构架开发手册和规范

|  |  |  |
| --- | --- | --- |
| 1.0 | 许俊飞 | 2017-12-27 |
|  |  |  |

目录

[操作手册 2](#_Toc502063815)

[部署项目 2](#_Toc502063816)

[开发流程 4](#_Toc502063817)

[JS插件 6](#_Toc502063818)

[Ajax 6](#_Toc502063819)

[上传 6](#_Toc502063820)

[城市选择 6](#_Toc502063821)

[时间选择 6](#_Toc502063822)

[树 7](#_Toc502063823)

[DataTables 7](#_Toc502063824)

[编辑器 7](#_Toc502063825)

[弹窗 8](#_Toc502063826)

[Util包 8](#_Toc502063827)

[convert 8](#_Toc502063828)

[date 8](#_Toc502063829)

[email 9](#_Toc502063830)

[json 9](#_Toc502063831)

[sms 9](#_Toc502063832)

[string 10](#_Toc502063833)

[view 10](#_Toc502063834)

[其他 10](#_Toc502063835)

[组件 10](#_Toc502063836)

[任务调度 10](#_Toc502063837)

[异步任务 10](#_Toc502063838)

[支付 11](#_Toc502063839)

[推送 11](#_Toc502063840)

[缓存 11](#_Toc502063841)

[权限 12](#_Toc502063842)

[开发规范 12](#_Toc502063843)

[包规范 12](#_Toc502063844)

[命名规范 12](#_Toc502063845)

[SQL规范 13](#_Toc502063846)

[接口规范 13](#_Toc502063847)

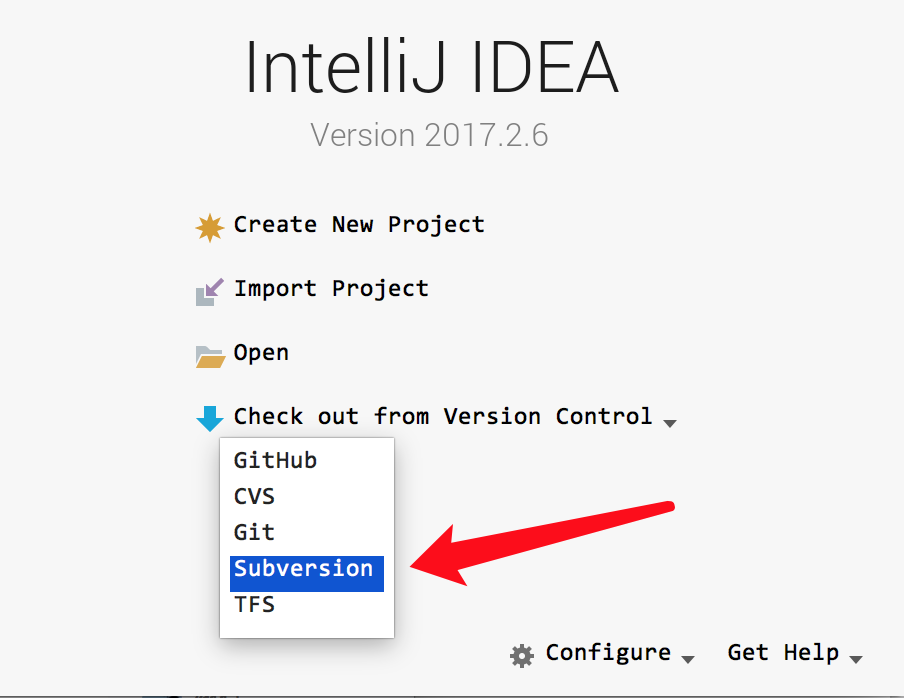
[SVN 13](#_Toc502063848)

[注释 13](#_Toc502063849)

