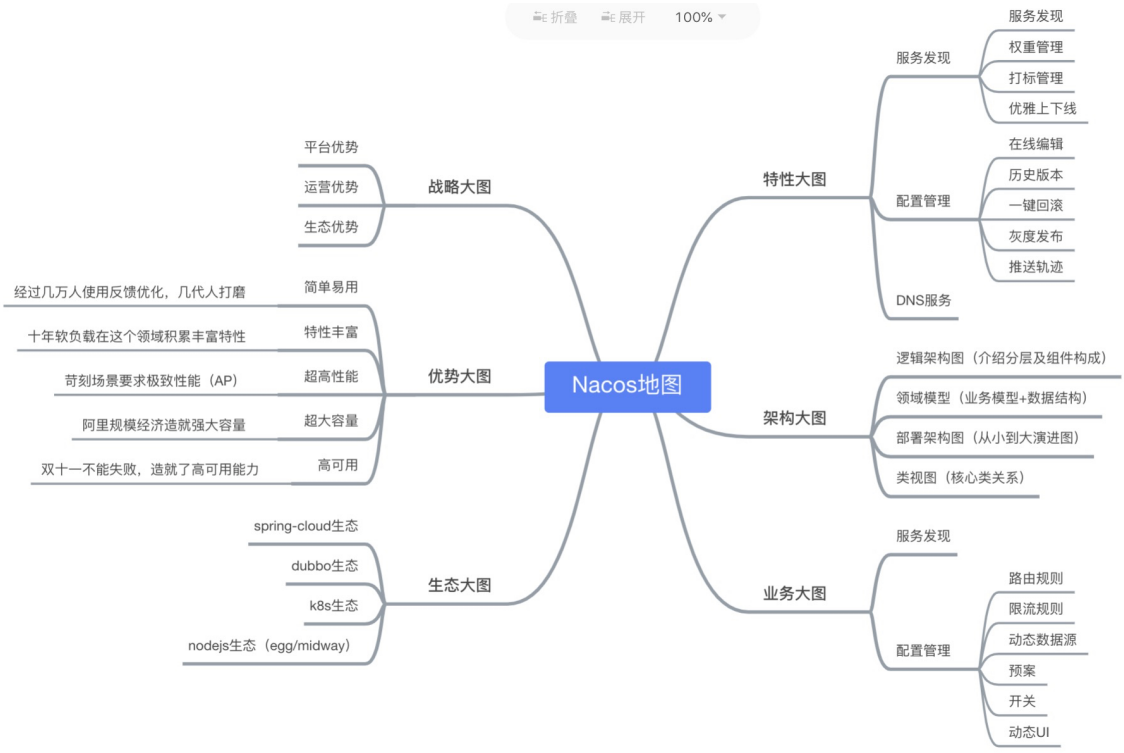
Nacos 分布式配置

什么是Nacos

Nacos 致力于帮助您发现、配置和管理微服务。Nacos 提供了一组简单易用的特性集，帮助您快速实现动态服务发现、服务配置、服务元数据及流量管理。

Nacos 帮助您更敏捷和容易地构建、交付和管理微服务平台。 Nacos 是构建以“服务”为中心的现代应用架构 (例如微服务范式、云原生范式) 的服务基础设施。

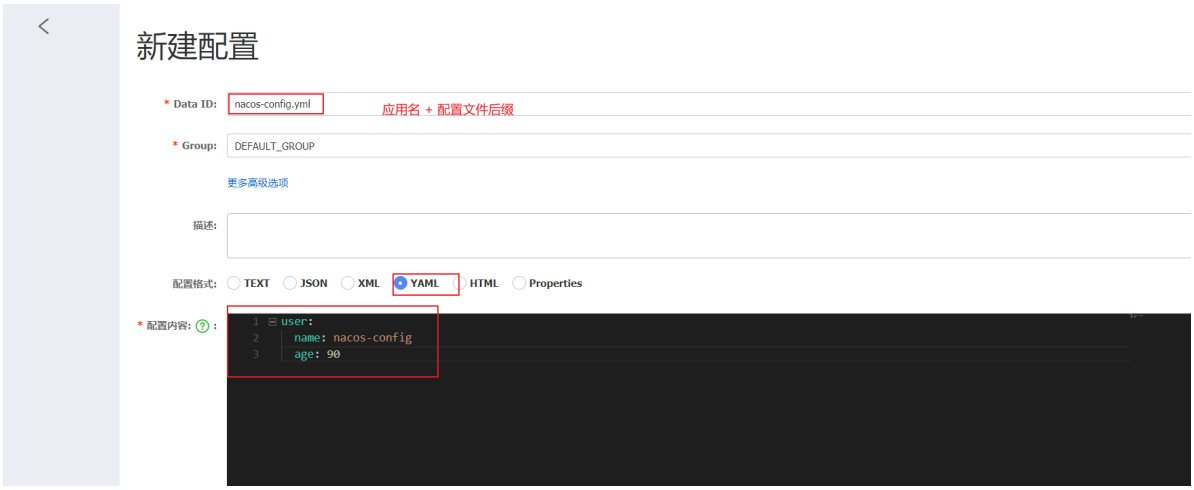


docker 部署nacos

```
docker pull nacos/nacos-server
docker run -d -p 8848:8848 --env MODE=standalone  --name nacos  nacos/nacos-server
```

访问http://localhost:8848/nacos 登录成功 nacos/nacos

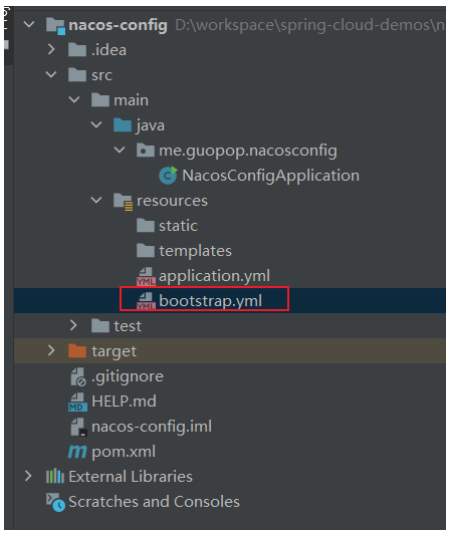
Nacos Config 入门配置



pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>me.guopop</groupId>
    <artifactId>nacos-config</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>nacos-config</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-alibaba-nacos-config</artifactId>
            <version>2.2.5.RELEASE</version>
