JWebPower 框架 2.1 版使用手册

一、前言

1）权限编号

通俗地说，就是对资源进行编号。

开发中，指请求路径。请求路径一般分为 静态资源请求路径（如图片）；和服务路径（如我们常说的 controller 方法）

权限编号运用示例

示例 1：

我们对服务路径 /hello/666 进行编号，号码为 001

那么，拥有这个 001 号码的用户，就可以访问到 /hello/666

2)权限等级

通过地说，就是方便解决一些权限的逻辑问题。比如 用作 [会员等级]

比如，某资源，需要 12 级会员以上，才能访问。

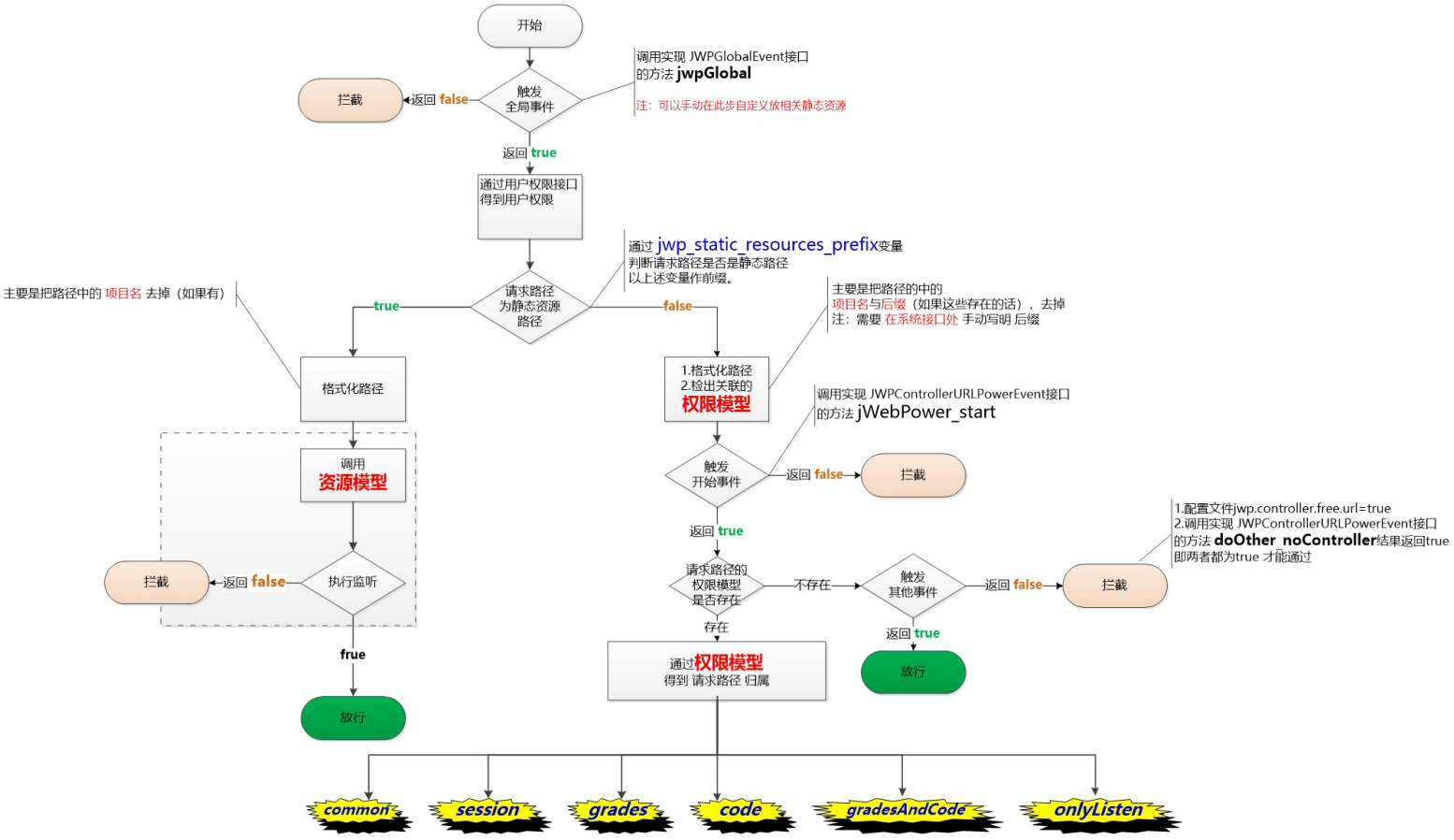
比如，游戏当中，需要 xx 级会员，才能开通某些特权，和购买一些 vip 包。

比如，配合上面的权限编号，需要 会员 1 级以上，并开通了[黑心钻]的用户才能访问。

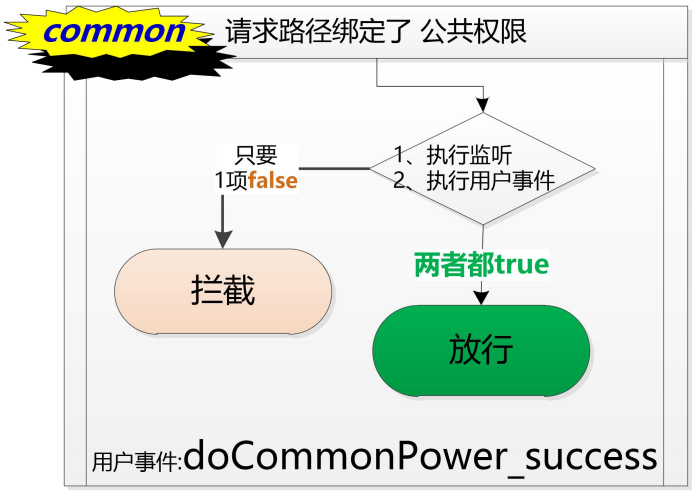
整体来说，以上的设计，配合监听器，无往不利！

二、附-权限管控流程图

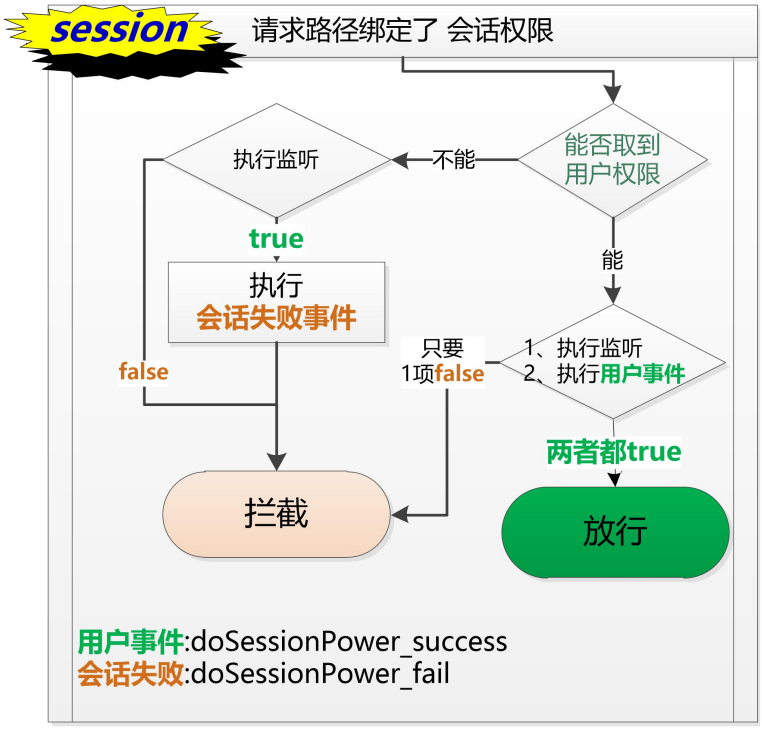
2.1 总图



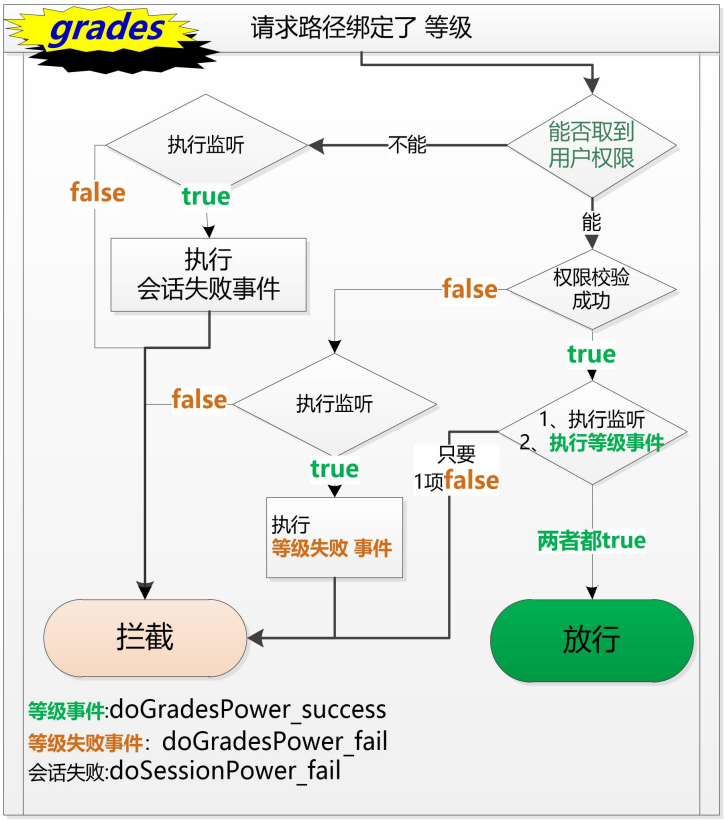
