图表插件文件 |
| css.screen1.css | 第一屏的层叠样式表，用来修饰第一屏界面 |
| css.screen2.css | 第二屏的层叠样式表，用来修饰第二屏界面 |
| css.screen3.css | 第三屏的层叠样式表，用来修饰第三屏界面 |
| css.main.css | 所有界面的通用层叠样式表，定义了界面整体风格 |
| css.bootstrap.css | bootstrap 库文件 |

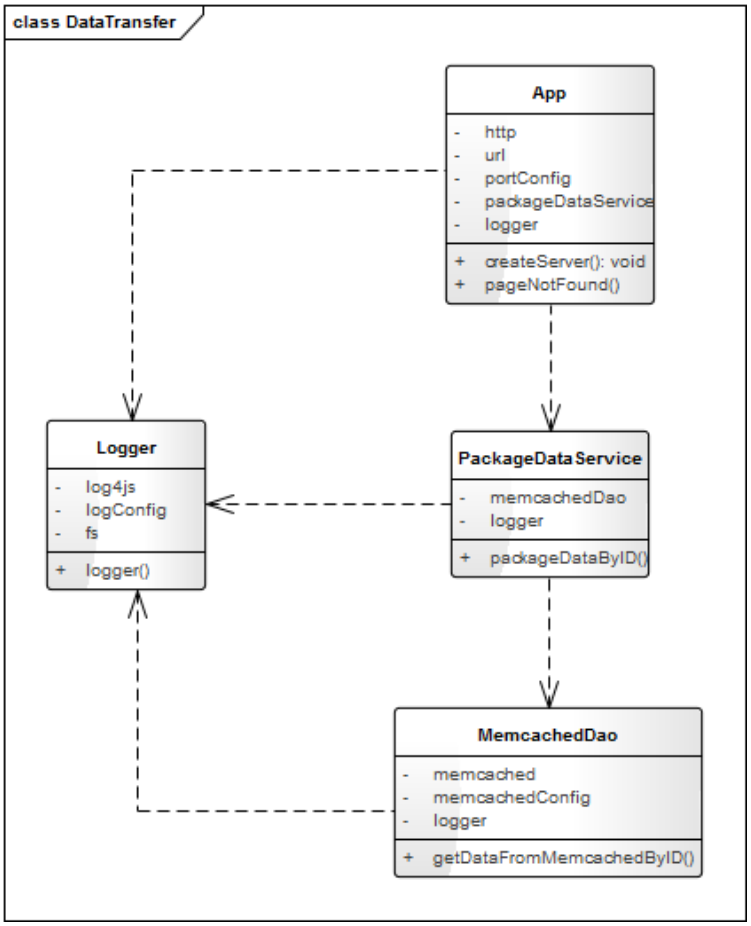
## 1.3 类间关系

GUI 包中包含 html、js 和 css 三个包。根据需求，将数据分三个屏幕显示，每一屏幕对应一个 html 文件，即 html 包中包含三个文件：screen1.html，screen2.html，screen3.html。每个屏幕分配一个 AngularJS 控制器，即对应一个 js 文件，即 js 包中的 screen1.js，screen2.js，screen3.js。js 包中还包含使用到的 angularJS 库文件和 Echarts 图表插件文件。css 包中包含定义了界面整体风格的 main.css 文件和每一屏幕对应的层叠样式表，以及使用到的 bootstrap 库文件。

html 包中的文件依赖相对应的 js 和 css 两个包中的文件以及 js 包中的 angular.min.js 文件和 echarts.min.js 文件，css 包中的 main.css 文件和 bootstrap.css 文件。

## 2. 数据推送前置模块

### 2.1 类图




### 2.2 类说明

| 包.类 | 职责 |
| --- | --- |
| App | 用于开启服务端口 |
| PackageDataService | 用于将获取的数据包装成前端需要的数据结构 |
| MemcachedDAO | 用于从 MemCached 中获取数据 |
| Logger | 用于输出日志 |

