**Sofn基础开发平台简介**

* sofn基础开发平台是基于分布式的系统架构,使用Spring整合
* 使用Maven对项目进行模块化管理，提高项目的易开发性、扩展性。
* 系统包括：代码生成模块、系统管理模块、Web展示模块。
* 代码生成模块：运行maven命令生成Mybatis文件和实体类到指定的模块。
* 系统管理模块：包括用户管理、权限管理、数据字典、系统参数管理等等。
* 调度管理模块：注册管理定时器，使用多种方式调用不同模块的任务实现。
* 每个模块都是独立的系统，可以无限的扩展模块，模块之间使用Dubbo或MQ进行通信。（推送）
* 每个模块服务多系统部署，注册到同一个Zookeeper集群服务注册中心，实现集群部署。

微应用==

对应用的监听

1、通用用户权限管理模块

2、应用集成组件

**主要功能**

1. 数据库：Druid数据库连接池，监控数据库访问性能，统计SQL的执行性能。 数据库密码加密。
2. 持久层：mybatis持久化，aop切换数据库实现读写分离，PageHelper分页。Transtraction注解事务。
3. MVC： 基于spring mvc注解,Rest风格Controller。Exception统一管理。
4. 调度：Spring+quartz, 可以查询、修改周期、暂停、删除、新增、立即执行，查询执行记录等。
5. 基于session的国际化提示信息，职责链模式的本地语言拦截器,Shiro登录、URL权限管理。会话管理，强制结束会话。
6. 缓存和Session：注解redis缓存数据，Spring-session和redis实现分布式session同步，重启服务会话不丢失。
7. 多系统交互：Dubbo,ActiveMQ多系统交互，ftp/sftp/fastdafs发送文件到独立服务器，使文件服务分离。
8. 前后端分离：没有权限的文件只用nginx代理即可。
9. 日志：log4j2打印日志，业务日志和调试日志分开打印。同时基于时间和文件大小分割日志文件。
10. 项目构建：maven构建项目，mybatis generator生成mybatis映射文件和Model。
11. 工具类：excel导入导出，汉字转拼音，身份证号码验证，数字转大写人民币，FTP/SFTP/fastDFS上传下载，发送邮件，redis缓存，加密等等。

**技术选型**

* 核心框架：Spring Framework 4.3.0 + Dubbo 2.5.3
* 安全框架：Apache Shiro 1.2
* 任务调度：Spring + Quartz
* 持久层框架：MyBatis 3.4
* 数据库连接池：Alibaba Druid 1.0
* 缓存框架：Redis
* 会话管理：Spring-Session 1.2
* 日志管理：SLF4J、Log4j2
* 前端框架：Angular JS + Bootstrap + Jquery

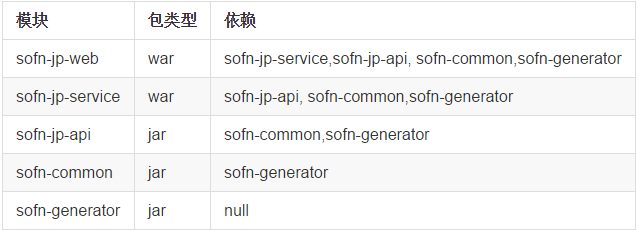
## 框架说明

每个模块都是一个单独的系统，每个系统可以发布多个service，不同的web间使用相同的redis缓存保证session相同。  
目前项目下有5个子项目：sofn-common, sofn-jg-pai, sofn-jg-service, sofn-jg-web, sofn-front  
1. sofn-common是通用配置和工具类；  
2. sofn-jg-pai是sofn-jg-service和sofn-jg-web交互需要的bridge和model；sofn-jg-service是系统管理相关的数据库操作服务,包括 业务处理层和 DAO 层；  
3. sofn-jg-web是连接sofn-front和sofn-jg-service的桥梁。  
4. sofn-jg-web和sofn-jg-service之间通过dubbo交互。Web和Service可以创建多个，Web使用相同的spring-session命名空间并注册到相同的redis服务实现session共享，Web可以调用不同Service的接口。  
5. sofn-front使用angularJS框架异步请求数据并展示。  
6. sofn-jg-service使用Druid连接数据库，可以监控sql执行情况;

使用AOP实现读写数据库切换，使用Mybatis持久化框架，使用PageHelper辅助分页，使用spring注解把数据缓存到redis；改造Dubbo使含有事务的类可以作为服务实现类；sofn-jg-web使用SpringMVC框架管理控制器；使用Shiro辅助管理权限，权限严格控制到给个URL。

### 模块依赖关系





### Controller 说明

@RestController *//RESTful 风格注解 @Controller +@ResponseBody*

@Api(value = "登录接口", description = "登录接口") *// swagger 注解 生成在线api 文档*

**public** **class** **LoginController** **extends** **BaseController** {

@Autowired

**private** SysUserService sysUserService;

*// swagger 接口注解*

@ApiOperation(value = "用户登录")

@PostMapping("/login") */\*等效于 @RequestMapping(value = "/login",method = RequestMethod.POST) \*/*

**public** Object **login**(ModelMap modelMap,

*/\*@ApiParam为swagger使用的参数注解\*/*

@ApiParam(required = **true**, value = "登录帐号")

@**RequestParam**(value = "account", required = **false**) String account,

@**ApiParam**(required = **true**, value = "登录密码")