2.2.1 公共区（common）



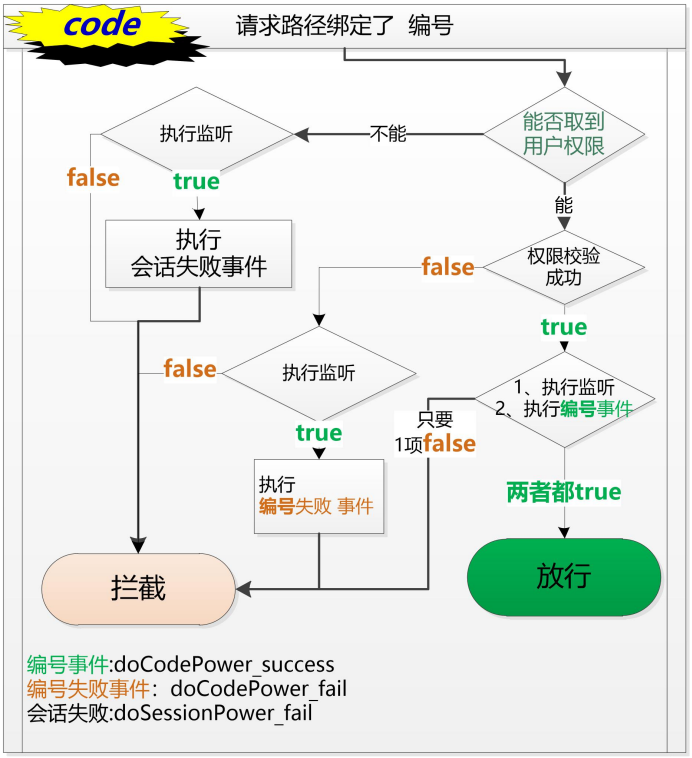
2.2.2 会话区（session）



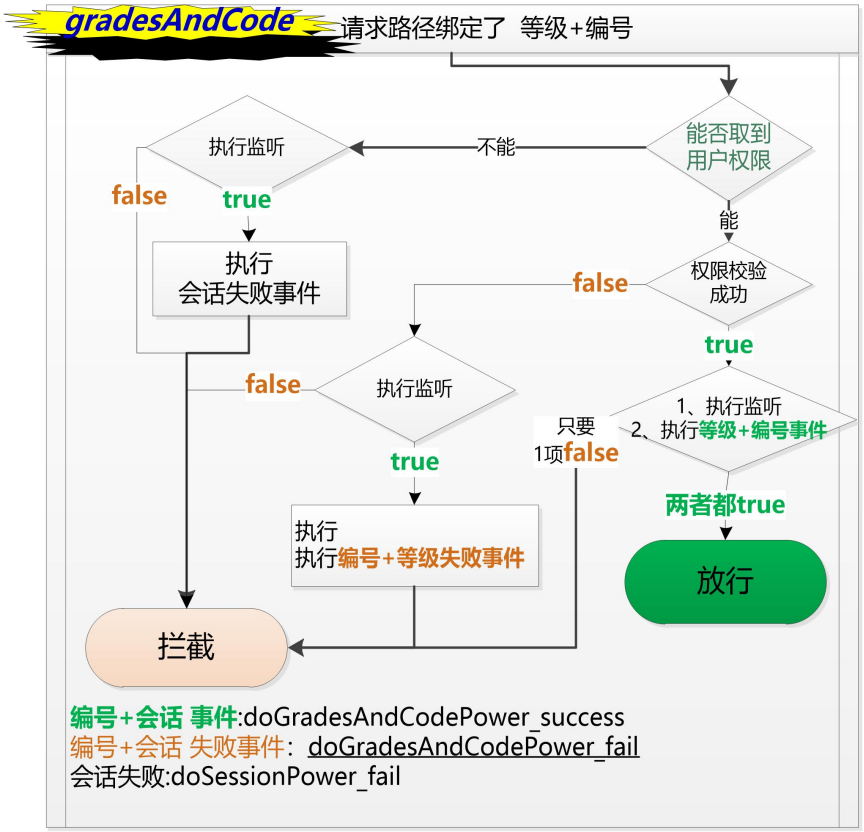
2.2.3 等级区（grades）



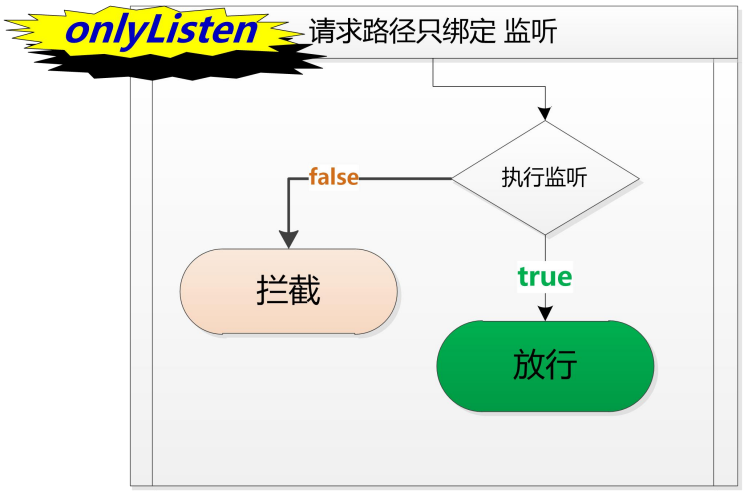
2.2.4 编号区（code）



2.2.5 等级编号区（gradesAndCode）



2.2.6 仅监听区（onlyListen）



三、启动框架

3.1）项目加入 jar 包
如果你的项目是 maven 项目，请如下方式加入 jar

```
<dependency>
    <groupId>weixinkeji.vip</groupId>
    <artifactId>jweb.power</artifactId>
    <version>2.1</version>
</dependency>
```

如果是其他需要直接引入 jar 包的项目，请把到 maven 仓库中，搜索 JWebPower ,然后下相关版本的 jar 包

3.2）创建系统对接类
JWebPower 框架启动时，会执行扫描，并加载对接类来取得用户系统的请求路径，及其他信息。