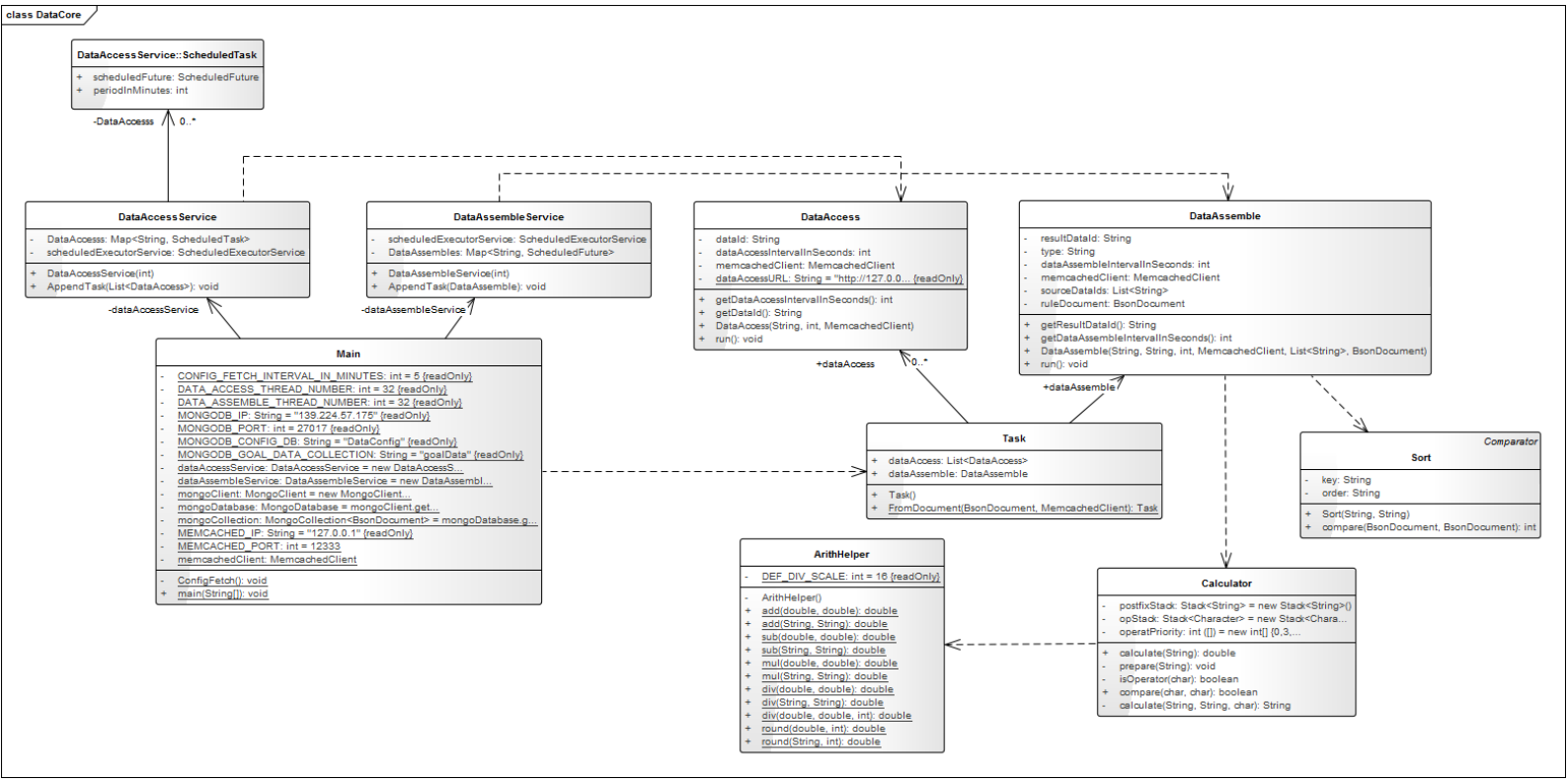
### 2.3 类间关系

DataTransfer 模块专门用于与前端进行通信。它是整个项目后台的入口。选择 NodeJS 作为运行环境，使用 MVC 架构编写。当数据请求到来时，App 将获得消息内容，经过解析之后，调用 PackageDataService 的 packageDataByID()

方法，向 PackageDataService 请求目标数据，即 App 依赖 PackageDataService。PackageDataService 调用 MemcachedDAO 的 getDataFromMemcachedByID()方法从内存数据库中获取相应数据，即 PackageDataService 依赖 MemcachedDAO。PackageDataService 获得目标数据后将数据包装成前端需要的形式后，数据通过 App 返回给前端。另外，App、PackageDataService 和 MemcachedDAO 都依赖 Logger 输出日志。

