

# Sentinel 规则持久化

---

## 一、修改order-service服务

---

修改OrderService，让其监听Nacos中的sentinel规则配置。

具体步骤如下：

### 1.引入依赖

在order-service中引入sentinel监听nacos的依赖：

```
<dependency>
  <groupId>com.alibaba.csp</groupId>
  <artifactId>sentinel-datasource-nacos</artifactId>
</dependency>
```

### 2.配置nacos地址

在order-service中的application.yml文件配置nacos地址及监听的配置信息：

```
spring:
  cloud:
    sentinel:
      datasource:
        flow:
          nacos:
            server-addr: localhost:8848 # nacos地址
            dataId: orderservice-flow-rules
            groupId: SENTINEL_GROUP
            rule-type: flow # 还可以是: degrade、authority、param-flow
```

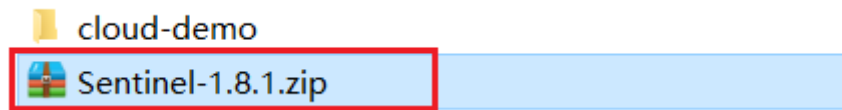
## 二、修改sentinel-dashboard源码

---

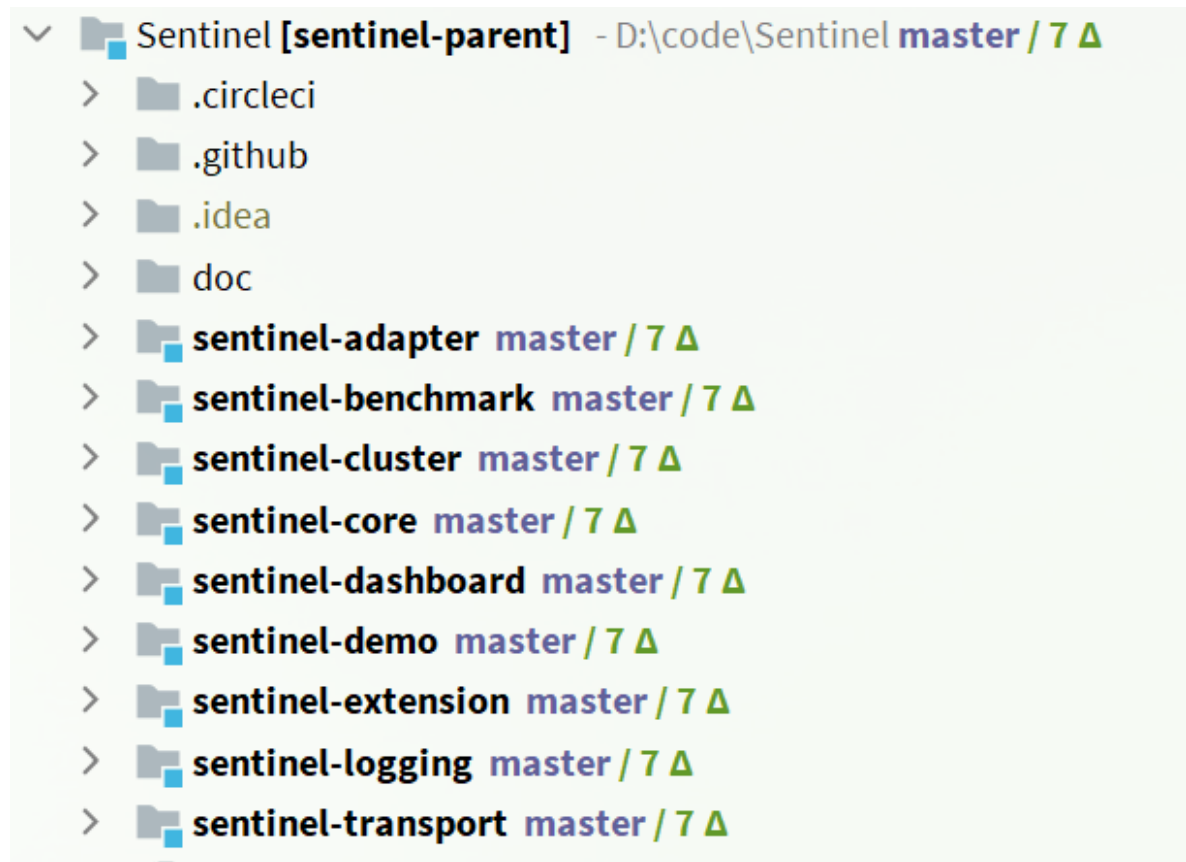
SentinelDashboard默认不支持nacos的持久化，需要修改源码。

### 1. 解压

解压课前资料中的sentinel源码包：



然后并用IDEA打开这个项目，结构如下：



## 2. 修改nacos依赖

在sentinel-dashboard源码的pom文件中，nacos的依赖默认的scope是test，只能在测试时使用，这里要去除：

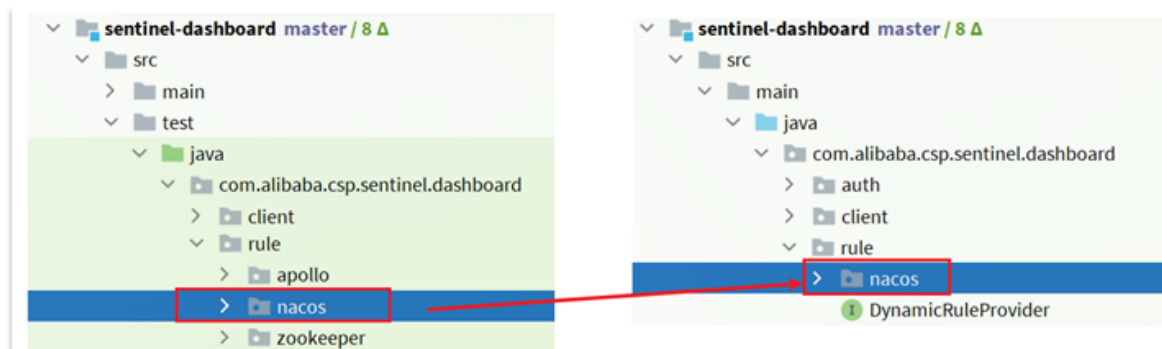
```
<!-- for Nacos rule publisher sample -->
<dependency>
  <groupId>com.alibaba.csp</groupId>
  <artifactId>sentinel-datasource-nacos</artifactId>
  <scope>test</scope>
</dependency>
```

将sentinel-datasource-nacos依赖的scope去掉：

```
<dependency>
  <groupId>com.alibaba.csp</groupId>
  <artifactId>sentinel-datasource-nacos</artifactId>
</dependency>
```