用途：获取标注在你的类、方法的请求路径（有些架构，类名+方法名就是 请求路径路径）
创建系统对接类，需要现实 JWPSystemInterfaceConfig **接口**
如：

```
/**
 * 示例
 * <p>
 * JWebPower框架 对接SrpingMVC框架 实例类
 * </p>
 *
 * @author wangchunzi
 *
 */
public class SpringMVCSIC implements JWPSystemInterfaceConfig
```

作者提供下一个与 SpringMVC 对接的例子。可到以下地址，找到代码。

```
<dependency>
    <groupId>weixinkeji.vip</groupId>
    <artifactId>jweb.power.expand.springMVC</artifactId>
    <version>2.1</version>
</dependency>
```

模板代码归属项目 https://github.com/weixinkeji/jweb.power.expand.springMVC.git
（**注：请选择 2.1 分支**）

对 nutzMVC 的系统接口示例与项目整合：https://github.com/weixinkeji/jweb.power.expand.nutzMVC.git

3.3）创建用户权限对接类

用途：框架执行检验时，需要用到用户的权限，框架就会调用用户的实现类的方法来取得用户权限

创建权限对接类，需要现实 **JWPUserInterface 接口**

如：

```
public class UserPower implements JWPUserInterface {
    @Override
    public JWPUserPower getUserPowerCode(HttpServletRequest req, HttpServletResponse resp)
            throws IOException, ServletException {
        Object obj=req.getSession().getAttribute("userPowerSession");
        if(null==obj) {
            return null;//未登录
        }
        return (JWPUserPower)obj;
    }
}
```