@**RequestParam**(value = "password", required = **false**) String password) {

*// 参数检查*

Assert.notNull(account, "ACCOUNT");

Assert.notNull(password, "PASSWORD");

*//登录逻辑*

**if** (LoginHelper.login(account,sysUserService.encryptPassword(password))) {

modelMap.addAttribute("account",account);

modelMap.addAttribute("passowrd",password);

**return** setSuccessModelMap(modelMap);

}

**throw** **new** LoginException(Resources.getMessage("LOGIN\_FAIL"));

}

备注：Swagger 是一个规范和完整的框架，用于生成、描述、调用和可视化 RESTful 风格的 Web 服务。总体目标是使客户端和文件系统作为服务器以同样的速度来更新。文件的方法，参数和模型紧密集成到服务器端的代码，允许API来始终保持同步。

### Service api 说明

@Service

**public** **class** **SysUserService** **extends** **BaseService**<**SysUserProvider**, **SysUser**> {

@Autowired

**public** **void** **setProvider**(SysUserProvider provider) {

**this**.provider = provider;

}

**public** String **encryptPassword**(String password) {

**return** provider.encryptPassword(password);

}

}

### Service 业务处理

### DAO

### Generator

根据 xml 配置自动生成实体类和 mapper 文件

## 工具安装

所有工具都使用单机模式,以后再使用集群模式.

### JDK

Java 8

### Zookeeper

使用 zookeeper–3.4.8.tar.gz 最新稳定版  
zookeeper 使用伪集群模式安装

linux 环境下:

**# 进去 zookeeper 配置文件所在的路径**

cd /usr/local/etc/zookeeper

**# 因为 zookeeper 默认使用 zoo.cfg 所以需要将 zoo\_sample.cfg 拷贝改名为 zoo.cfg,如果使用 brew 安装的时候,已经存在 zoo.cfg 文件了**

cp zoo\_sample.cfg zoo.cfg

**# 编辑 zoo.cfg 文件**

vi zoo.cfg

**# 心跳检测毫秒数**

tickTime=2000

**# follower 初始化连接最长忍受的心跳数时间间隔 2000\*10=20秒**

initLimit=10

**# leader 和 follower 之间发送消息请求和应答时间长度 5\*2000=10秒**

syncLimit=5

**# 数据持久化的目录**

dataDir=/usr/local/var/run/zookeeper/data

**# 向 client 暴露的端口**

clientPort=2181

**伪集群模式**  
作为集群测试的一种安装方式  
[Zookeeper 下载](https://mirrors.tuna.tsinghua.edu.cn/apache/zookeeper/)  
本地文件在 工具目录下  
解压命令  
tar -zxvf zookeeper-3.4.8.tar.gz  
解压之后复制2份,我们将创建一个具有3台 Server 的 Zookeeper 集群.



1. zookeeper–1配置  
- 创建 data 目录: mkdir /Users/xxxx/zookeeper/zookeeper-1/data  
- 创建 log 目录: mkdir /Users/xxxx/zookeeper/zookeeper-1/logs  
- 创建 myid 文件:touch /Users/xxxx/zookeeper/zookeeper-1/data/myid,编辑 myid 文件,输入1(其他2个 myid 分别输入2,3),这个跟 Leader 选举有关  
2. 创建 zoo.cfg 文件并配置  
- touch /Users/xxxx/zookeeper/zookeeper-1/conf/zoo.cfg  
- vi /Users/xxxx/zookeeper/zookeeper-1/conf/zoo.cfg

@echo off

echo Starting zookeeper...

**start** "zookeeper" "zookeeper\bin\zkServer.cmd"

echo **Starting** redis...

**start** "redis" "redisbin\redis-server.exe"

echo **Starting** activemq...

**start** "activemq" "activemq\bin\win64\activemq.bat"

echo **Starting** nginx...

**start** "nginx" "nginx\start.bat"

zookeeper–2,zookeeper–3的配置跟zookeeper–1步骤一样,需要根据具体情况修改.记得换数字啊…

3.启动  
分别进入三个服务的bin目录 启动服务  
./zkServer.sh start  
启动的时候可能会报错,不过不用担心,这是因为启动一个 zookeeper 后,会根据配置文件寻找其他 server, 另外2个 server 并没有启动,所以为报错,当另外2台启动后,会根据 Zap 协议来选举 Leader, 然后就好了.可以分别进入 bin 目录使用 ./zkServer.sh status查看当前这台 server 是属于 follower 还是 leader  
4.连接服务  
./zkCli.sh -server 127.0.0.1:2181

### Redis

使用64位的3.x 版本,支持集群  
本地文件在 工具目录下  
直接解压  
双击 redis-server.exe 将以默认配置运行  
更改redis的配置需要修改redis.conf文件,  
进入 redis 目录,以 redis-server.exe redis.conf运行,以下是它一些主要的配置注释：