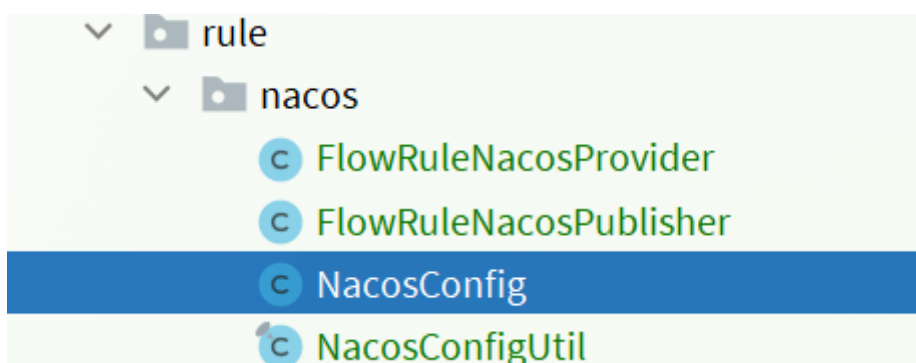
## 3. 添加nacos支持

在sentinel-dashboard的test包下，已经编写了对nacos的支持，我们需要将其拷贝到main下。



## 4. 修改nacos地址

然后，还需要修改测试代码中的NacosConfig类：



修改其中的nacos地址，让其读取application.properties中的配置：

```
@Configuration
@ConfigurationProperties(prefix = "nacos")
public class NacosConfig {
    // nacos地址
    private String addr;

    @Bean
    public ConfigService nacosConfigService() throws Exception {
        return ConfigFactory.createConfigService(addr);
    }

    public String getAddr() {
        return addr;
    }

    public void setAddr(String addr) {
        this.addr = addr;
    }

    @Bean
    public Converter<List<FlowRuleEntity>, String> flowRuleEntityEncoder() {

    }

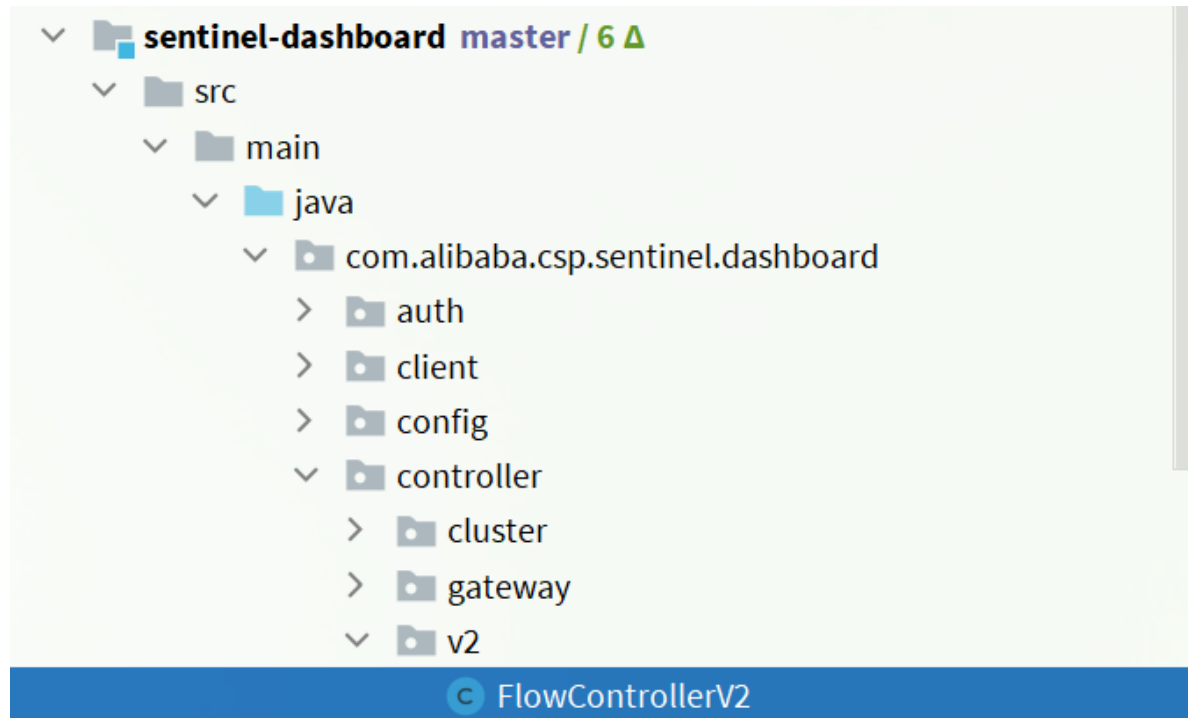
    @Bean
    public Converter<String, List<FlowRuleEntity>> flowRuleEntityDecoder() {
        return s -> JSON.parseArray(s, FlowRuleEntity.class);
    }
}
```

在sentinel-dashboard的application.properties中添加nacos地址配置：

```
nacos.addr=localhost:8848
```