3.4）创建配置文件

默认在项目的根目录下。创建名字如下的属性文件 **JWP.properties**

配置内容参考：

```
#扫描的包
jwp.scan.package=你自己 controller 类所在的包，及 JWebPower 所在的包路径
#静态资源
jwp.static.resources.prefix=/static/
#true：不在管理范围内的游离路径，允许任何人访问
#false：  不在管控范围的游离路径，不允许任何人访问！（默认）。
jwp.controller.free.url=false
#默认是支持动态路径（路径即参数的意思），使用{}表示动态参数
jwp.controller.dynamics.url=true
#在控制台输出框架启动信息（默认为 false)
jwp.print.console=false
```

3.5）配置 JWebPower 过滤器

```
<filter>
   <filter-name>jwebPower</filter-name>
   <filter-class>weixinkeji.vip.jweb.power.JWPFilter</filter-class>
</filter>
<filter-mapping>
   <filter-name>jwebPower</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

四、控制区 路径监控

注：

1、 请求路径须与某个方法或类绑定一起。如果你的路径没有方法绑定，那么，请在表达式里，直接存入【完整】的请求路径。

2、 绑定权限优先级方法>类>表达式:

a) 如果方法有权限注解，则使用方法的（其他的全部无视）

b) 如果方法没有权限注解，类有。则使用类上的权限（无视表达式）

c) 如果方法、类都没有权限注解，则使用表达式的权限

4.1）控制类上：注解方式（推荐）

绑定权限 之注解类解说

注解	权重	作用	备注
@JWPCommon	0	公共访问区	放行
@JWPSession	1	会话区	表示用户已经登录
@JWPGrades	2	等级区	权限等级（锁定已登录）
@JWPCode	2	编号区	权限编号（锁定已登录）

说明：

1、可以标注在类或方法上。

3、2、当多个注解同时对某个方法或类 进行标注时(如果方法上有权限注解，仅生效方法上的)，权重大的，会覆盖权重小的。同权重相融。

以 JWebPower 在 SpringMVC 的示例：

4.1.1）@JWPCommon

作用：放行

```
@JWPCommon  //注意看这里。
@Controller
public class LoginController {

    @RequestMapping(path = "/", method = RequestMethod.GET)
    public String index() {
        return "index";
    }
```

解说：

1、index()没有任何的权限注解，所以，采用类上的权限注解

2、当访问请求路径/ （假设是 http://localhost:8080) 时，框架是直接放行此路径的

4.1.2）@JWPSession

作用：会话（登陆状态）状态，才给放行

```
@JWPSession //注意看这里。
@JWPCommon  //注意看这里。
@Controller
public class LoginController {

    @RequestMapping(path = "/", method = RequestMethod.GET)
    public String index() {
        return "index";
    }
```

解说：

1、 index()没有任何的权限注解，所以，采用类上的权限注解

2、 多个权限同时存在时，权重大的一方存活，权重小的一方死亡。**同样的**权重，则执行**融合。**

a) **@JWPSession （权重是 1）**与**@JWPCommon** （权重是 0） 同时存在类上，**@JWPCommon** 被覆盖掉（即，只有**@JWPSession** 生效）。

b) 只能同级别比较。（类上的权限注解，只能跟类上的其他权限注解 pk ； 方法上的 VS 方法上的）

3、 当访问请求路径/ （即 http://localhost:8080) 时

a) 如果用户已经登陆，则放行

b) 如果用户未登陆，执行拦截

4.1.3）@JWPGrades

作用：需要 xxx 等级，方可以访问

```
@JWPGrades("1") //注意看这里。
@JWPSession //注意看这里。
@JWPCommon  //注意看这里。
@Controller
public class LoginController {

    @RequestMapping(path = "/", method = RequestMethod.GET)
    public String index() {
        return "index";
    }
```

解说：

1、 index()没有任何的权限注解，所以，采用类上的权限注解

2、 **@JWPGrades（权重是 2）**,大于其他权限注解，固只有**@JWPGrades** 生效，其他的全部失效。

3、 当访问请求路径/ （即 http://localhost:8080) 时，如果用户拥有权限等级 **1**，则放行.否则拦截。

4、 **如果把 @JWPGrades("1") 换成 @JWPGrades ，则表示用户拥有任意的权限等级，都可以放行**

5、 假设你希望，只要用户拥有 1 或 2 等级都放行时，可以这样写：**@JWPGrades("1,2")**

4.1.4）@JWPCode

作用：需要 xxx 权限编号，方可以访问

```
@JWPCode("add")  //注意看这里。
//@JWPGrades("1") //注意看这里。
@JWPSession //注意看这里。
@JWPCommon  //注意看这里。
@Controller
public class LoginController {

