主讲老师：Fox

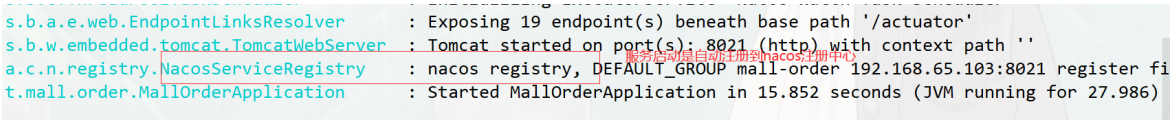
1. Spring扩展点

- BeanFactoryPostProcessor
  - BeanDefinitionRegistryPostProcessor
- BeanPostProcessor
  - InstantiationAwareBeanPostProcessor
  - AbstractAutoProxyCreator
- @Import
  - ImportBeanDefinitionRegistrar
  - ImportSelector
- Aware
- InitializingBean
- FactoryBean
- SmartInitializingSingleton
- ApplicationListener
- Lifecycle
  - SmartLifecycle
  - LifecycleProcessor
- HandlerInterceptor
- MethodInterceptor

2. Spring扩展点应用场景

2.1 整合Nacos

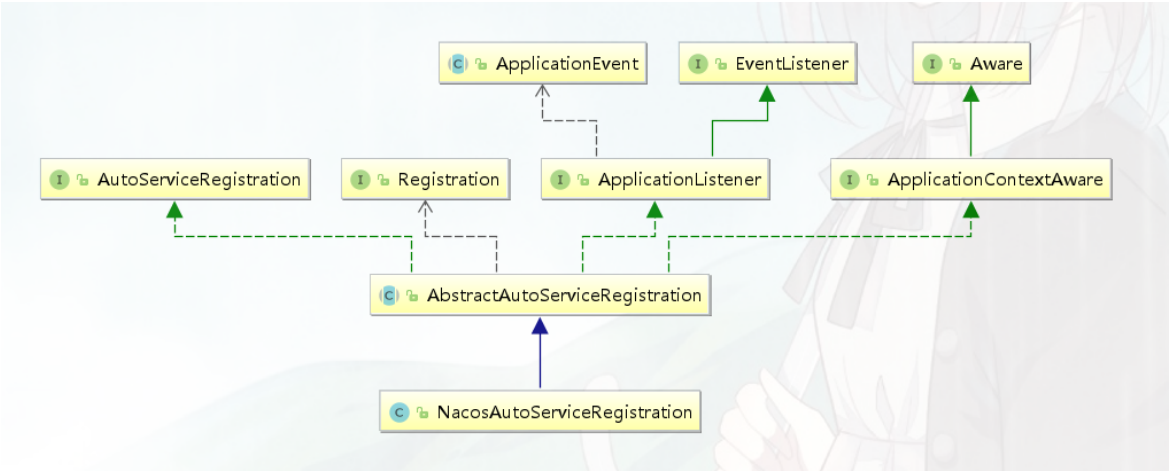
思考： 为什么整合Nacos注册中心后，服务启动就会自动注册，Nacos是如何实现自动服务注册的?



NacosAutoServiceRegistration

```
# 对ApplicationListener的扩展
AbstractAutoServiceRegistration#onApplicationEvent
# 服务注册
```

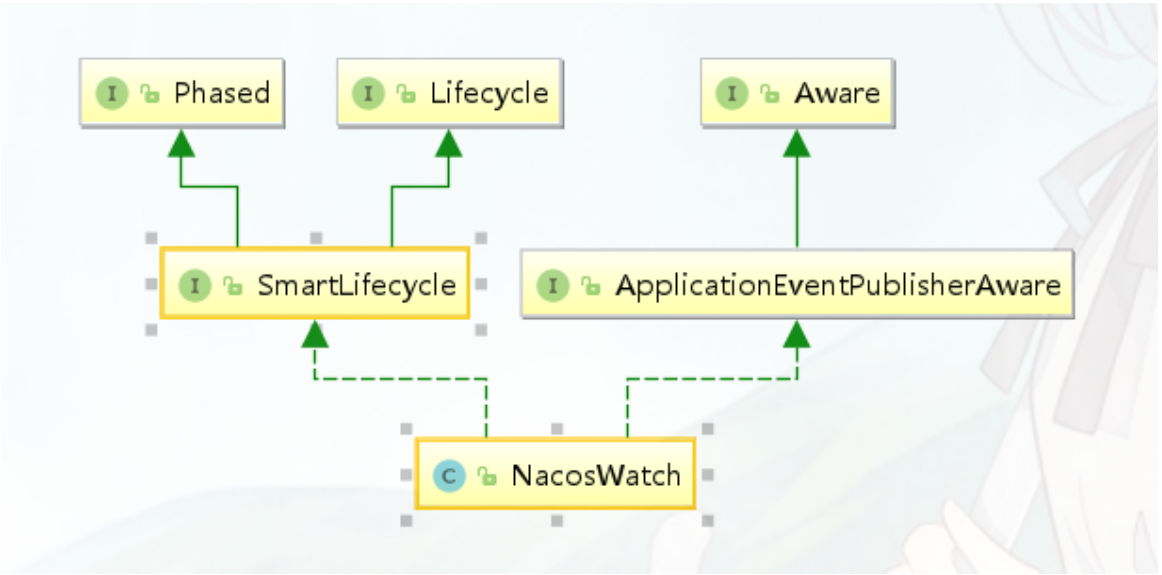
```
4 》NacosServiceRegistry#register
```