## 5. 配置nacos数据源

另外，还需要修改com.alibaba.csp.sentinel.dashboard.controller.v2包下的FlowControllerV2类：



让我们添加的Nacos数据源生效：

```
@Autowired
@Qualifier("flowRuleDefaultProvider")
private DynamicRuleProvider<List<FlowRuleEntity>> ruleProvider;
@Autowired
@Qualifier("flowRuleDefaultPublisher")
private DynamicRulePublisher<List<FlowRuleEntity>> rulePublisher;

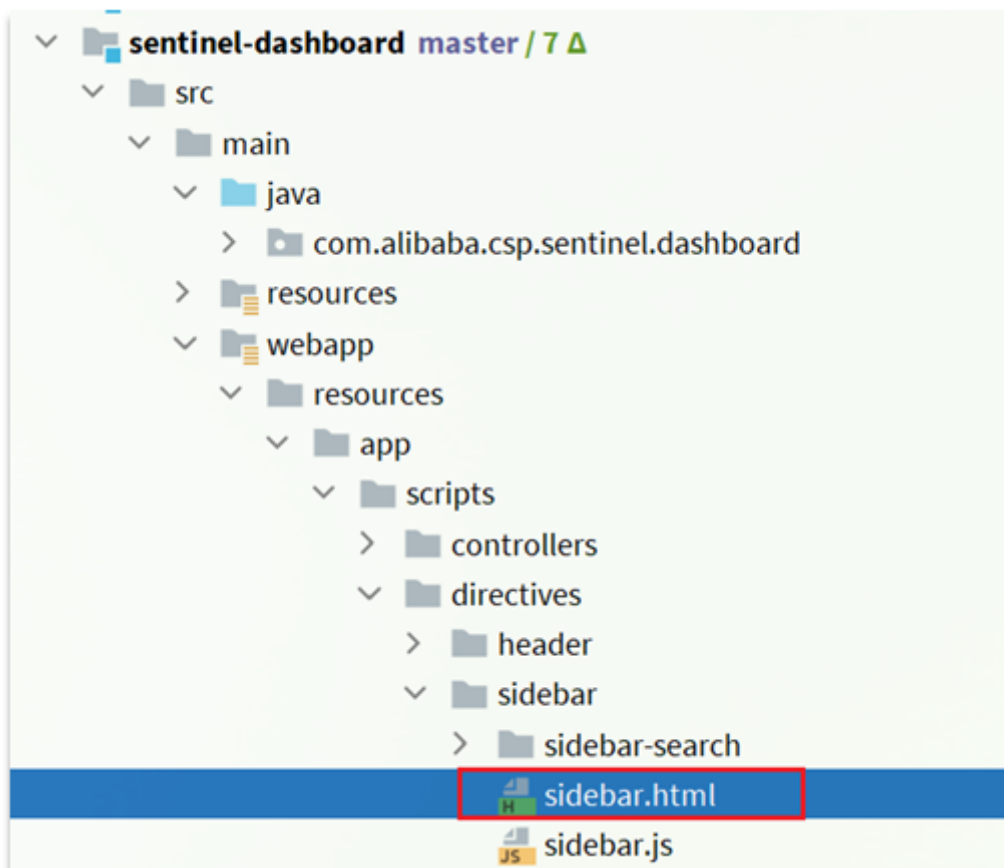
@Autowired
@Qualifier("flowRuleNacosProvider")
private DynamicRuleProvider<List<FlowRuleEntity>> ruleProvider;
@Autowired
@Qualifier("flowRuleNacosPublisher")
private DynamicRulePublisher<List<FlowRuleEntity>> rulePublisher;
```

The image shows two code blocks. The top block contains the default providers and publishers. The bottom block contains the Nacos providers and publishers. Red and blue boxes highlight the qualifier strings, and arrows indicate the mapping between them.