        </dependency>
```

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>cn.hutool</groupId>
            <artifactId>hutool-all</artifactId>
            <version>5.5.9</version>
        </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

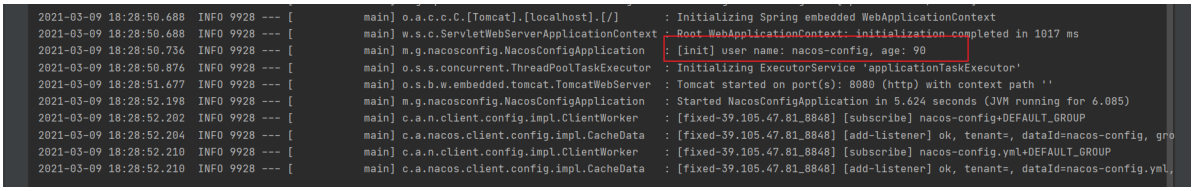


新建bootstrap.yml文件

```yaml
spring:
  application:
    name: nacos-config
  cloud:
    nacos:
      config:
        server-addr: 39.105.47.81:8848
        file-extension: yml
```

```java
package me.guopop.nacosconfig;

import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import javax.annotation.PostConstruct;

@Slf4j
@SpringBootApplication
public class NacosConfigApplication {

    @Value("${user.name}")
    private String userName;

    @Value("${user.age}")
    private int userAge;