    @RequestMapping(path = "/", method = RequestMethod.GET)
    public String index() {
        return "index";
    }
```

解说：

1、 index()没有任何的权限注解，所以，采用类上的权限注解

2、 **@JWPCode（权重是 2）**,大于其他权限注解，固只有**@JWPCode** 生效，其他的全部失效。

3、 当访问请求路径**/** （即 http://localhost:8080) 时，如果用户拥有权限编号 add，则放行.否则拦截。

4、 **@JWPCode 注解，必须要带一个值。否则报错**

5、假设你希望，只要用户拥有 add 或 update 编号都放行时，可以这样写：**@JWPCode("add,update")**

6、假设**@JWPGrades("1")** 不注释，则会与**@JWPCode("add")**融合。表示需要具备 add 编号和等级 1 的权限，才可以访问。

4.2）表达式注入权限——Controller

需要实现**接口 JWPControllerURLExpresstion**

```
public class ControllerExpression    implements    JWPControllerURLExpresstion{
    //公共区 0
    @Override
    public void setRequestURL_common(Set<String> set) {}
    //会话 1
    @Override
    public void setRequestURL_session(Set<String> set) {}
    //等级 2
    @Override
    public void setRequestURL_grades(Set<String> set) {}
    //编号 2
    @Override
    public void setRequestURL_code(Set<String> set) {
        set.add("regex:[/a-zA-Z0-9]+user/[.a-zA-Z0-9]{1,4} [[add]]");//符合此表达式的路径，会加编号 add
    }
}
```

4.2.1）正则表达式

即 java 中的本土 正则表达式。

建议：会正则表达式的，强烈建议使用 正则表达式

4.2.2）简化表达式

简化表达式的本质，还是正则表达式。只是它由 JWebPower 插在中间，先进行两次替换，再执行正则表达式检验。

第一次替换，是把`**`换成`[./a-zA-Z0-9_-}{?=&]`

第二次替换，是把`*`替换成`[.a-zA-Z0-9_-}{?=&]`

执行完替换后，再执行正则表达式检验。

4.2.3）完整路径

直接把完整的请求路径写在表达中。

对于没有实体方法绑定的请求路径，我们可以使用这种方法处理。

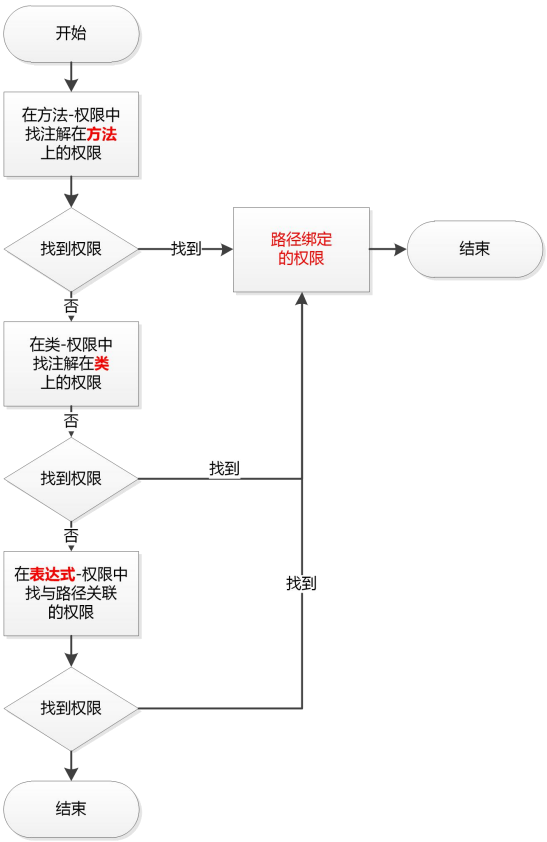
4.2.4）语法汇总

公共区 会话区	regex:正则表达式
	:简单表达式
	直接写完整的路径

权限等级	regex:正则表达式 [[等级 1,等级 2..等级 n]]
	:简单表达式 [[等级 1,等级 2..等级 n]]
	直接写完整的路径 [[等级 1,等级 2..等级 n]]

