新致Manager Cloud框架

构建指南

**更新记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **更新日期** | **说明** | **更新人** |
| v0.1 | 2017-07-20 | 草稿 | 余磊 |
| v0.2 | 2017-09-26 | 增加前端(web)框架相关说明，增加多租户、日志、定时任务配置说明 | 余磊 |
| v0.3 | 2017-10-31 | 增加调试命令以及配置方式；调整MC前端框架代码组织 | 余磊 |
| v0.4 | 2017-11-01 | 增加开发环境准备 | 余磊 |
| v0.5 | 2017-11-03 | 整理内容，增加代码结构、前端框架、后端框架的说明 | 余磊 |
|  |  |  |  |
|  |  |  |  |

**目录**

[1. 前言 4](#_Toc497577255)

[2. 概述 4](#_Toc497577256)

[2.1. 架构目标 4](#_Toc497577257)

[2.2. 架构特点 4](#_Toc497577258)

[2.3. 组成 5](#_Toc497577259)

[2.4. 交付清单 5](#_Toc497577260)

[3. 代码结构 6](#_Toc497577261)

[3.1. 总体目录结构 6](#_Toc497577262)

[3.2. 后端框架目录 6](#_Toc497577263)

[3.3. 前端框架目录 7](#_Toc497577264)

[4. 后端框架 9](#_Toc497577265)

[4.1. 外部组件 9](#_Toc497577266)

[4.2. MC组件 10](#_Toc497577267)

[4.3. 架构说明 10](#_Toc497577268)

[5. 前端框架 10](#_Toc497577269)

[5.1. 外部组件 11](#_Toc497577270)

[5.2. 架构说明 11](#_Toc497577271)

[6. 应用配置说明 12](#_Toc497577272)

[6.1. 持久化容器配置 13](#_Toc497577273)

[6.2. 数据源配置 13](#_Toc497577274)

[6.3. 多租户（动态数据源） 14](#_Toc497577275)

[6.3.1. 启用多租户模式 14](#_Toc497577276)

[6.3.2. 数据源配置 16](#_Toc497577277)

[6.4. 日志配置 18](#_Toc497577278)

[6.5. 定时任务 18](#_Toc497577279)

[6.6. Hibernate实体配置 18](#_Toc497577280)

[7. 项目初始化 18](#_Toc497577281)

[7.1. 配置项 18](#_Toc497577282)

[7.2. 新模块代码目录 19](#_Toc497577283)

[8. 集成说明 20](#_Toc497577284)

# 前言

本文描述了Manager Cloud框架（简称为MC框架，下同）的架构目标、环境、配置、部署等方面的内容。

本文供项目技术负责人和开发者阅读。

项目技术负责人可以参考本文，借助MC框架快速搭建工程项目。

开发者可参考本文，快速掌握MC框架项目开发方法。

# 概述

## 架构目标

MC框架注重于开发企业信息管理系统（简称为mis，Management Information System）或后端管理系统，为类似项目开发提供了一整套项目解决方案。MC框架不适用于开发偏向于互联网的项目。

## 架构特点

MC框架在架构搭建过程中，充分考虑了面临解决的问题、团队的技术的延续性、框架的扩展性等多个方面的问题。最终：

* B/S结构

便于使用，符合业界主流使用习惯。

* 跨平台部署

后端采用j2ee开发，因此可使用各类主流的web容器部署，如Tomat、Weblogic、WAS等。

* 支持多种数据库

MC框架后端核心代码采用ORM模式开发，可无缝切换到各类主流数据库，包括Oracle、MySQL、SQLServer等。

* 支持多租户模式

以此支持一个应用连接多个数据库，可用来作为公司新致云产品开发框架。

* 完整的web界面解决方案

拥有一整套完整的web界面开发解决方案，包括风格一致的组件、常见布局，以及代码样例。使MC框架拥有Web界面的快速开发能力。

* 较低的学习成本

考虑到部门团队的技术领域、技术延续性，以及预计的使用范围。未采用目前主流的前端框架，采用jQuery为基础进行前端架构搭建，并配合较完整的代码样例，来降低学习转型成本。