https://www.processon.com/view/link/5ea27ca15653bb6efc68eb8c

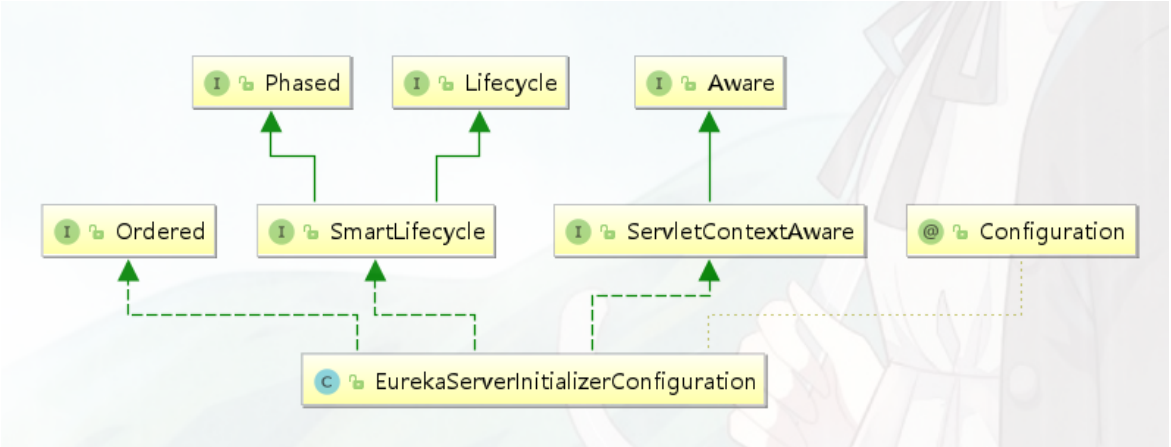
NacosWatch:

```
1 #对SmartLifecycle的扩展
2 NacosWatch#start
3 #订阅服务接收实例更改的事件
4 》NamingService#subscribe
```