**#是否作为守护进程运行**

daemonize no

**#Redis 默认监听端口**

port 6379

**#客户端闲置多少秒后，断开连接**

timeout 300

**#日志显示级别**

loglevel verbose

**#指定日志输出的文件名，也可指定到标准输出端口**

logfile redis.log

**#设置数据库的数量，默认最大是16,默认连接的数据库是0，可以通过select N 来连接不同的数据库**

databases 32

**#Dump持久化策略**

**#当有一条Keys 数据被改变是，900 秒刷新到disk 一次**

**#save 900 1**

**#当有10 条Keys 数据被改变时，300 秒刷新到disk 一次**

save 300 100

**#当有1w 条keys 数据被改变时，60 秒刷新到disk 一次**

save 6000 10000

**#当dump .rdb 数据库的时候是否压缩数据对象**

rdbcompression yes

**#dump 持久化数据保存的文件名**

dbfilename dump.rdb

**########### Replication #####################**

**#Redis的主从配置,配置slaveof则实例作为从服务器**

**#slaveof 192.168.0.105 6379**

**#主服务器连接密码**

**# masterauth <master-password>**

**############## 安全性 ###########**

**#设置连接密码**

**#requirepass <password>**

**############### LIMITS ##############**

**#最大客户端连接数**

**# maxclients 128**

**#最大内存使用率**

**# maxmemory <bytes>**

**########## APPEND ONLY MODE #########**

**#是否开启日志功能**

appendonly no

**# AOF持久化策略**

**#appendfsync always**

**#appendfsync everysec**

**#appendfsync no**

**################ VIRTUAL MEMORY ###########**

**#是否开启VM 功能**

**#vm-enabled no**

**# vm-enabled yes**

**#vm-swap-file logs/redis.swap**

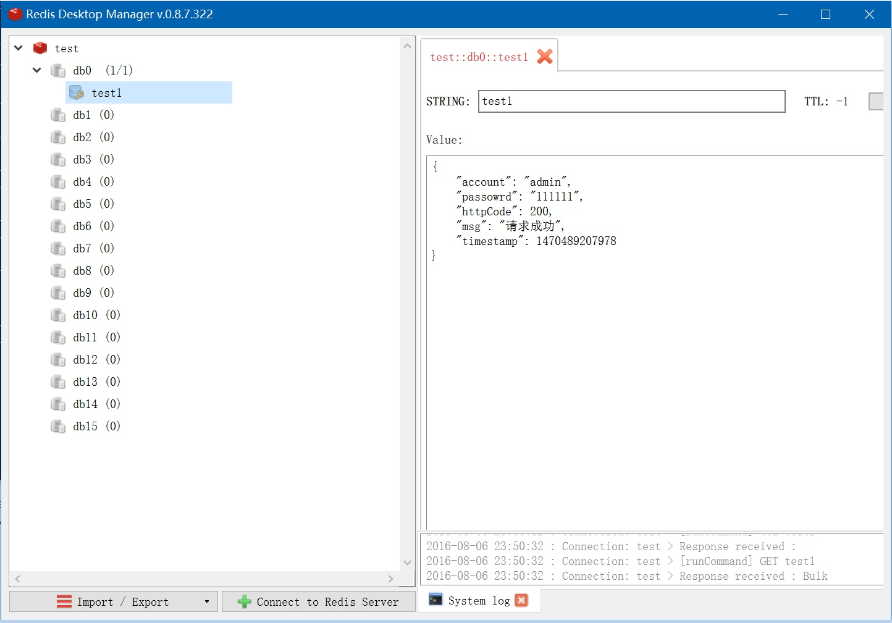
**#vm-max-memory 0**

**#vm-page-size 32**

**#vm-pages 134217728**

**#vm-max-threads 4**

**命令客户端的使用:**  
redis-cli -h 服务器 –p 端口 –a 密码  
redis-cli.exe -h 127.0.0.1 -p 6379  
**可视化客户端使用:**  
redis manager desktop



### ActiveMQ

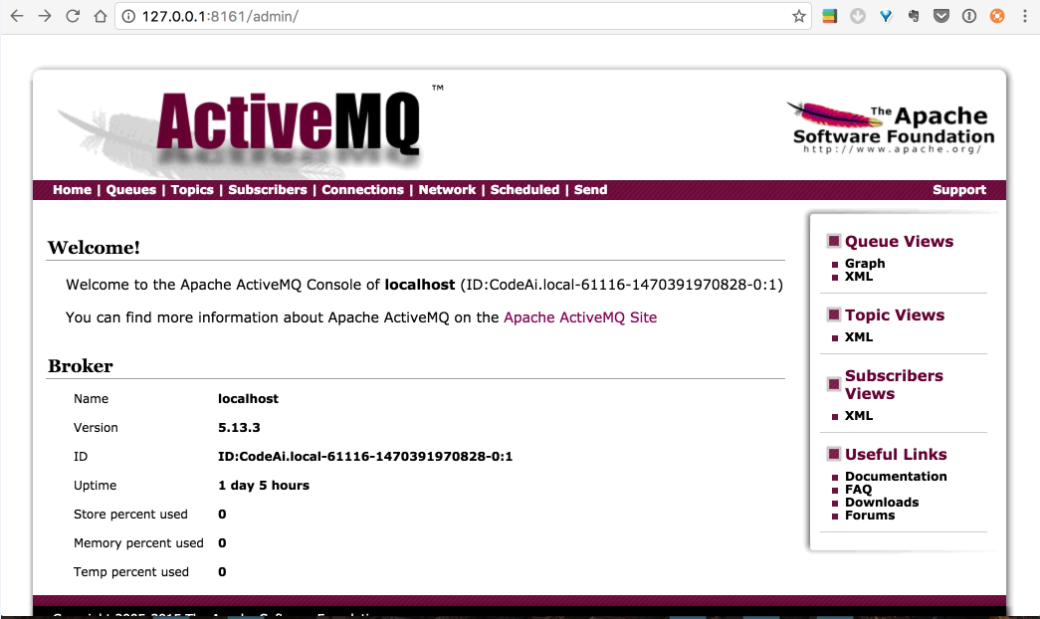
**1. 环境:**  
windows 10 apache-activemq–5.13.4  
**2. 安装:**  
解压缩到apache-activemq–5.13.4-bin.zip到一个目录，比如C:\apache-activemq  
**3. 配置:**  
配置就在C:\apache-activemq–5.2.0\conf目录下三个文件  
activemq.xml  
credentials.properties  
log4j.properties