权限编号	regex:正则表达式 [[编号 1,编号 2…编号 n]]
	:简单表达式 [[编号 1,编号 2…编号 n]]
	直接写完整的路径 [[编号 1,编号 2…编号 n]]

4.3）附-给路径绑定权限的顺序


开始
在方法-权限中
找注解在方法
上的权限
找到权限
找到
路径绑定
的权限
结束
否
在类-权限中
找注解在类
上的权限
否
找到权限
找到
否
在表达式-权限中
找与路径关联
的权限
找到
找到权限
结束

五、控制区 事件触发

5.1）全局事件

需要 实现 `JWPGlobalEvent` 接口

解说：全局事件，由 JWebPower 框架首次接到用户请求时，触发的事件。
使用全局事件，我们可以手工检验路径的合法性，或需要放行的请求。

5.2）Controller 事件

需要 实现 `JWPControllerURLPowerEvent` 接口

事件区	方法	事件
起点	jWebPower_start	进入控制区时，自动调用执行的方法。必定执行
非监控区	doOther_noController	不在监控内 的请求地址。执行此方法
公共区	doCommonPower_success	通过【公共区】验证，执行此方法
会话	doSessionPower_success	通过【会话区】验证，执行此方法
	doSessionPower_fail	未通过【会话区】验证，执行此方法
等级	doGradesPower_success	通过【等级】验证，执行此方法
	doGradesPower_fail	未通【等级】过验证，执行此方法
编号	doCodePower_success	通过【权限编号】验证，执行此方法
	doCodePower_fail	未通过【权限编号】验证，执行此方法
混合区 等级+编号	doGradesAndCodePower_success	通过【权限等级、编号】验证，执行此方法
	doGradesAndCodePower_fail	未通过【权限等级、编号】验证，执行此方法

解说

1. 上述 权限检验失败事件（_fail），是没有返回值的
2. 校验成功事件（_success）、jWebPower_start、doOther_noController，有 boolean 返回值。事件方法返回 false，依然会中止请求！返回 true 才会放行。

六、**监听**

解说：实例一个监听，需要我们创建一个类，并实现接口 JWPListenInterface 然后使用@JWPListen 或@JWPRegListenUrl 注册我们的监听类

用途：多用于范围式的权限管理

比如：用户只能看自己的身份证图片或拥有指定权限的人方可以查看

6.1）两种注册方式区别

@JWPListen

1. 直接作用在控制类或其方法上
2. 多个监听类注册时，我们写的顺序即是触发监听的顺序

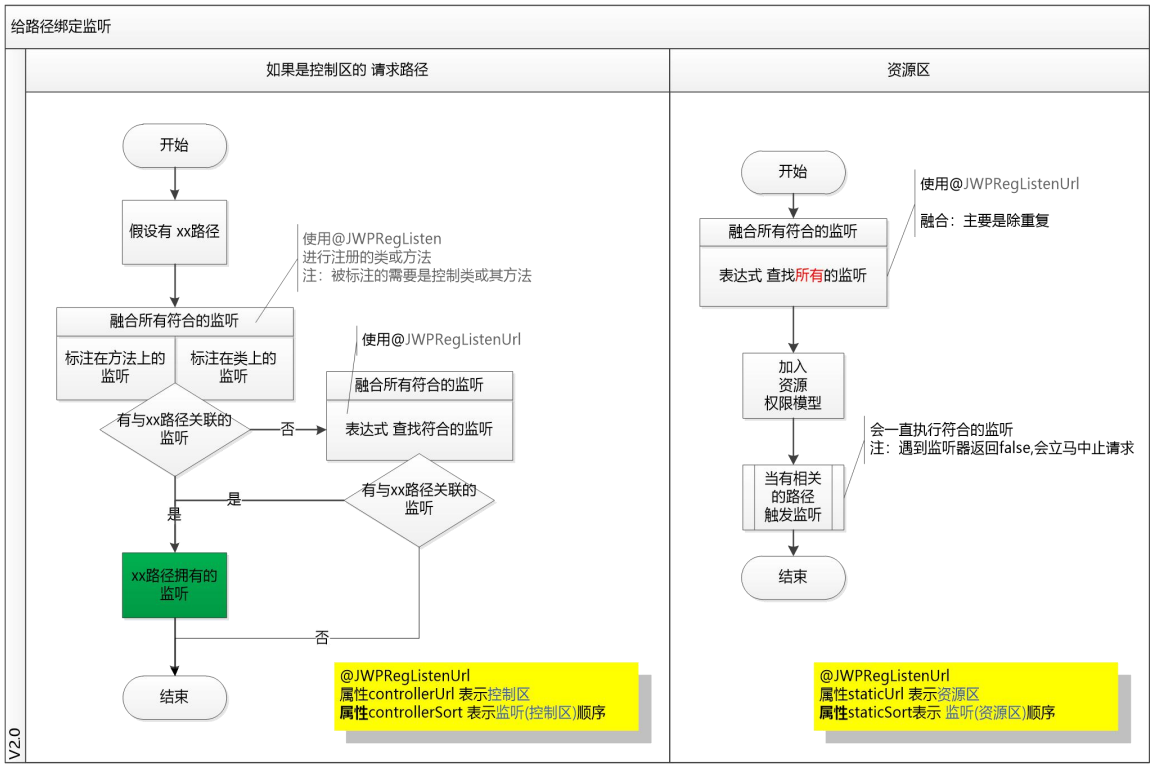
@JWPRegListenUrl

1. 直接在注解属性定义绑定监听的路径或路径表达式

regex:正则表达式
:简单表达式
直接写完整的路径

2. 除了作用于控制区，还作用于资源区（静态资源，比如 js、图片...）
   (1) 如果路径是资源区，路径锁定以**前缀**方式校验
3. 需要我们指定顺序（通过注解的属性 controllerSort 或 staticSort）

6.2）框架给路径绑定监听 流程图



七、辅助优化——前缀后缀