扩展： Eureka Server端上下文的初始化是在SmartLifecycle#start中实现的

EurekaServerInitializerConfiguration

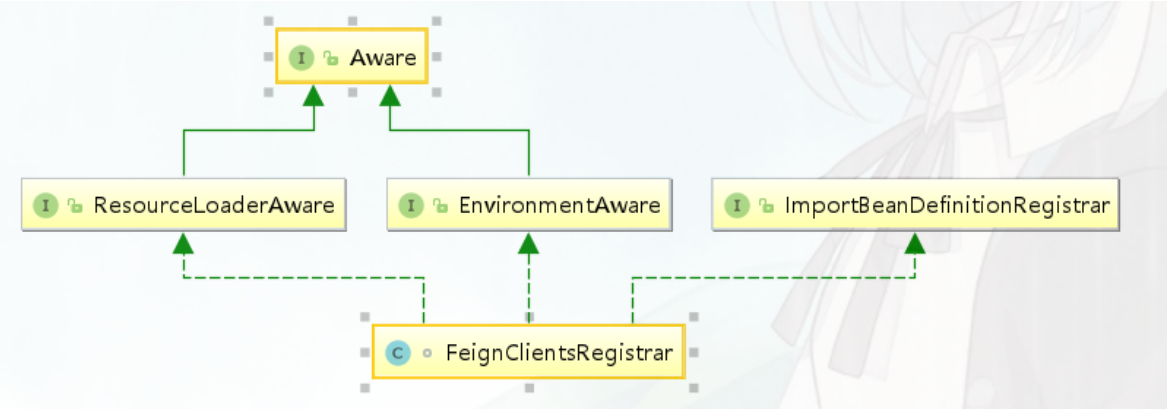


2.2 整合Feign

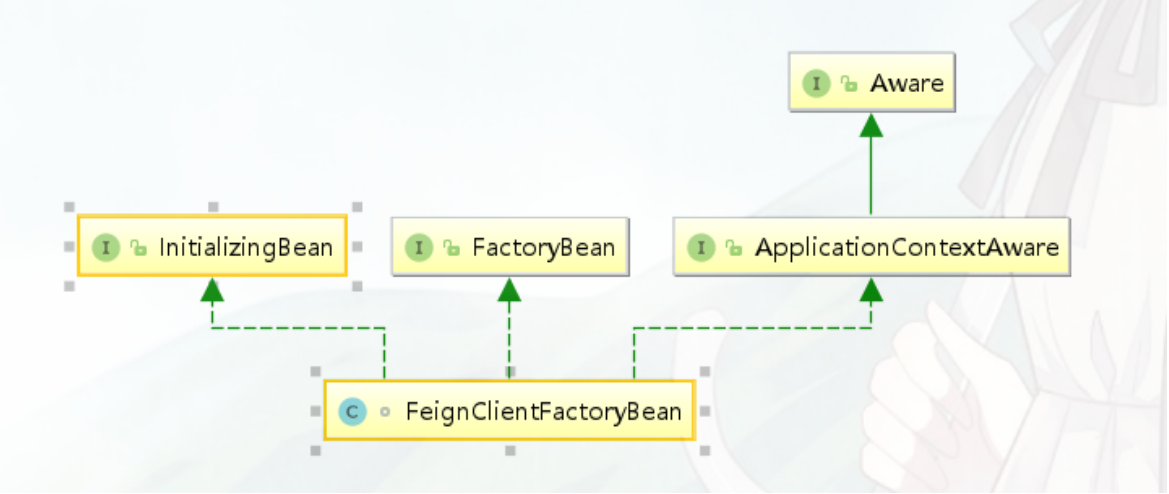
思考：为什么Feign接口可以通过@Autowired直接注入使用？Feign接口是如何交给Spring管理的?

```
@FeignClient(value = "mall-order",path = "/order")
public interface OrderFeignService {

 @RequestMapping("/findOrderByUserId/{userId}")
 R findOrderByUserId(@PathVariable("userId") Integer userId);
}

@RestController
@RequestMapping("/user")
 public class UserController {

  @Autowired
  OrderFeignService orderFeignService;

  @RequestMapping(value = "/findOrderByUserId/{id}")
  public R findOrderByUserId(@PathVariable("id") Integer id) {
  //feign调用
  R result = orderFeignService.findOrderByUserId(id);
  return result;
  }
 }
```

FeignClientsRegistrar



FeignClientFactorybean



https://www.processon.com/view/link/5e80ae79e4b03b99653fe42f

2.3 整合Ribbon

思考：为什么@Bean修饰的RestTemplate加上@LoadBalanced就能实现负载均衡功能?

```
@Bean
@LoadBalanced
public RestTemplate restTemplate() {
 return new RestTemplate();
}
```

LoadBalancerAutoConfiguration

对SmartInitializingSingleton的扩展，为所有用@LaodBalanced修饰的restTemplate（利用了@Qualifier）绑定实现了负载均衡逻辑的拦截器LoadBalancerInterceptor

```java
public class LoadBalancerAutoConfiguration {

    @LoadBalanced
    @Autowired(required = false)
    private List<RestTemplate> restTemplates = Collections.emptyList();

    @Autowired(required = false)
    private List<LoadBalancerRequestTransformer> transformers = Collections.emptyList();

    @Bean
    public SmartInitializingSingleton loadBalancedRestTemplateInitializerDeprecated(
            final ObjectProvider<List<RestTemplateCustomizer>> restTemplateCustomizers) {
        return () -> restTemplateCustomizers.ifAvailable(customizers -> {
            for (RestTemplate restTemplate : LoadBalancerAutoConfiguration.this.restTemplates) {
                for (RestTemplateCustomizer customizer : customizers) {
                    customizer.customize(restTemplate);
                }
            }
        });
    }
```