# 操作手册

## 部署项目

1、使用IntelliJ IDEA，打开软件，从SVN检出项目

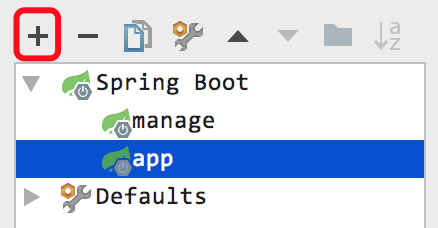


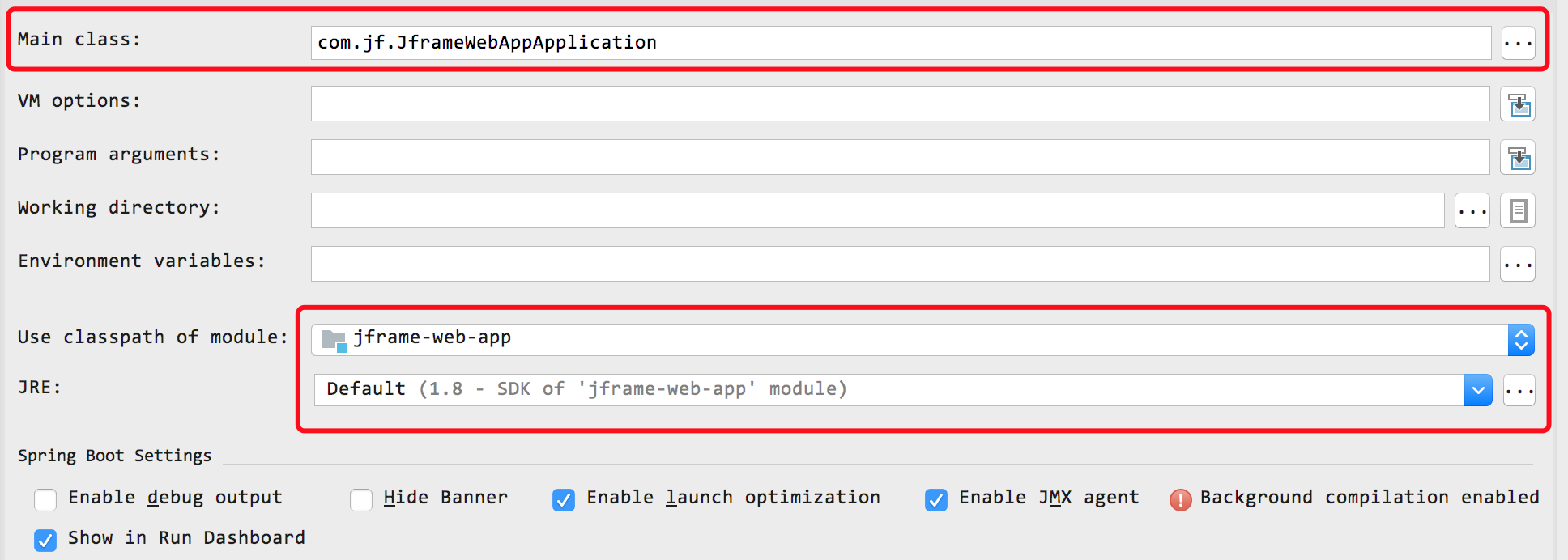
2、点击Open，打开项目所在路径（最外层文件夹），不建议使用Import Project。

3、点击最右侧的选项卡Maven Project，，,

再点击‘+’，选择 jframe/pom.xml文件，之后会自动关联模块，并下载Jar包。

4、 

**点击上图1处**，显示，点击‘+’，找到SpringBoot(如没有，则可能未安装专业版IDEA)。



配置如上，选择对应的引导类和模块，确认即可。

**点击2处**，选择JDK版本，最低**1.8**.（数据库版本最低5.6）

5、配置文件

**数据库**：jframe/doc/jframe.sql

**日志**：jframe-core/src/main/resources/logback-boot.xml

找到LOG\_HOME，配置日志存储位置，如使用dev（开发）环境，则不需要配置此项。

**项目配置**：jframe-core/src/main/resources/application-dev.yml

找到datasource.url，配置本地数据库，如已是测试服务的地址，则不需要更改；找到system.static\_path，建立本地static目录（如：D://project/static/），然后将绝对地址复制到此，必须以/或\结尾，static\_host为项目启动后的IP地址。

项目的端口号在各个端下 application.yml文件中设置。

6、以上流程成功完成后，就可以启动项目



上图1为编译，用于热部署（DEBUG模式下），2为选择端，3为启动项目，4为DEBUG模式下启动。

## 开发流程

1、数据库实体类和Mapper生成

jframe-utils/src/main/java/com/jf/db/GenerateBeansAndMybatisUtil



**建立文件夹**，**修改数据库连接池**，右键Run该类，即可在文件夹内生成对应的实体类和Mapper文件。

将对应的实体类和Mapper分别放置到jframe-model模块中。

实体类都需要继承BaseVo类。

mapper提供的接口：

findByCondition (BaseVo baseVo);

findById (Long id);

insert (User bean);

insertBatch (List<User> list);

update (User bean);

delete (Long id);