**4. 启动ActiveMQ:**  
运行C:\apache-activemq\bin\activemq.bat  
**5. 测试:**  
ActiveMQ默认使用的TCP连接端口是61616, 通过查看该端口的信息可以测试ActiveMQ是否成功启动  
netstat -an|find "61616"

C:\**Documents** **and** **Settings**\**Administrator**>**netstat** -**an**|**find** "61616"

**TCP** 0.0.0.0:61616 0.0.0.0:0 **LISTENING**

**6. 监控:**  
ActiveMQ5.0版本默认启动时，启动了内置的jetty服务器，提供一个demo应用和用于监控ActiveMQ的admin应用。  
admin：http://127.0.0.1:8161/admin/  
默认用户名: admin 密码: admin  
在安装目录下的 conf 目录下的 users.properties文件中修改登录验证



### Dubbo-admin

使用 dubbo-admin–2.5.4.war 支持 jdk8  
用tomcat发布dubbo-admin 并修改 WEB-INF下的dubbo.properties  
dubbo.registry.address=zookeeper://127.0.0.1:2181 地址和端口修改为对应的zookeeper  
默认账号  
root root



### Nginx

**1.启动:**　　 解压至c:\nginx，运行nginx.exe(即nginx -c conf\nginx.conf)，默认使用80端口，日志见文件夹C:\nginx\logs

**2.常用配置:**  
C:\nginx\conf\nginx.conf,使用自己定义的conf文件如my.conf，  
命令为nginx -c conf\my.conf

nginx 的简单配置:

*#user nobody;*

**worker\_processes** 1;

**events** {

**worker\_connections** 1024;

}

**http** {

**include** mime.types;

**default\_type** application/octet-stream;

**sendfile** on;

**keepalive\_timeout** 65;

**server** {

*# 设置监控8001端口,可以根据自己的情况更改*

**listen** 8001;

**server\_name** localhost;

*# 设置静态页面目录*

**root** sofn-front所在目录;

*# 默认首页*

**index** index.html;

**location** / {

*# 用户浏览器端的缓存设置,静态资源直接返回,缓存1小时*

**location** ~ .\*\.(js|css|jpg|jpeg|gif|png|swf|htm|html|json|xml|svg|woff|ttf|eot|map|ico)$ {

**expires** 1h;

**if** (-f $request\_filename) {

break;

}

}

*# 动态页面*

**if** ( !-e $request\_filename) {

*# 此处为 sofn-jg-web 所在服务器的 ip + 端口*

**proxy\_pass** http://127.0.0.1:8088;

}

}

**error\_page** 500 502 503 504 /50x.html;

**location** = /50x.html {

**root** html;

}

}

}

**3.nginx常用命令:**  
nginx -s stop 强制关闭  
nginx -s quit安全关闭  
nginx -s reload 改变配置文件的时候，重启nginx工作进程，来时配置文件生效  
nginx -s reopen 打开日志文件

### 一键启动所有项目依赖工具

windows:

@echo off

echo Starting zookeeper...

**start** "zookeeper" "zookeeper\bin\zkServer.cmd"

echo **Starting** redis...

**start** "redis" "redisbin\redis-server.exe"

echo **Starting** activemq...

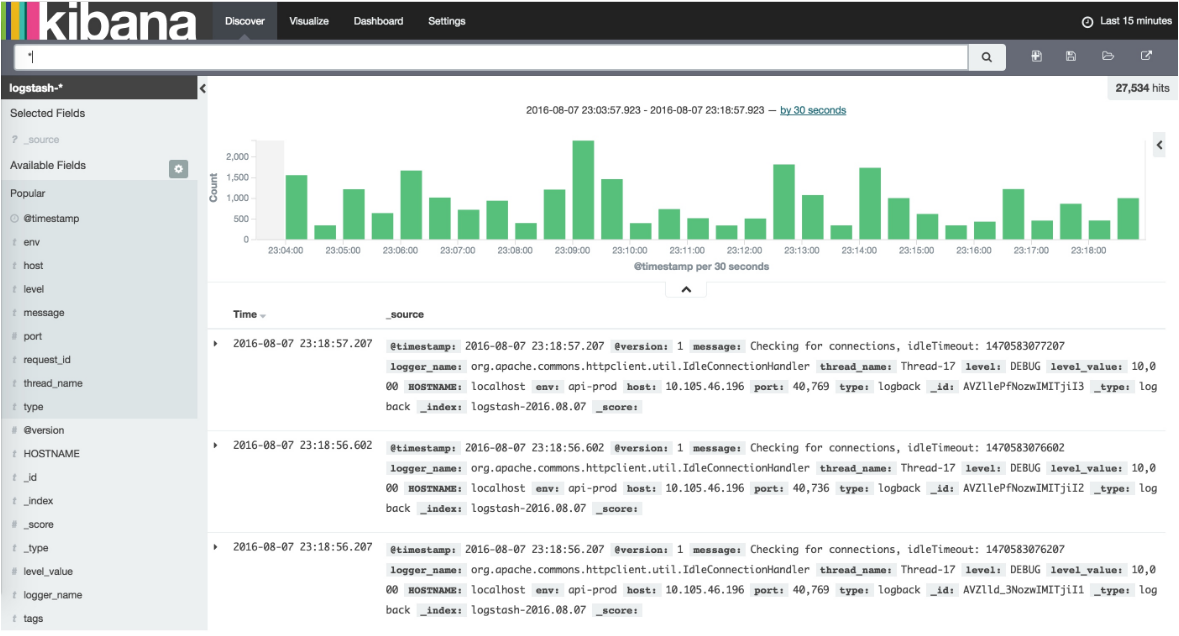
**start** "activemq" "activemq\bin\win64\activemq.bat"