7.1）@JWPDecorate

作用域：控制类上。

用途：对当前类及其方法上的权限等级、编号 强制加入前缀、后缀

7.2）接口 JWPDecorateConfig

作用域：控制区

用途：与@JWPDecorate 用途一样。但没有@JWPDecorate 优先级强。

图解 两种方式加入前后缀，冲突时，**只采用@JWPDecorate** 上的

特色：相对@JWPDecorate，使用 JWPDecorateConfig 实现的优势为，批量，统一。无须在每个类上打上注解符@JWPDecorate

7.3）@JWPIgnoreDecorate

作用域：控制类的方法上，用于**对抗（拒接）** @JWPDecorate 与 JWPDecorateConfig 的实现类注入

用途：不准给此方法上的编号、等级加入前缀、后缀。

八、辅助优化——方法名即编号

8.1）开关（默认 false ,不启用）

开关位置：系统对接类中的 **boolean** methodIsCode() 方法。如需启用，返回 true 即可。

用途：当类、方法上没有任何的权限注解，**也没有@JWPIgnoreCode 注解**（新增的注解，专门用来抵抗【方法名即编号】），那么，方法名就是编号 .即等于 @JWPCode("方法名"). 示例：

假设条件成立，方法名即编号			
方法名	转成编号	等于	备注
select	select	@JWPCode("select")	1 个编号
select_update	[select , update]	@JWPCode("select , update")	2 个编号

8.2）注意项

当方法名存在下划线时，方法名会被切割。加下划线，自动切割的用途是：比如我们修改，往往也要先查询，再修改。那么，有可能某个查询方法，即是查询权限可以访问，也是修改权限也可以访问

同时，一般配合【辅助优化——前后缀】来用。因为，我们开发过程中，往往有很多同名方法。比如添加 add.... 之类

九、其他

8.1）关于控制台打印

如果把配置文件中的 jwp.print.console 设置成 true，**一定要记得，上线项目时，把其改回 false.**

即： jwp.print.console=false。因为 jwp.print.console=true 时，会使用大量的打印语句！

8.2）如何强制放行不在监控范围的路径

权限是个敏感的东西。一个不小心，就容易写出后门。所以，对于不在监控范围的请求路径，应该阻止其访问。当然，如果你非要其能被访问，那么，需要设置两个地方

1.在配置文件中，把此参数如下设置 jwp.controller.free.url=true

2.对控制区的【非监控区】事件方法 doOther_noController，重写，改方法返回值为 true

十、需求示例与实战

1、用户查询权限，才能访问 xxx 信息

2、拥有[绿钻]或[黄钻]的用户，才能访问 xxx 信息

3、12 级会员才能访问 xxx 信息

4、12 级会员，以及 12 级会员以上，才能访问 xxx 信息

5、用户身份证，只能自己看，和有指定权限的人可以看

6、12 级会员以上，并购买了 12 级会员专属 vip 礼包， 才能进入 xxx 副本

7、直接开了包年，或购买了 1 月的 A 功能和 B 功能...

包年=a 功能+b 功能+...后续的一些功能（动态，所以必定要查数据库）

8、比如流行的 token=xxxx，通过校验后，才能访问