* 可扩展性强

MC架构采用了业界主流的的三方组件搭建，如Spring、Hibernate、jQuery等，因此可以方便借助这些主流技术生态进行扩展。

## 组成

MC框架由以下几部分组成：

* MC后端框架
* MC前端框架

## 交付清单

MC框架交付清单包括：

|  |  |  |
| --- | --- | --- |
| **序号** | **交付物** | **说明** |
| 1 | mc框架初始可运行war包 | 一个开箱即用的war包 |
| 2 | mc框架初始数据库 | 仅包含必要的初始化数据。 |
| 3 | 《新致MC框架-构建指南》 |  |
| 4 | 《新致MC框架-开发指南》 |  |
| 5 | 相关数据库说明文档 |  |
| 6 | 工作流开发和设置文档 |  |

# 代码结构

MC框架初始项目，参照Maven标准建立，并使用maven插件管理。但在多java目录这一点上并未完全照搬Maven标准。

## 总体目录结构

项目主要代码目录如下：

|  |  |  |
| --- | --- | --- |
| **序号** | **目录** | **说明** |
| 1 | / | 项目代码根目录 |
| 2 | /pom.xml | 项目配置文件，供maven解析。 |
| 3 | /src | 项目源代码根目录 |
| 4 | /src/main/java/ | 项目java源代码根目录 |
| 5 | /src/main/platform/ | mc框架java源代码根目录 |
| 6 | /src/main/resources/ | 项目资源根目录，编译打包时会自动复制到targer/classes目录下。包括各类java配置文件 |
| 7 | /src/main/webapp/ | 项目web应用根目录，包含WEB-INF、MATE-INF等。 |
| 8 | /src/test/java/ | 单元测试java源代码目录 |
| 9 | /src/test/resources/ | 单元测试资源目录 |
| 10 | /target | 项目输出目录。无需建立，maven插件编译时自动生成。 |

## 后端框架目录

灰色项目为框架提供公用代码，一般不需要修改。

|  |  |  |
| --- | --- | --- |
| **序号** | **目录** | **说明** |
| **后端目录** | | |
| 1 | /src/main/java/ | 项目java源代码根目录 |
| 2 | /src/main/platform/ | mc框架java源代码根目录。包括登录页、首页等。 |
| 3 | /src/main/demo/ | 后端样例代码(demo) |
| 4 | /src/main/resources/ | 项目资源根目录，编译打包时会自动复制到targer/classes目录下。包括各类java配置文件。 |
| 5 | /src/main/resources/spring/ | spring配置文件目录。包括springmvc启动配置文件spring.xml |
| 6 | /src/main/resources/persistence/ | 持久化配置文件目录。包括数据库连接配置文件proxool.xml |
| 7 | /src/main/resources/hbm/ | 持久化配置的实体配置文件目录。 |
| 8 | /src/main/resources/workflow/ | 工作流引擎相关配置文件。 |
| 9 | /src/main/resources/scripts/ | 工作流引擎读取脚本目录。一般不修改。 |
| 10 | /src/main/resources/jetty/ | 自带jetty启动插件环境配置。一般不修改。 |
| 11 | /src/test/java/ | 单元测试java源代码目录 |
| 12 | /src/test/resources/ | 单元测试资源目录 |

## 前端框架目录

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **目录** | **说明** | **读写控制** |
| **前端目录** | | | |
| 15 | /src/main/webapp/ | 项目web应用根目录，包含WEB-INF、MATE-INF等。 |  |
| 16 | /src/main/webapp/platform/ | mc框架web源代码根目录。包括登录页、首页等。 |  |
| 17 | /src/main/webapp/jquery/ | jQuery核心库、jQuery插件、依赖于jQuery的UI组件 | 不要修改 |
| 18 | /src/main/webapp/thridparty/ | 独立第三方js库 | 不要修改 |
| 19 | /src/main/webapp/mc/ | MC前端框架。 | MC框架提供，一般不修改。 |
| 20 | /src/main/webapp/mc/common/ | mc核心库s、组件、引用等 | MC框架提供，一般不修改。 |
| 21 | /src/main/webapp/mc/sm/ | 公用系统管理功能。 | MC框架提供，一般不修改。 |
| 22 | /src/main/webapp/mc/wf/ | 工作项管理和表单审批 | MC框架提供，一般不修改。 |
| 23 | /src/main/webapp/mc/demo/ | MC前端框架样例代码(demo) | MC框架提供，无需修改 |
| 24 | /src/main/webapp/ext/ | ExtJS库。用于老项目兼容。 | MC框架提供，一般不修改。 |
| 25 | /src/main/webapp/cloud/ | 基于ExtJS的前端开发框架。用于老项目兼容。 | MC框架提供，一般不修改。 |
| 26 | /src/main/webapp/WEB-INF/web.xml | j2ee项目配置文件，包括欢迎页面、静态资源访问绕过spring拦截器、session失效时间、编码配置、proxool初始化、spring初始化 |  |
| 27 | /src/main/webapp/resources/ | web前端资源目录，包括css、图片、国际化资源等 |  |
| 28 | /src/main/webapp/resources/themes/ | web前端皮肤 |  |
| 29 | /src/main/webapp/resources/themes/default/ | web前端默认皮肤 | MC框架提供，一般不修改。 |
| 30 | /src/main/webapp/resources/themes/default/css/ | web前端默认皮肤 | MC框架提供，一般不修改。 |
| 31 | /src/main/webapp/resources/themes/default/images/ | web前端默认皮肤 | MC框架提供，一般不修改。 |
| 32 | /src/main/webapp/resources/themes/default/js/ | web前端默认皮肤 | MC框架提供，一般不修改。 |