echo **Starting** nginx...

**start** "nginx" "nginx\start.bat"

## 日志系统

使用 Kibana+Logstash+Elasticsearch 搭建日志查询系统



## 测试环境搭建

### Swagger

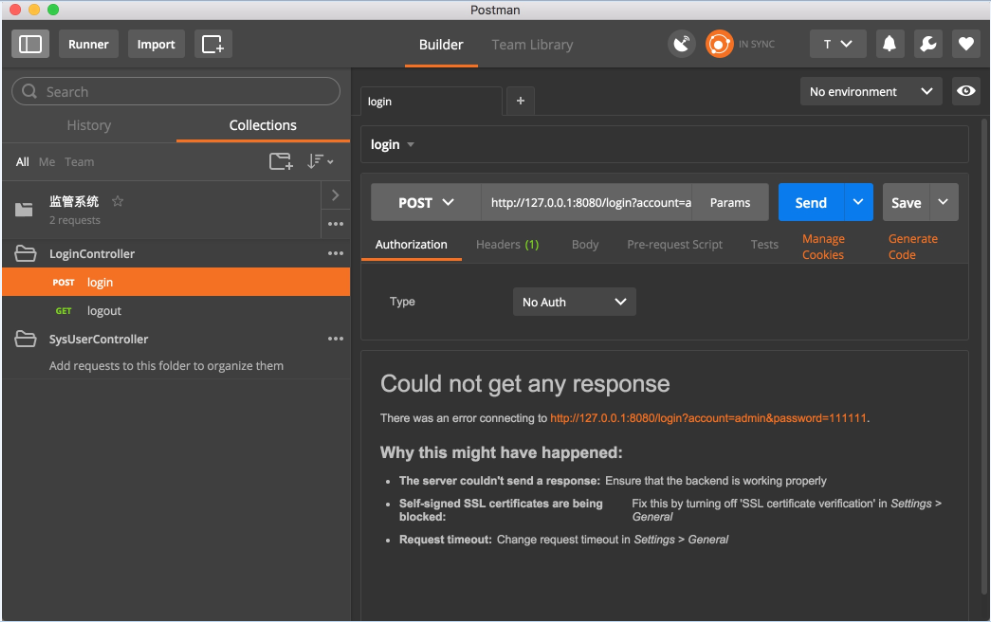
http://sofn-jg-web 所在服务器 ip+ 端口/swagger-ui.html#/

### druid

访问durid查看SQL执行情况(用户密码：druid/druid)：  
http://localhost:8087/sofn-jg-service/druid

### Postman

**共用 Gmail 账号**  
账号:sofntest2016@gmail.com  
密码:sofn@2016  
**翻墙 hosts**  
替换 C:\Windows\System32\drivers\etc\hosts  
chrome 应用市场下载 postman 插件