LoadBalancerInterceptor

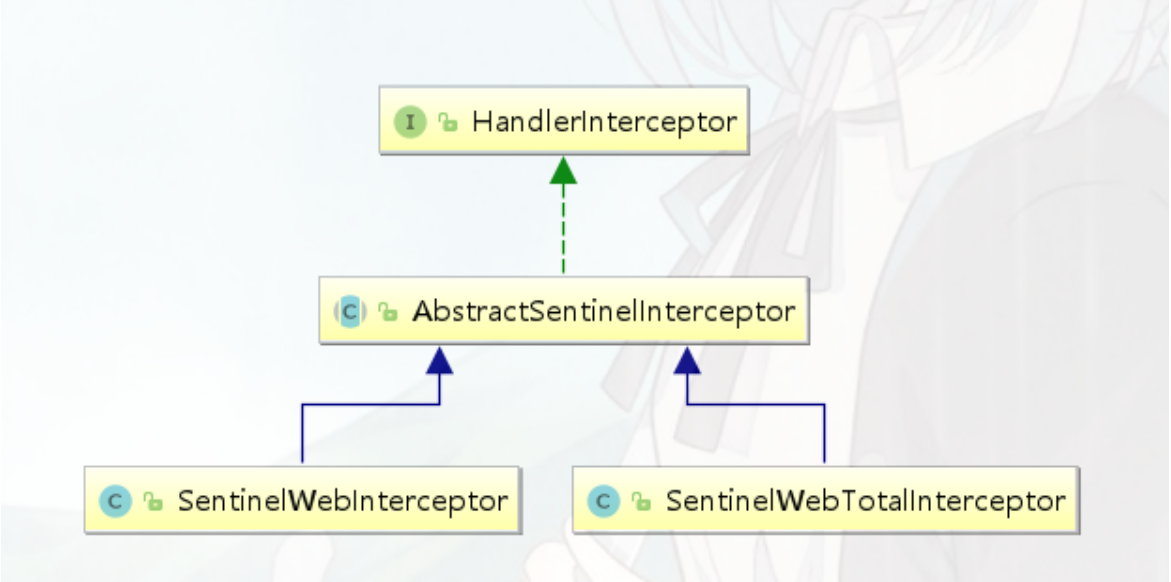
```java
@Bean
@ConditionalOnMissingBean
public RestTemplateCustomizer restTemplateCustomizer(
        final LoadBalancerInterceptor loadBalancerInterceptor) {
    return restTemplate -> {
        List<ClientHttpRequestInterceptor> list = new ArrayList<>(
                restTemplate.getInterceptors());
        list.add(loadBalancerInterceptor);
        restTemplate.setInterceptors(list);
    };
}
```

https://www.processon.com/view/link/5e7466dce4b027d999bdaddb

2.4 整合sentinel

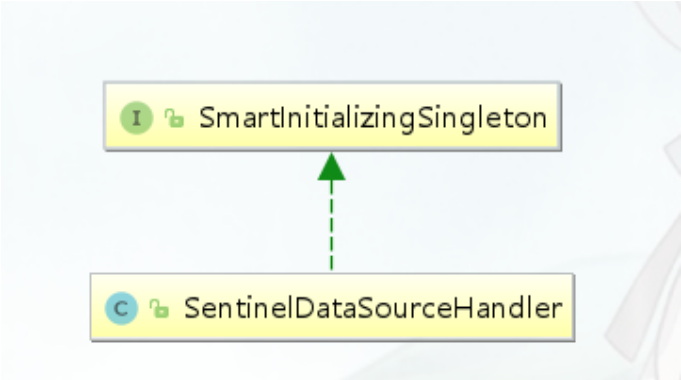
AbstractSentinelInterceptor

```
# Webmvc接口资源保护入口
AbstractSentinelInterceptor#preHandle
```

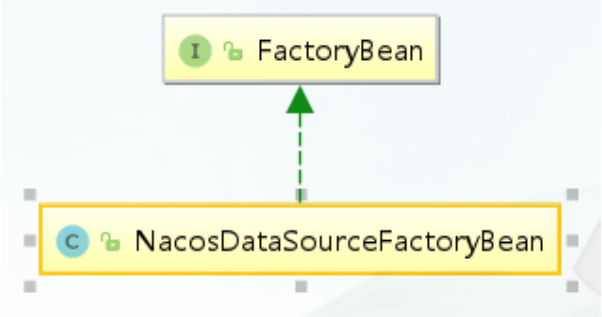


SentinelDataSourceHandler

```
#Sentinel持久化读数据源设计，利用了SmartInitializingSingleton扩展点
SentinelDataSourceHandler#afterSingletonsInstantiated
# 注册一个FactoryBean类型的数据源
》SentinelDataSourceHandler#registerBean
》》NacosDataSourceFactoryBean#getObject
# 利用FactoryBean获取到读数据源
》》new NacosDataSource(properties, groupId, dataId, converter)
```