# 后端框架

## 外部组件

项目中使用到的三方组件包括：

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **组件** | **版本** | **说明** |
| 1 | Spring | 4.3.7.RELEASE |  |
| 2 | Hibernate | 5.1.5.Final | 数据访问和持久化框架 |
| 3 | proxool | 0.9.1 | 数据库连接池 |
| 4 | poi | 3.7 | office访问组件 |
| 5 | jasperreports | 5.6 | 报表组件 |
|  |  |  |  |

## MC组件

有部分MC公用组件以Maven依赖引用方式引入项目。

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **组件** | **版本** | **说明** |
| 1 | newtouch-cloud-commons |  | 公用类组件 |
| 2 | newtouch-cloud-security |  | 安全策略组件，提供登录认证、密码的安全策略实现 |
| 3 | newtouch-cloud-sm |  | 系统管理组件。提供公司、部门、用户、角色、权限等管理功能 |
| 4 | newtouch-cloud-bcm |  | 预算控制组件。提供预算项目、责任中心、预算数的维护和预算控制核心功能。 |
| 5 | newtouch-workflow-core |  | 工作流引擎。 |
| 6 | newtouch-workflow-service |  | 工作流管理服务功能。Web界面使用的服务都在此包里 |
|  |  |  |  |

## 架构说明

MC框架开发框架，后端主要基于spring + springMVC + Hibernate构建。action层取消了struts，改用springMVC组件。

第三方组件使用Maven方式管理依赖项。

# 前端框架

前端开发框架包含

* 一套经过选择和匹配的第三方类库，包含jQuery以及组件
* 一套界面组件。
* 一套界面样例，包括UI组件、界面布局等。
* 功能开发的最佳实践。

## 外部组件

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **组件** | **版本号** | **说明** |
| 1 | Java Script | ES5 | 浏览器自带 |
| 2 | jQuery | v1.12.4 | 基础js类库 |
| 3 | jQuery-UI | v1.12.1 | 提供widget组件机制；日期选择(datepaicker)；下拉框(selectmenu) |
| 4 | jQGrid | v5.2.1 | 表格 |
| 5 | layer | v3.1.0 | 弹出对话框（弹出层）、提示框、遮罩 |
| 6 | layout | v1.4.4 | border布局 |
| 7 | zTree | 待定 | 树 |
| 8 | layui | 2.1.5 | 首页布局。提供了Tab标签页。 |
| 9 | alert-dialogs | 1.1 | 提示框。备用。 |
| 10 | blockui |  | 遮罩。备用 |
|  |  |  |  |

## 架构说明

前端取消了Ext组件库，改用JQuery作为核心包，围绕JQuery建立界面UI体系。界面UI组件主要使用Html原生 + CSS样式 + JQueryUI +　其他JQuery组件组成，并加入了成体系的CSS风格，达到统一、易更换风格的UI布局方式。其中Html负责界面元素布局；CSS负责页面风格渲染；js + JQuery执行操作。