#### Postman的使用

1. 请求方式
2. Environmnents
3. Tests

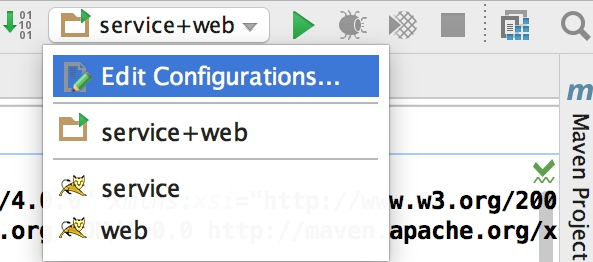
### Postman Interceptor

### Newman

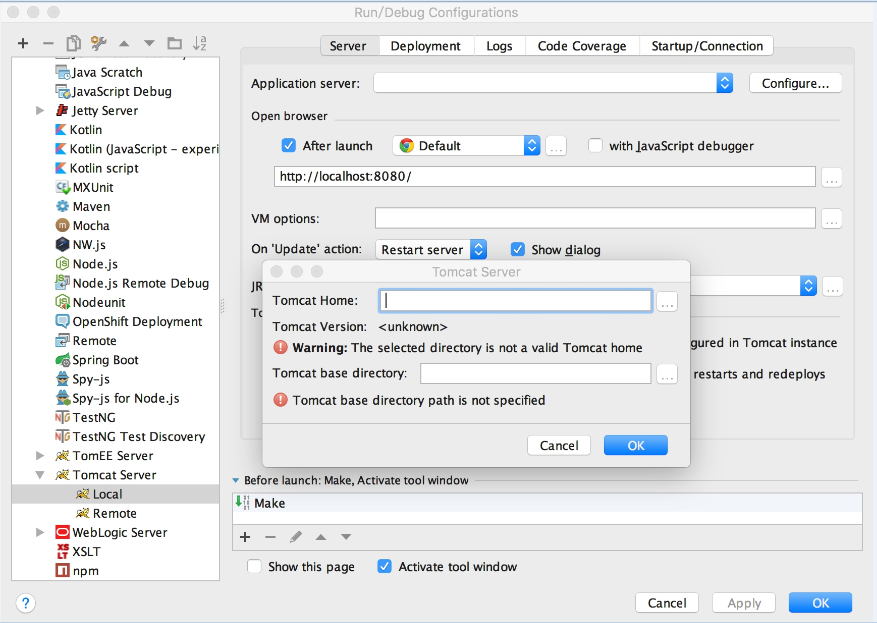
## 项目部署

### 本地开发部署

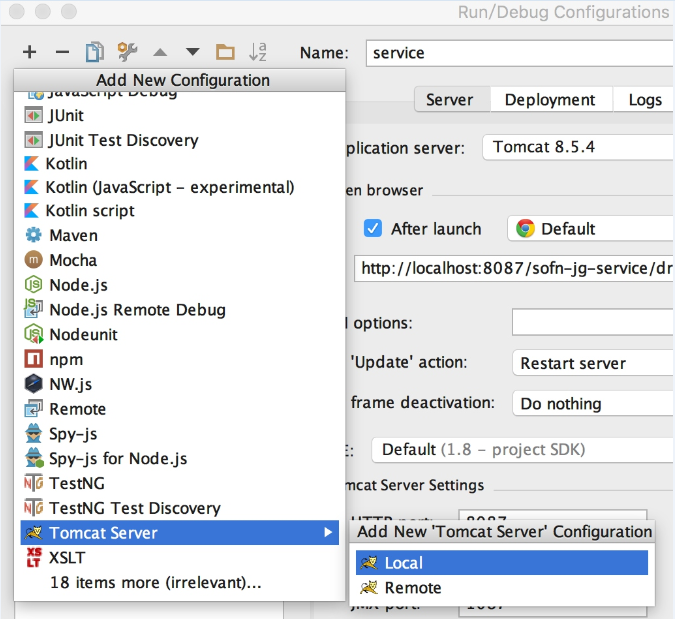
基于 intellij idea 16  
idea 的基本使用方法在文档中有说明  
**1.tomcat:**



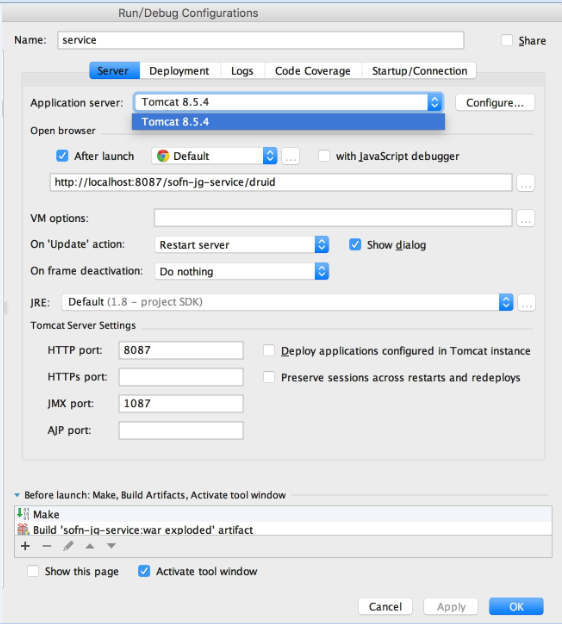
进入后选择左边的 Defaults 设置



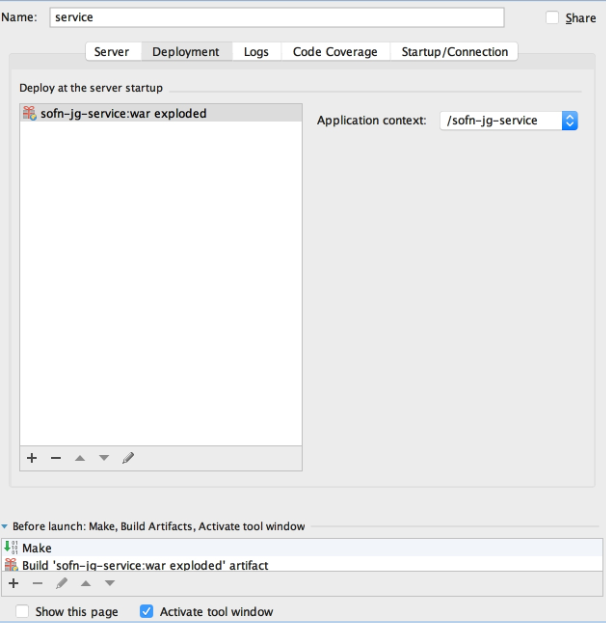
选择 tomcat 安装目录  
此次操作主要是配置默认的 tomcat 环境,以后就不用每次都设置 tomcat 的安装目录了.  
**设置项目使用的 tomcat:**  
选择左上角的 “+”添加项目 tomcat 配置