提供的xml代码片段：

COMMON.DATE COMMON.PAGE COMMON.USER

baseCondition simpleColumn allColumn

*对于SQL规范会在最后的规范中说明。*

2、**Service**

jframe-core/../service

如果Service需要事务，请在类前加上@Transactional

注入Mapper，使用@Resource或@Autowire

3、在jframe-web-manage开发页面

如果是页面，直接在类注解**@Controller**，方法上@RequestMapping，返回String；

如果全是JSON数据，使用**@RestController**，并且方法只需要注解@GetMapping或@PostMapping，返回ResMsg类型。

页面模板在resources/templates，以ftl结尾，按模块划分。

head标签加入<#include "include.ftl"/>引入全局JS/CSS。

页面布局统一，查询栏，工具栏，列表，分页。

在jframe-web-app开发APP接口

一般使用**@RestController+**@RestController或@PostMapping注解，返回ResMsg。

操作Token，直接在需要的方法参数前加上：**@Token**，目前是token和id进行绑定。

# JS插件

## Ajax

Ajax.ajax({  
 **url**: **'/admin/user/userEdit'**,  
 **params**: $form.serialize(),  
 success: **function** (data) {  
 **if** (data.**code** == 0) {  
 showMsg(data.msg, 1, **function** () {  
 });  
 } **else** {  
 showMsg(data.msg, 2);  
 }  
 }  
});

## 上传

<**a class="img-area-btn" id="avatar"**></**a**>

**$**(**"#avatar"**).Uploader({  
 **limit**: 1, 限制数量  
 **default**: **'${user**.**avatar}'**, 默认图片  
 **type**: 1 1.图片 2.文件  
});

## 城市选择

<**input type="text" id="city-picker" name="address" value="${user**.**address }" class="form-control" placeholder="住址"**>

cityPicker(**"#city-picker"**, **"distrdict"**, **"address"**);

参数：id；类型（city|district|street）；返回值类型（address|code）

## 时间选择

<**input type="text" class="form-control input-sm" id="startDate" name="startDate" placeholder="开始时间"**>

datePicker(**'#startDate,#endDate'**, **"yyyy-mm-dd"**, 2);

参数：id（多个）；格式化（*4 3 2 1*）；格式化（*yyyy yyyy-mm yyyy-mm-dd yyyy-mm-dd hh:ii*）

## 树

<**ul id="tree" class="ztree"**></**ul**>

**var *zindex***;  
**var *tree*** = **""**;  
**var *setting*** = {  
 **check**: {  
 **chkboxType**: {**"Y"**: **"p"**, **"N"**: **"ps"**},// 勾选checkbox对于父子节点的关联关系  
 **chkStyle**: **"checkbox"**,  
 **enable**: **true** // 是否复选框  
 },  
 **data**: {  
 **simpleData**: {  
 **enable**: **true** }  
 }  
};

***tree*** = **$**.**fn**.zTree.init(**$**(**"#tree"**), ***setting***, data);  
***tree***.expandAll(**true**);

## DataTables

<**table id="table" class="table table-striped table-bordered table-hover" width="100%"**></**table**>

*具体JS参考：user/user\_list.ftl*

## 编辑器

<**textarea name="content" style="width**: 100%;**height**: 500**px**;**"**>**${**article.content **}**</**textarea**>  
<**div class="word-count"**>字数统计：<**span id="wc"**></**span**></**div**>

**var** editor;  
**KindEditor**.ready(**function** (K) {  
 editor = K.create(**'textarea[name="content"]'**, {  
 **resizeType**: 1,  
 **allowPreviewEmoticons**: **false**,  
 **allowImageUpload**: **true**,  
 **uploadJson**: **'<%=**basePath**%>uploadwithKE.do'**,  
 afterBlur: **function** () {  
 **this**.**sync**();  
 },  
 afterChange: **function** () {  
 K(**'#wc'**).html(**this**.count(**'text'**));  
 }  
 });  
});