前端开发框架类库包含以下部分

* mc核心js：包管理、工具类
* jQuery v1.12
* 基于jQuery的插件和三方UI组件
* 自定义的jQueryUI组件

其依赖关系如下：



# 应用配置说明

配置文件如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **路径** | **文件** | **说明** |
| 1 | resources/spring/ | spring.xml | spring初始化入口，引用所有spring配置文件 |
| 2 | resources/spring/ | spring-context.xml | spring公用配置。包括注解扫描、自动装配、拦截器、转换器等 |
| 3 | resources/spring/ | spring-ds.xml | 数据源定义 |
| 4 | resources/spring/ | spring-persistence-em.xml | 持久化容器定义，使用JPA标准的EntityManager管理 |
| 5 | resources/spring/ | spring-persistence-sf.xml | 持久化容器定义，使用Hibernate的SessionFactory管理 |
| 6 | resources/spring/ | spring-view-jasperreports.xml | jasperreports报表视图定义 |
| 7 | resources/persistence/ | proxool.xml | 数据库连接配置 |
| 8 | resources/workflow/ | spring-context-workflow.xml | workflow工作流配置入口 |
| 9 | resources/ | logback.xml | logback日志配置 |
| 10 | resources/ | WfDBConfig.properties | 工作流数据源配置 |

## 持久化容器配置

* Mcloud框架使用Hibernate作为持久化组件。但是具体的持久化容器实现类，可以选择使用符合JPA标准的EntityManager，或Hibernate自有的SessionFactory。但不能两者同时使用。
* 不管使用哪一种持久化容器，底层都是使用了Hibernate的SessionFactory，所以都可以使用Session相关的高级特性。
* 具体配置方法上，在resources/spring/spring.xml入口中，要么引用spring-persistence-em.xml，要么引用spring-persistence-sf.xml，不能两者都引用。

<!-- 使用JPA标准的EntityManager作为持久化容器 -->

<import resource=*"classpath:/spring/spring-persistence-em.xml"* />

<!-- 使用Hibernate.SessionFactory作为持久化容器 -->

<!--

<import resource="classpath:/spring/spring-persistence-sf.xml" />

-->

* 数据库连接池组件必须使用proxool。因为目前工作流引擎内部限制，必须用该组件。后续可能会改进。

## 数据源配置

一般情况下，数据库连接参数配置在resources/persistence/proxool.xml文件中。Spring数据源配置在resources/spring/spring-ds.xml文件中。

完整配置方式为：

spring上下文 => 持久化容器（EntityManager/SessionFactory） => spring数据源（DriverManagerDataSource） => 数据库连接池组件（proxool）=> 数据库（Oracle/MySQL/SQLServer）

目前由于工作流引擎限制，只能使用proxool作为连接池组件。

多租户下数据源连接配置，请参见多租户配置相关章节。

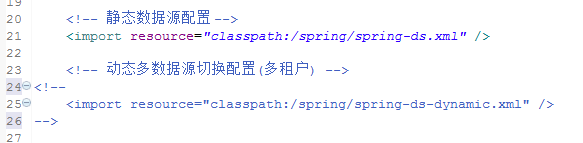
## 多租户（动态数据源）

* 支持多租户（即一个Web应用对应多个数据库）。具体配置需要使用spring的动态数据源。

### 启用多租户模式

* 默认情况下使用单数据源配置，如果需要切换到多租户配置，需要做一下配置：

1. 在resources/spring/spring.xml中，注释掉“静态数据源配置”，启用“动态多数据源配置”



1. 在resources/spring/spring-context.xml中，修改拦截器设置，把红框注释掉的这个动态多数据源奇幻拦截器启用起来（com.newtouch.cloud.platform.interceptor.DynamicDataSourceInterceptor）。

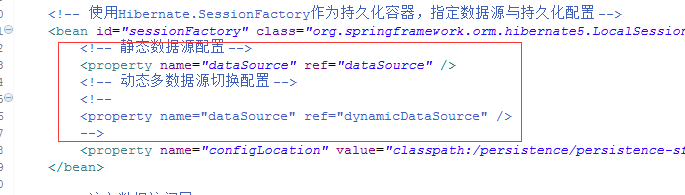


* 在持久化容器配置的dataSouce，改用使用动态切换数据源dynamicDataSource

如果使用JPA.EntityManager作为持久化容器，那么修改resources/spring/spring-persistence-em.xml



如果使用HibernateSession作为持久化容器，那么修改resources/spring/spring-persistence-sf.xml



### 数据源配置

多租户模式下，数据库连接参数配置在resources/persistence/proxool.xml文件中。Spring数据源配置在resources/spring/spring-ds-dynamic.xml文件中。