## 3. 数据组装模块

### 3.1 类图




### 3.2 类说明

| 包.类 | 职责 |
| --- | --- |
| Main | 程序入口，读取配置数据 |
| Task | 负责解析数据配置 |
| DataAccess | 负责与数据获取模块通信，请求获取源数据 |
| DataAssemble | 负责目标数据的组装 |
| DataAccessService | 负责管理数据获取线程 |
| DataAssembleService | 负责管理数据组装线程 |
| Sort | 负责排序功能 |

| | |
|---|---|
| ArithHelper | 负责计算功能 |
| Calculator | 负责字符串表达式解析功能 |

### 3.3 类间关系

Main 按照一定频率读取配置数据，将最新的配置发送至 Task。它拥有 DataAccessService 和 DataAssembleSerice 类型的成员变量，即 Main 单向关联 DataAccessService 和 DataAssembleSerice。Main 使用 Task 类型的局部变量作为参数调用 DataAccessService 和 DataAssembleSerice 维护和管理数据读取和组装任务，Main 依赖 Task。

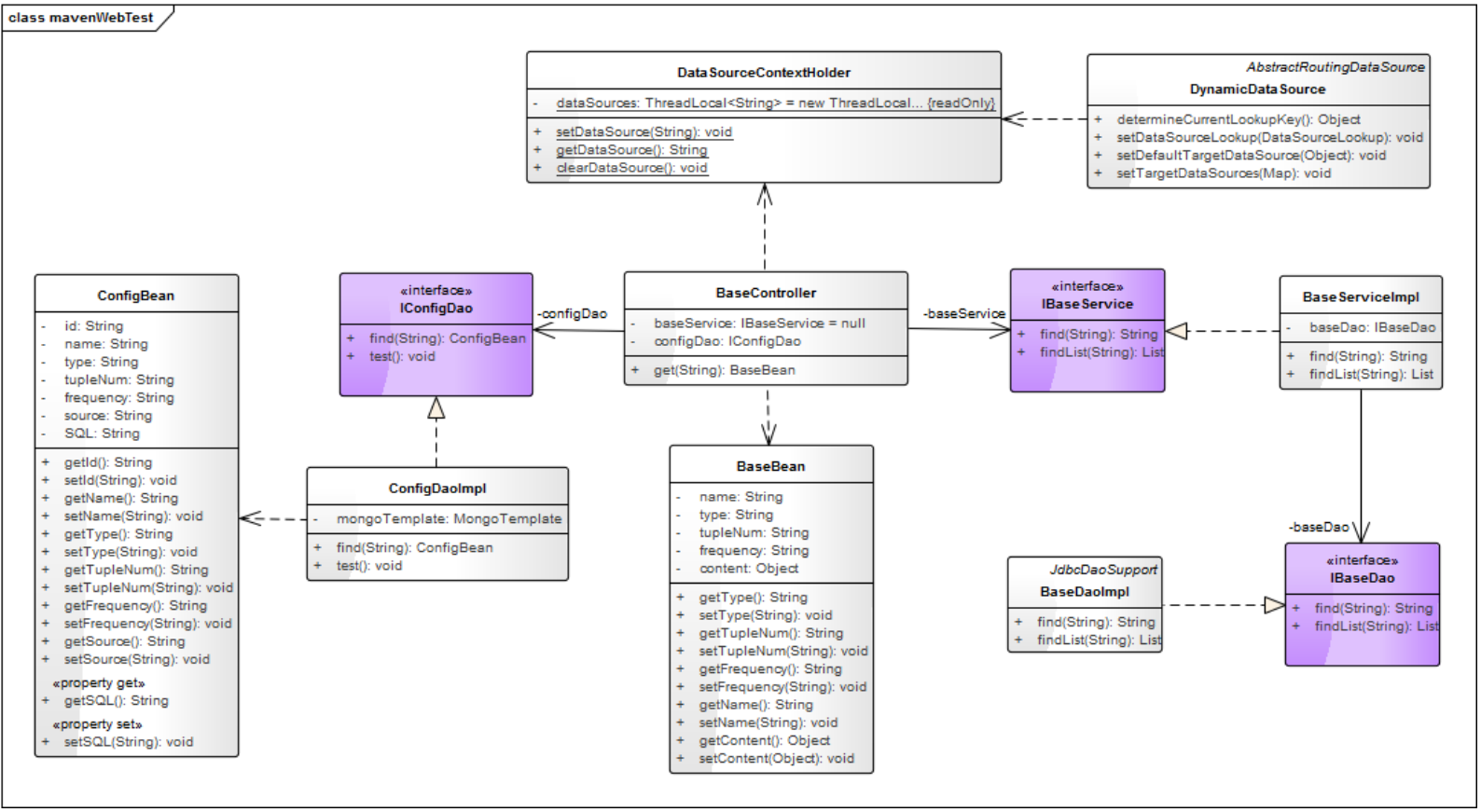
Task 负责解析数据配置，拥有 DataAccess 和 DataAssemble 类型的成员变量，即 Task 单向关联 DataAccess 和 DataAssemble。DataAccess 负责与数据获取模块通信，请求获取源数据。DataAssemble 负责目标数据的组装。