## 6. 修改前端页面

接下来，还要修改前端页面，添加一个支持nacos的菜单。

修改src/main/webapp/resources/app/scripts/directives/sidebar/目录下的sidebar.html文件：



将其中的这部分注释打开：

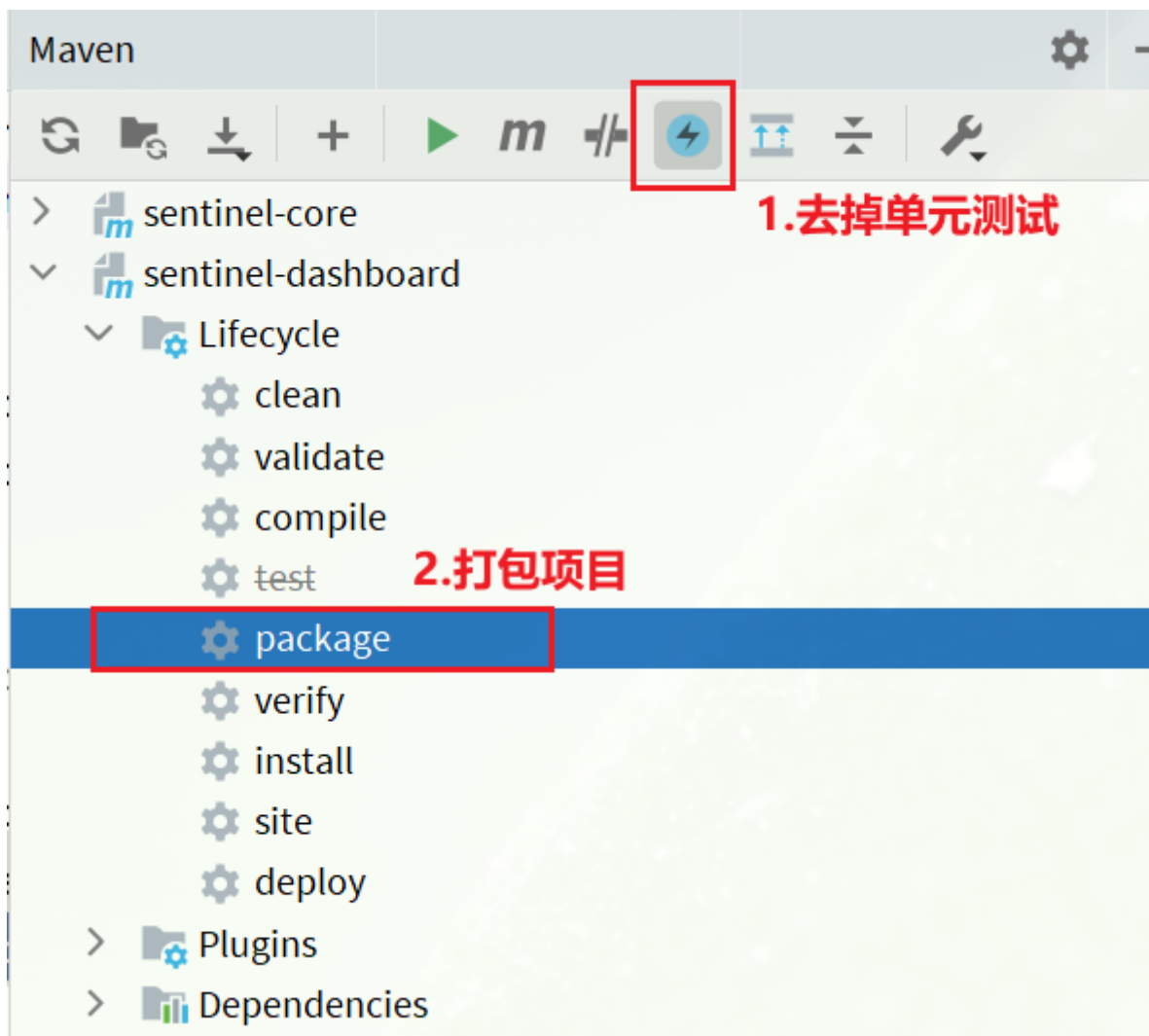
```
<!--<li ui-sref-active="active" ng-if="entry.appType==0">
  <a ui-sref="dashboard.flow({app: entry.app})">
    <i class="glyphicon glyphicon-filter"></i>&nbsp;&nbsp;&nbsp;流控规则-V1</a>
</li>-->
```

修改其中的文本：

```
<li ui-sref-active="active" ng-if="entry.appType==0">
  <a ui-sref="dashboard.flow({app: entry.app})">
    <i class="glyphicon glyphicon-filter"></i>&nbsp;&nbsp;&nbsp;流控规则-NACOS</a>
</li>
```

## 7. 重新编译、打包项目

运行IDEA中的maven插件，编译和打包修改好的Sentinel-Dashboard：



## 8.启动

启动方式跟官方一样：

```
java -jar sentinel-dashboard.jar
```

如果要修改nacos地址，需要添加参数：

```
java -jar -Dnacos.addr=localhost:8848 sentinel-dashboard.jar
```