完整配置方式为：

spring上下文 => 持久化容器（EntityManager/SessionFactory） => spring数据源（DynamicDataSource） => 数据库连接池组件（proxool）=> 数据库（Oracle/MySQL/SQLServer）

* 数据库连接池配置文件（proxool.xml），需要配置多个连接池，每个对应一个数据库或数据库用户。（可参考resources/persistence/proxool-dynamic.sample）



* 动态数据源配置文件（spring-ds-dynamic.xml），需要对每个数据库连接池配置一个数据源，然后再组合成为一个动态数据源。



动态数据源，列出每个可以切换的数据源。



这样在后台程序内可以随意切换数据库。

## 日志配置

默认采用logback作为日志组件。

日志配置文件：resources/logback.xml

## 定时任务

默认采用Quartz作为定时任务组件

定时任务配置文件：resources/quartz.xml

## Hibernate实体配置

实体配置文件一般存放在：resources/hbm目录下。

可以在持久化容器配置中，直接扫描该目录

# 项目初始化

当要把MC框架应用到具体项目上时，需要做调整。

## 配置项

**敬请期待**

* web.xml

欢迎页

* pom.xml

groupId

artifactId

name

打包方式

打包名称

* 确定数据库类型
* 多租户或单租户
* 使用EntityManager或Hiberate SessionFactory
* 划分模块和目录

后端

后端配置

spring-content

hbm

log

quartz

前端

前端资源

## 新模块代码目录

每一个模块大致按照以下方式组织代码。新模块建议也采用此类组织方式，各模块之间降低耦合。

|  |  |  |
| --- | --- | --- |
| **序号** | **目录** | **说明** |
| **后端** | | |
| 1 | src/main/jara/com/newtouch/$module/ | xxx业务模块java代码 |
| 2 | src/main/jara/com/newtouch/$module/common/ | xxx业务模块公用组件 |
| 3 | src/main/jara/com/newtouch/$module/$submodule1/ | xxx业务模块/子模块1 |
| 4 | src/main/jara/com/newtouch/$module/$submodule1/action/ | xxx业务模块/子模块1/action层 |
| 5 | src/main/jara/com/newtouch/$module/$submodule1/bp/ | xxx业务模块/子模块1/业务处理层 |
| 6 | src/main/jara/com/newtouch/$module/$submodule1/dao/ | xxx业务模块/子模块1/数据访问层 |
| 7 | src/main/jara/com/newtouch/$module/$submodule1/entity/ | xxx业务模块/子模块1/实体类 |
| 8 | src/main/jara/com/newtouch/$module/$submodule2/ | xxx业务模块/子模块2 |
| 9 | src/main/jara/com/newtouch/$module/$submodule2/action/ | xxx业务模块/子模块2/action层 |
| 10 | src/main/jara/com/newtouch/$module/$submodule2/bp/ | xxx业务模块/子模块2/业务处理层 |
| 11 | src/main/jara/com/newtouch/$module/$submodule2/dao/ | xxx业务模块/子模块2/数据访问层 |
| 12 | src/main/jara/com/newtouch/$module/$submodule2/entity/ | xxx业务模块/子模块2/实体类 |
| **前端** | | |
| 13 | src/main/webapp/$module/ | xxx业务模块web功能代码 |
| 14 | src/main/webapp/$module/common/ | xxx业务模块公用js |
| 15 | src/main/webapp/$module/component/ | xxx业务模块公用组件 |
| 16 | src/main/webapp/$module/resources/ | xxx业务模块资源，包括css、图片、国际化资源等 |
| 17 | src/main/webapp/$module/$submodule1/ | xxx业务模块/子模块1 |
| 18 | src/main/webapp/$module/$submodule1/$fun1/ | xxx业务模块/子模块1/功能1 |
| 19 | src/main/webapp/$module/$submodule1/$fun2/ | xxx业务模块/子模块1/功能2 |
| 20 | src/main/webapp/$module/$submodule2/ | xxx业务模块/子模块2 |
| 21 | src/main/webapp/$module/$submodule2/$fun1/ | xxx业务模块/子模块2/功能1 |
| 22 | src/main/webapp/$module/$submodule2/$fun2/ | xxx业务模块/子模块2/功能2 |

Web前端公用引用

# 集成说明

支持集成到其他使用spring4.x的框架中。