DataAccessService 管理数据读取任务时使用 DataAccess 类型的参数，即 DataAccessService 依赖 DataAccess。DataAccessService 使用 ScheduledTask 类型的成员变量来维护调度任务，即 DataAccessService 单向关联 ScheduledTask。DataAssembleSerice 管理数据组装任务时使用 DataAssemble 类型的参数，即 DataAssembleSerice 依赖 DataAssemble。

DataAssemble 在组装数据过程中将根据规则，要么直接包装源数据、要么对数据进行排序、要么对源数据做表达式计算，当需要对数据进行排序时，将使用到 Sort，当对源数据进行表达式计算时，将使用到 Calculator，Calculator 将字符串表达式进行解析，然后使用 ArithHelper 计算出结果。

# 4. 数据获取模块

## 4.1 类图




## 4.2 类说明

| 包.类 | 职责 |
| --- | --- |
| BaseBean | 源数据对象 |
| ConfigBean | 源数据配置对象 |
| BaseController | 开启监听请求服务 |
| IBaseService | 源数据获取服务接口 |
| BaseServiceImpl | 源数据获取服务实现 |
| IBaseDao | 源数据获取接口 |
| BaseDaoImpl | 源数据获取实现 |
| IConfigDao | 源数据配置获取接口 |
| ConfigDaoImpl | 源数据获取配置实现 |
| DynamicDataSource | 动态管理数据源 |
| DataSourceContextHolder | 获取数据源上下文 |

## 4.3 类间关系

该模块是一个使用 SpringMVC 架构的 web 应用，通过 BaseController 获取来自数据组装模块的 Http 请求，Http 请求中包括源数据名称。BaseController 拥有两个成员变量，分别为 IBaseService 类型和 IConfigDao 类型，即 BaseController 和 IBaseService、IConfigDao 单向关联。BaseController 以 BaseBean 的形式返回数据，即依赖 BaseBean。

IConfigDao 从 Mongodb 中获取源数据配置信息,以 ConfigBean 的形式返回。ConfigDaoImpl 实现了 IConfigDao 接口，依赖 ConfigBean。

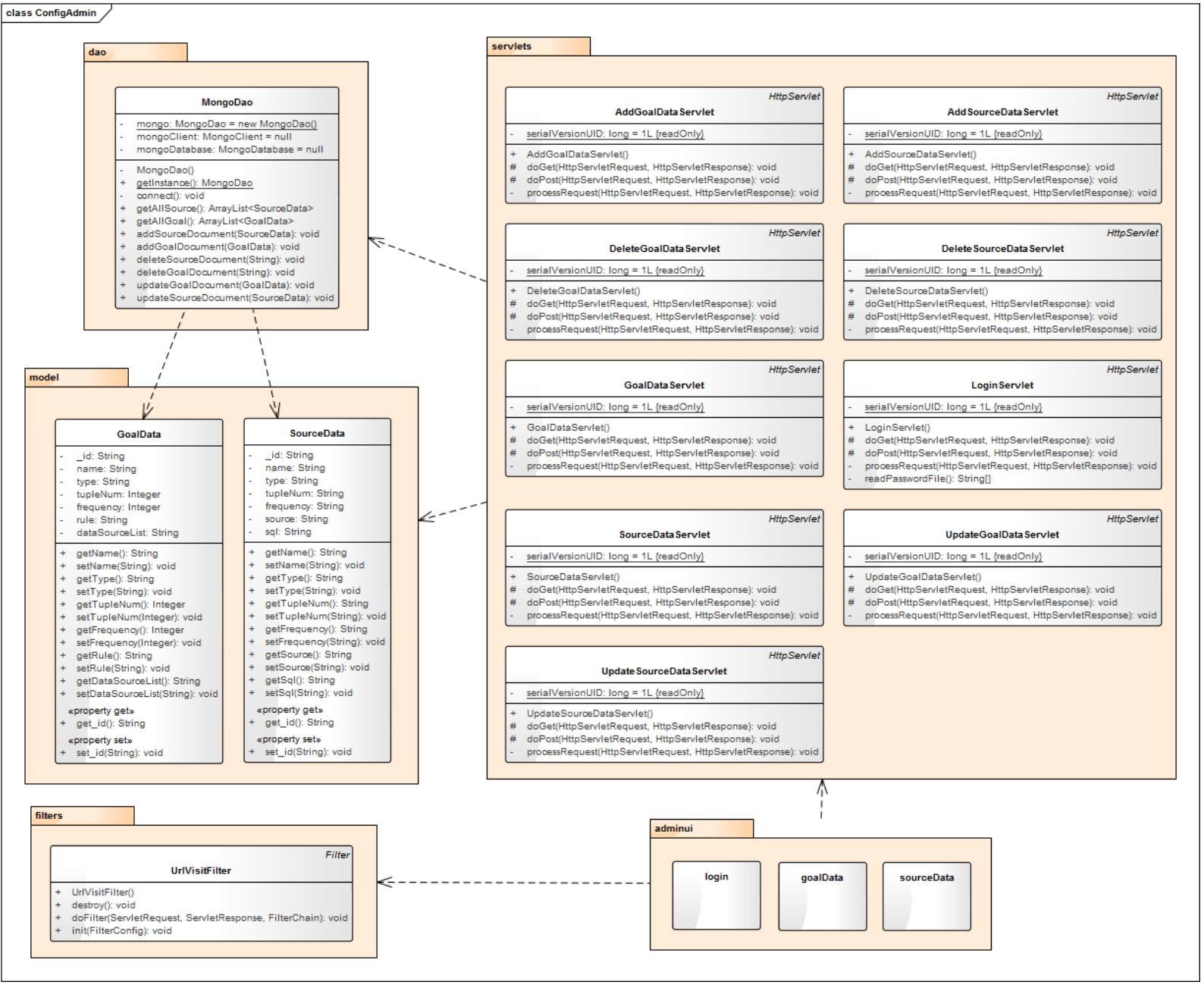
得到了源数据的配置信息之后，BaseController 根据配置信息中的 `source`使用 DataSourceContextHolder 的静态方法设置数据源，即 BaseController 依赖 DataSourceContextHolder。