## 弹窗

*openLayerUrl*(title, width, height, url, maxmin) *打开layer url*

*openLayerContent*(title, width, height, $obj) *打开HTML窗口【对象】*

*openLayerHTML*(html, width, height, func) *打开HTML窗口【HTML】*

*openLayerLoading*() *加载动画*

具体参考*/global.js* 或者直接参考layui：http://layer.layui.com/

*\*大部分插件都是二次封装，方便调用，可在jframe/public/static/common/js/global.js查看。*

*\*其他插件见jframe/public/static/library/plugins*

# Util包

## convert

Convert.doubleAdd(**double** d1, **double** d2) 浮点数相加

Convert.doublesAdd(**double**... doubles) 不定参数相加【请确保最少有两个参数】

Convert.doubleFormat(**double** d) double保留两位小数点

MoneyUtil.transMoney(Double money) 万元金额转换

number2CN(BigDecimal numberOfMoney) 把输入的金额转换为汉语中人民币的大写

## date

DateUtil.getCurrentTime(Integer type) 获取当前时间

DateUtil.dateToStr(Date date) date转string

DateUtil.strToDate(String dateString) string转date

DateUtil.getYearAndMonth(**boolean** flag) 获取年月

DateUtil.between(String d1, String d2, String date) 检查某时间是否在时间段内 格式：yyyy-MM-dd

DateUtil.getMonthFirstDay() 获取当月第一天

DateUtil.getMonthLastDay() 获取当月的最后一天

DateUtil.diffDays(Date startDate, Date endDate) 计算两个时间之间相差的天数

DateUtil.remainDateToString(Date startDate, Date endDate) 计算剩余时间

DateUtil.add(Date date, **int** type, **int** value) 日期加减

RelativeDateFormat.format(Date date) 获取相对时间

## email

注入EmailService

调用sendTemplateMail (Map<String, Object> root, String toEmail, String subject, String templateName)

root 参数，键值对

toEmail 收件人

subject 主题

templateName 模板名称

## json

JSONUtils.toJSONString(List<T> list) 将List对象序列化为JSON文本

JSONUtils.toJSONString(Object object) 将对象序列化为JSON文本

JSONUtils.toJSONString(JSONArray jsonArray) 将对象转换为JSON对象数组

JSONUtils.toJSONObject(Object object) 将对象转换为JSON对象

。。。

## sms

注入SMService

调用send(String content, String phone)

content 内容

phone 手机号

模板发送sendSms(Map<String, String> root, String templateName, String... phones)

root 参数，键值对

templateName 模板名称

phones 手机号

## string

StringUtil.getOrderCode() 获取订单编号

StringUtil.getSmsCode(**int** size) 获取手机验证码

StringUtil.strToArray(String answer) 将字符串转换为一个新的字符数组

StringUtil.MD5Encode(String origin) 对密码进行加密

StringUtil.randomFilename(String file) 文件随机命名

StringUtil.getColumnNum(String num) 阿拉伯数字转中文

StringUtil.getTokenId() 构造SESSIONID

## view

新建类，继承AbstractXlsView

参考jframe-web-manage/src/main/java/com/jf/controller/view/UserExcel.java

## 其他

db、file、http、system、page更多请参考具体类

# 组件

## 任务调度

jframe-core/src/main/java/com/jf/system/job

新建任务类，如日任务：DayJob，使用注解：**@EnableScheduling,@Component**

在具体的方法上加上注解：**@Scheduled**(cron = "0 0 23 \* \* ?")

可在类中直接注入Service

## 异步任务

jframe-utils/src/main/java/com/jf/sms/SMService.java

异步方法所在类需要注解：**@Component**

在需要异步的方法上加上注解：**@Async**

## 支付

jframe-core/src/main/java/com/jf/system/third/alipay

下单：AliPayUtil.alipay(String body, String subject, Double price, String orderNum, String url)

转账：transfer(String orderNum, Double price, String account, String realname)

检查到账状态：check(String orderNum, String thirdNum)

调用：AliPayUtil.alipay("充值", "账户充值", money, orderNum)

## 推送

jframe-core/src/main/java/com/jf/system/third/jpush/JPushService.java

注入JPushService，调用方法

sendPush(**final int** devive, **final** String alert, **final** Map<String, String> extras, **final** String... id)

devive 设备类型

alert 推送内容

extras 额外参数

id 设备id

## 缓存

jframe-web-app/src/main/java/com/jf/system/cache/RedisClientTemplate.java

注入：RedisClientTemplate

调用接口方法

如：**template.set**(String key, String value)

## 权限

对于后台权限，事先在后台 系统管理->模块管理 配置好菜单和按钮功能后，

在具体方法中加入注解：**@AuthPassport**，默认验证登录和权限，可配置login=true/false , right=true/false，login表示登录，right表示操作权限。

@RequestMapping(**"/userList"**)  
@AuthPassport(right = **false**)  
**public** String userList() {}