    @PostConstruct
    public void init() {
        log.info("[init] user name: {}, age: {}", userName, userAge);
    }

    public static void main(String[] args) {
        SpringApplication.run(NacosConfigApplication.class, args);
    }
}
```

启动应用



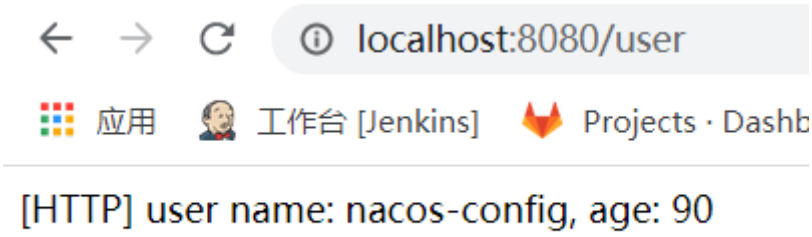
配置成功！！！

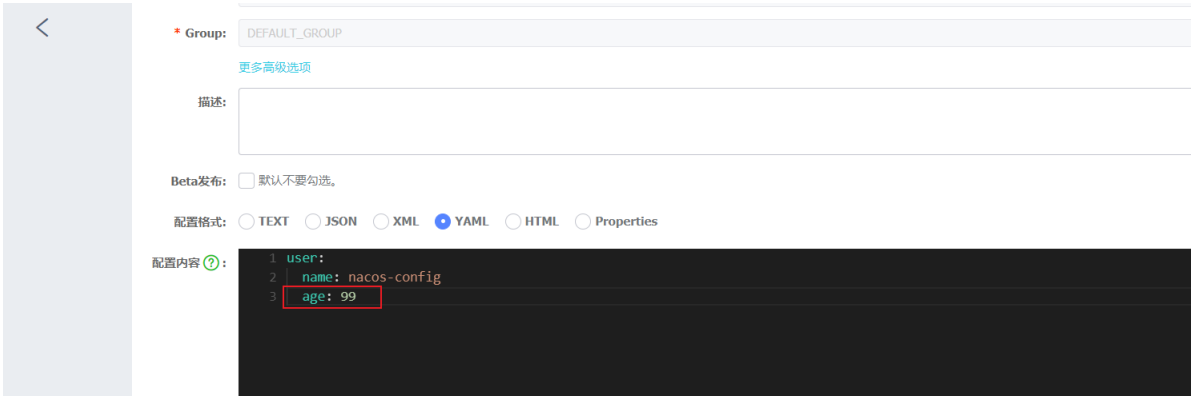
Nacos Config 实现 Bean 动态刷新

@RefreshScope + @Value实现

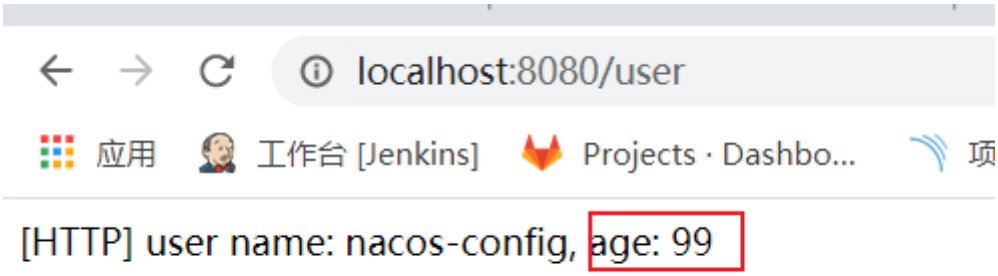
```
package me.guopop.nacosconfig;

import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.context.config.annotation.RefreshScope;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import javax.annotation.PostConstruct;

@RestController
@RefreshScope
@Slf4j
@SpringBootApplication
public class NacosConfigApplication {

    @Value("${user.name}")
    private String userName;

    @Value("${user.age}")
    private int userAge;

    @PostConstruct
    public void init() {
        log.info("[init] user name: {}, age: {}", userName, userAge);
    }