DynamicDataSource 重写了 determineCurrentLookupKey()方法，能够根据数据库标示符取得当前的数据库。其中使用了 DataSourceContextHolder 的静态方法 getDataSource()，故 DynamicDataSource 依赖 DataSourceContextHolder。

BaseServiceImpl 实现了 IBaseService 接口。BaseServiceImpl 调用 IBaseDao 类型的成员变量的方法从源数据库中查询源数据，即单向关联 IBaseDao。BaseDaoImpl 实现了 IBaseDao 接口。

# 5. 配置管理模块

## 5.1 类图




## 5.2 类说明

| 包.类 | 职责 |
| --- | --- |
| MongoDao | 与 MongoDB 交互的 Dao |

| | |
|---|---|
| GoalData | 目标数据的整合类 |
| SourceData | 源数据的整合类 |
| UrlVisitFilter | 登录控制过滤器类 |
| AddGoalDataServlet | 增加目标数据 Servlet |
| AddSourceDataServlet | 增加源数据 Servlet |
| DeleteGoalDataServlet | 删除目标数据 Servlet |
| DeleteSourceDataServlet | 删除源数据 Servlet |
| GoalDataServlet | 查找目标数据 Servlet |
| LoginServlet | 登录 Servlet |
| SourceDataServlet | 查找源数据 Servlet |
| UpdateGoalDataServlet | 更新目标数据 Servlet |
| UpdateSourceDataServlet | 更新源数据 Servlet |
| login | 登录 jsp |
| goalData | 目标数据 jsp |
| sourceData | 源数据 jsp |

### 5.3 类间关系

该模块是一个 MVC 架构的 web 平台，实现对 MongoDB 中源数据和目标数据的配置管理信息的增删改查。用户登录页面，通过 JSP 向服务器端发送 Servlet 请求，Servlet 调用 MongoDao，因此 servlets 依赖 dao，与 MongoDB 进行交互，SourceData 和 GoalData 分别为源数据和目标数据的整合类，被 Servlet 和 MongoDao 调用，因此存在依赖关系。

View 层即各页面，JSP 页面获取用户输入的数据，传递给 Servlet 进行处理，因此 adminui 依赖 servlets，再将结果返回到 JSP 页面呈现；Model 层包括 SourceData 和 GoalData，完成对数据库的映射；Controller 包括 MongoDao 和各 Servlet，负责业务管理，Servlet 继承自 HttpServlet，隐藏了实现细节，将开发任务集中在业务实现上。

在 JSP 页面，实现了对插入数据的格式检查，以确保数据的正确性和可用性。且增加登录过滤器，实现登录后允许操作，因此 adminui 依赖 filters。