# 开发规范

## 包规范

包名：全部小写，尽量一个英文单词，如：user order article

service通常以业务类型进行分包，如，用户模块：user；订单模块：order；充值模块：charge

controller通常以模块进行分包，如，订单模块：order

## 命名规范

**1、Java**：

**类名**，首字母大写，多个英文时，遵循驼峰形式

如：AppController OrderItem UserService

**常量**，命名全部大写，多个使用‘\_’ 隔开

如：SESSION\_NAME

**枚举**，命名全部大写

如：SUCCESS YES

**代码格式化，**快捷键：Ctrl+Alt+L

**2、数据库**：全部英文小写，多个英文使用‘\_’ 隔开，表使用‘t\_’开头

如：**表名** t\_user t\_order

**字段名**user\_name user\_id order\_num

**3、HTML**：全部英文小写，多个英文使用‘\_’ 隔开

如：user\_list.html user\_edit.ftl index.jsp

**4、JavaScript**：文件全部英文小写，多个英文使用‘\_’或‘-’ 隔开

方法和变量也遵循驼峰形式，但首字母是小写

如：**文件**：app-index.js global.js

**方法**：getVer() showMenu()

## SQL规范

关键字全部**大写**

如：**SELECT** id,name **FROM** t\_user

**UPDATE** t\_user **SET** name = '' **WHERE** ...

对于mapper，需对生成的代码进行精简和优化，如有默认值的不需要新增，不需要更新的字段删除。

格式化，结构清晰；对于id、时间、类型、状态等设置默认值

## 接口规范

1、APP接口统一以/app开头，返回类型为：

无数据：ResMsg(0, **SUCCESS**)

有数据：ResMsg(0, **SUCCESS, data**)

地址：/app/user/userList，如下：

user是根据类名，如UserController来命名的，或是模块名

userList为请求名称，首字母小写，后遵循驼峰形式

2、分页：**PageInfo**类，直接注入ResMsg.data中。

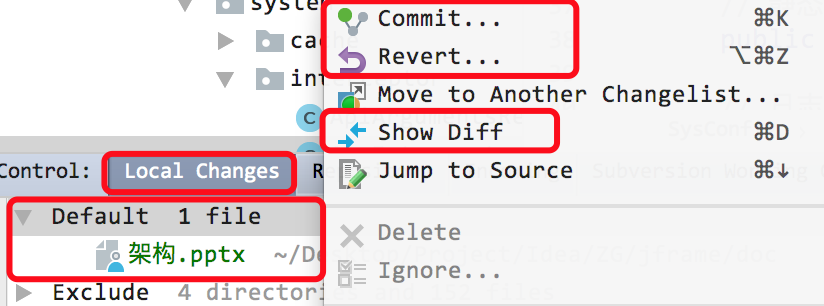
3、**restful**风格（建议）： /app/user/10001 等同于：/app/user?userId=10001。

4、app请求需要在**header**中注入：Req-Type=APP token

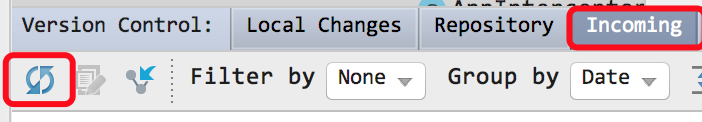
5、可通过IDEA自带的插件进行接口测试：Tools -> **REST Client**，将测试的结果录入到API文档中供APP开发人员参考。

## SVN

一般开发人员只负责某一块，不涉及其他人员的模块，如果涉及到，最先提交的等待，后一个需要提交的先更新，后提交，第一个人再更新（相同类会直接覆盖）。



以上为本地提交常用的功能；更新版本（右键->Update）↓↓↓



对于一些**配置文件**、**JUnit测试类、IDE自动生成的配置文件**，请不要提交。

如将日志的路径改为自己电脑的路径，该文件不应提交，将其移至排除目录（新建目录Exclude）。

## 注释

任何类、方法、常量都需要编写注释

模板（可在IDE里设置）：

/\*\*  
 \* Created with IntelliJ IDEA.  
 \* Description: 说明  
 \* User: xujunfei  
 \* Date: 2017-11-28  
 \* Time: 15:52  
 \*/

**public class** ClassName {

/\*\*  
 \* 删除用户  
 \*  
 \* **@param *userId*** \* **@return** \*/  
**public int** deleteUser(Long userId) {

// APPID  
**public static** String *appid*;