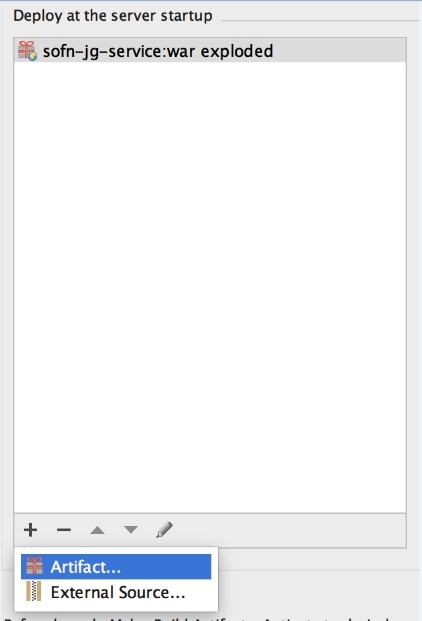
下拉框选择使用的 tomcat 版本



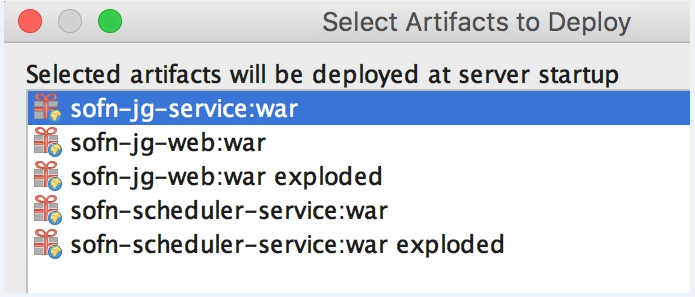
设置Deployment



添加需要部署的项目



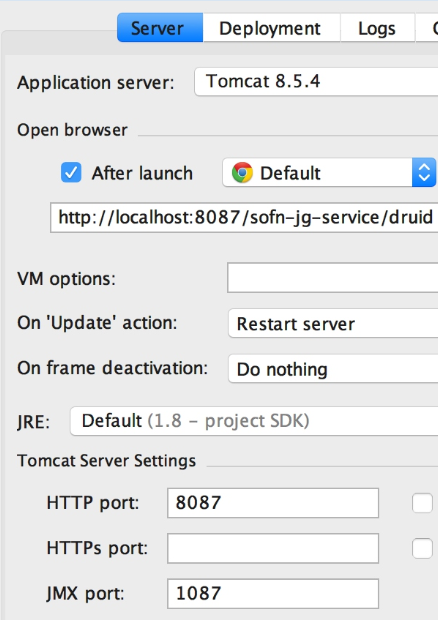
开发时选择 war exploded 类型, 部署时选择 war 类型(或者直接使用 Maven 打包成 war)