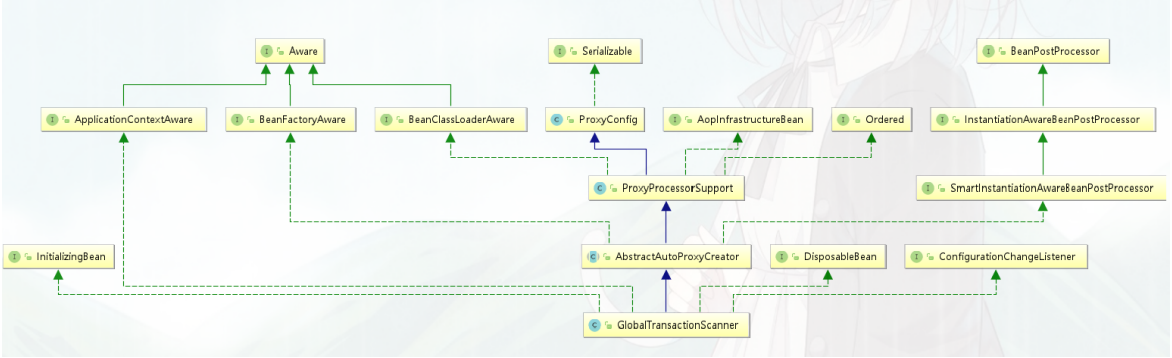


NacosDataSourceFactoryBean

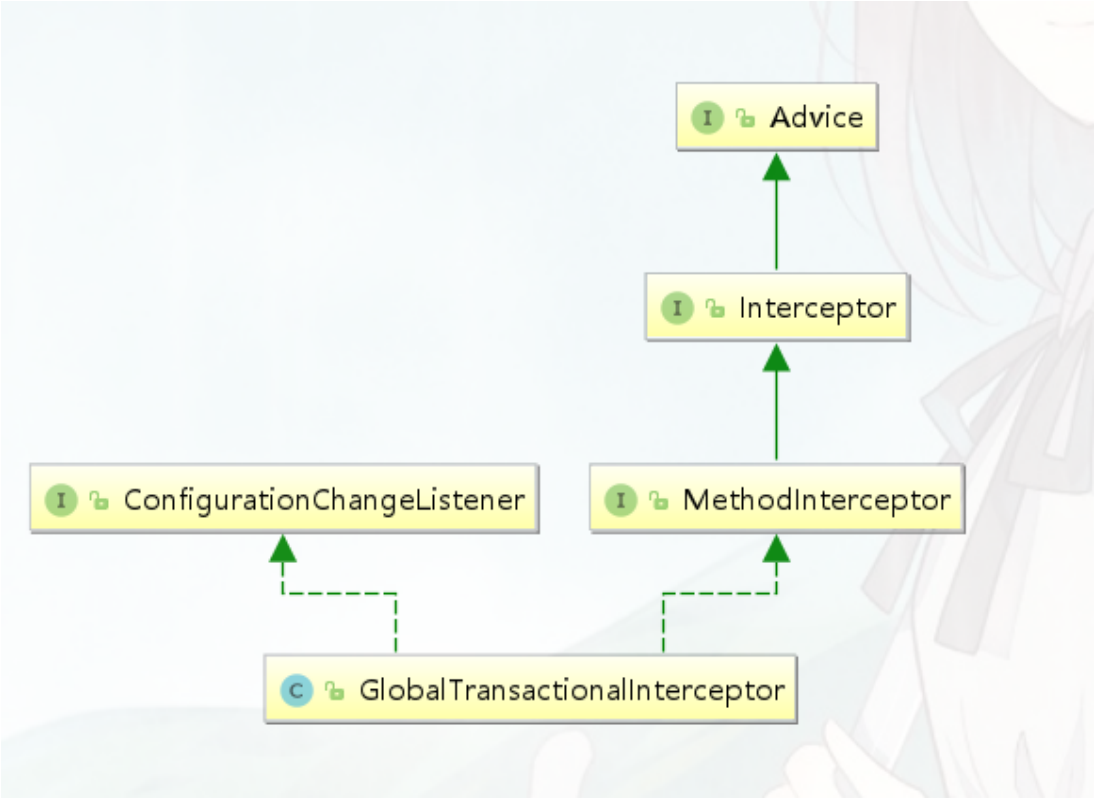


2.5 整合seata

GlobalTransactionScanner



GlobalTransactionalInterceptor



https://www.processon.com/view/link/5f743063e0b34d0711f001d2

3. Nacos配置中心源码分析

https://www.processon.com/view/link/603f3d2fe401fd641adb51f1

文档：22 Spring扩展点在微服务组件中的应用....
链接：http://note.youdao.com/noteshare?id=83d3b8e4598a2e071ae653de66832ae0&sub=8076A124D7C146B491FBDA9F9C78039E