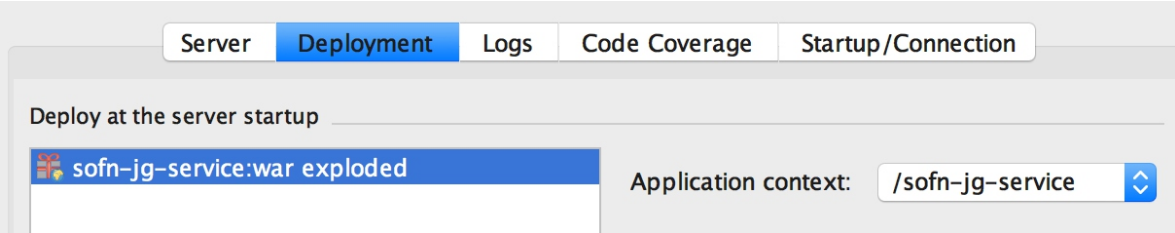


**context 配置**

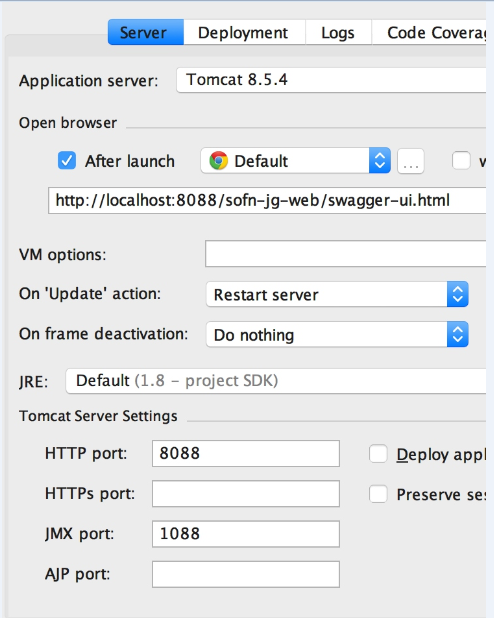


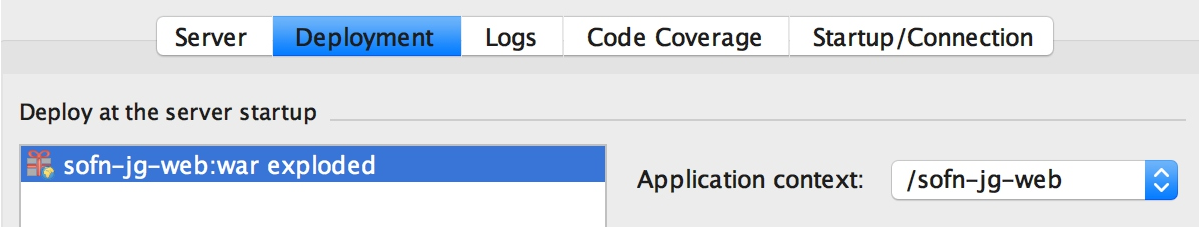
**sofn-jg-service 具体配置**



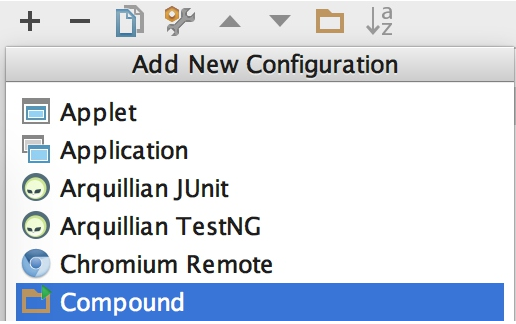


**sofn-jg-web 具体配置**





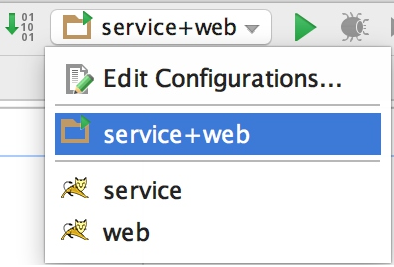
**启动顺序:**  
整个项目是基于dubbo的分布式架构,sofn-jg-web作为 consumer,sofn-jg-service 作为 provider  
sofn-jg-web 依赖于 sofn-jg-service  
所以先启动provider向 zookeeper 注册服务,如果先启动sofn-jg-web则会报错  
**使用 Compound 一键启动2个项目**  
创建Compound



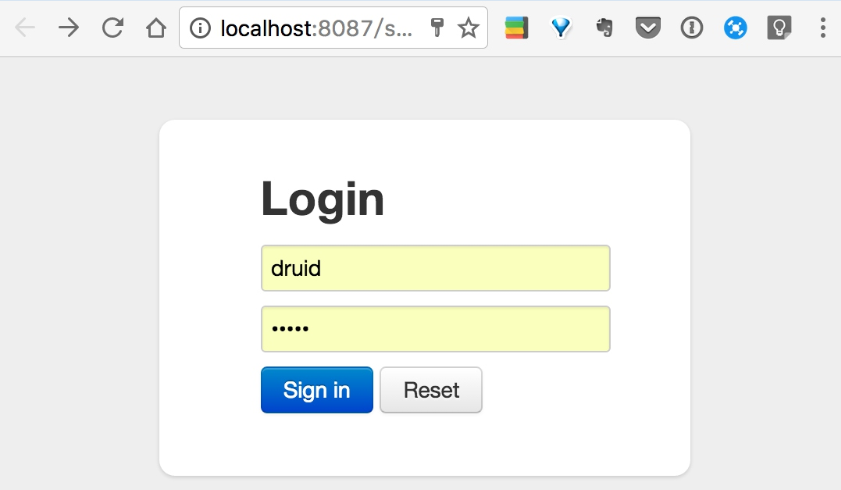
先添加 service, 再添加 web



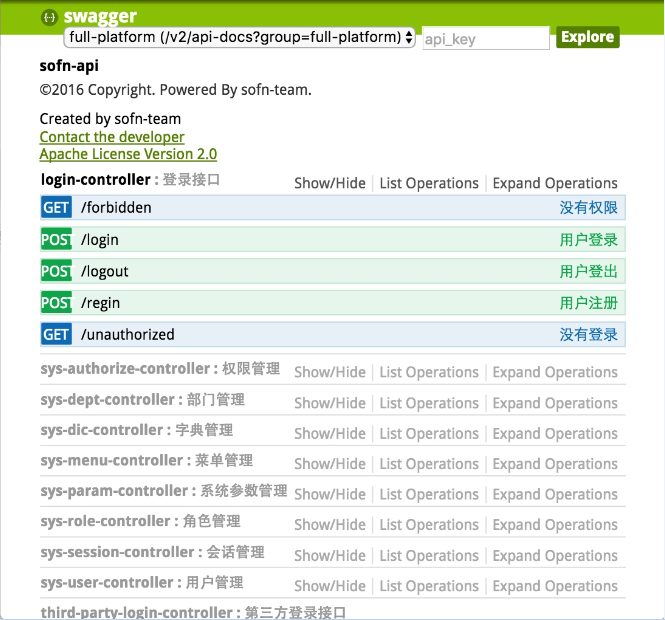
选择 service+web 启动整个项目



service 模块启动成功后会打开默认浏览器并跳转到:  
http://localhost:8087/sofn-jg-service/druid/login.html  
默认账号: druid 密码: druid



web 模块启动成功后会打开默认浏览器并跳转到:  
http://localhost:8088/sofn-jg-web/swagger-ui.html



## 持续集成

### Jenkins

Todo

## 自动化测试

使用 jenkins + postman + newman 进行自动化接口测试