    @GetMapping("/user")
    public String user() {
        return String.format("[HTTP] user name: %s, age: %d", userName,
userAge);
    }

    public static void main(String[] args) {
        SpringApplication.run(NacosConfigApplication.class, args);
    }
}
```

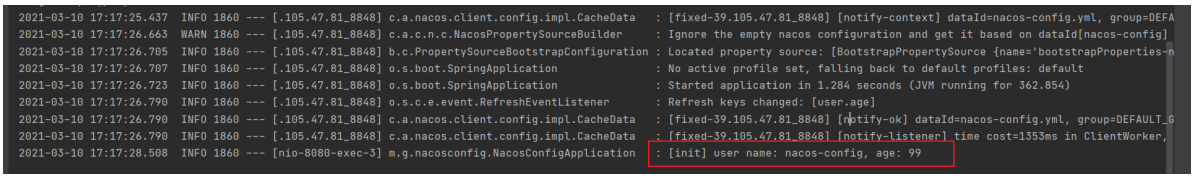
访问http://localhost:8080/user





修改配置，进行发布





数据得到刷新

@RefreshScope + @ConfigurationProperties 实现

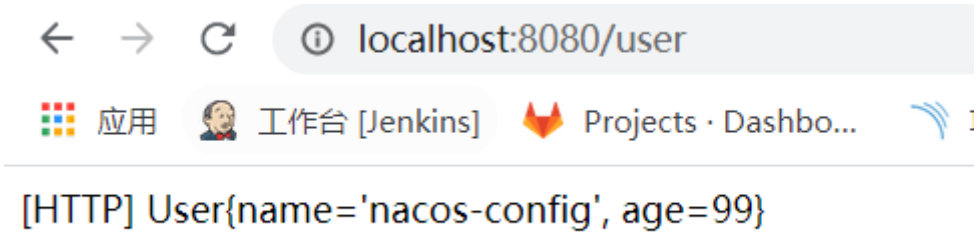
```
package me.guopop.nacosconfig;

import lombok.Getter;
import lombok.Setter;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.cloud.context.config.annotation.RefreshScope;

/**
 * @author guopop
 * @date 2021/3/10 17:26
 */
@Getter
@Setter
@RefreshScope
@ConfigurationProperties(prefix = "user")
public class User {
    private String name;

    private int age;

    @Override
    public String toString() {
        return "User{" +
                "name='" + name + '\'' +
```

```
                ", age=" + age +
                '}';
    }
}
```

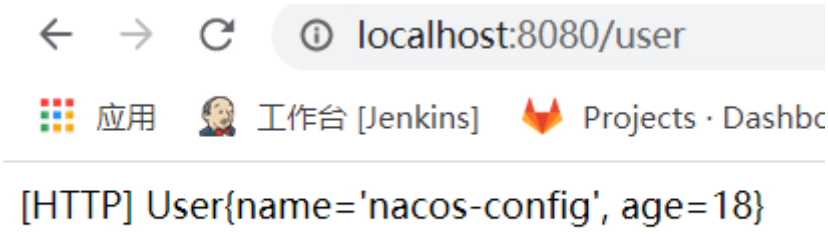
```
package me.guopop.nacosconfig;

import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.cloud.context.config.annotation.RefreshScope;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;

@EnableConfigurationProperties(User.class)
@RestController
@RefreshScope
@Slf4j
@SpringBootApplication
public class NacosConfigApplication {

    @Value("${user.name}")
    private String userName;

    @Value("${user.age}")
    private int userAge;

    @Autowired
    private User user;

    @PostConstruct
    public void init() {
        log.info("[init] user name: {}, age: {}", userName, userAge);
    }

    @PreDestroy
    public void destroy() {
        log.info("[destroy] user name: {}, age: {}", userName, userAge);
    }

    @GetMapping("/user")
    public String user() {
//      return String.format("[HTTP] user name: %s, age: %d", userName,
userAge);
        return "[HTTP] " + user;
    }

    public static void main(String[] args) {
        SpringApplication.run(NacosConfigApplication.class, args);
    }
```

```
}
```

访问http://localhost:8080/user



修改配置



刷新成功！！！

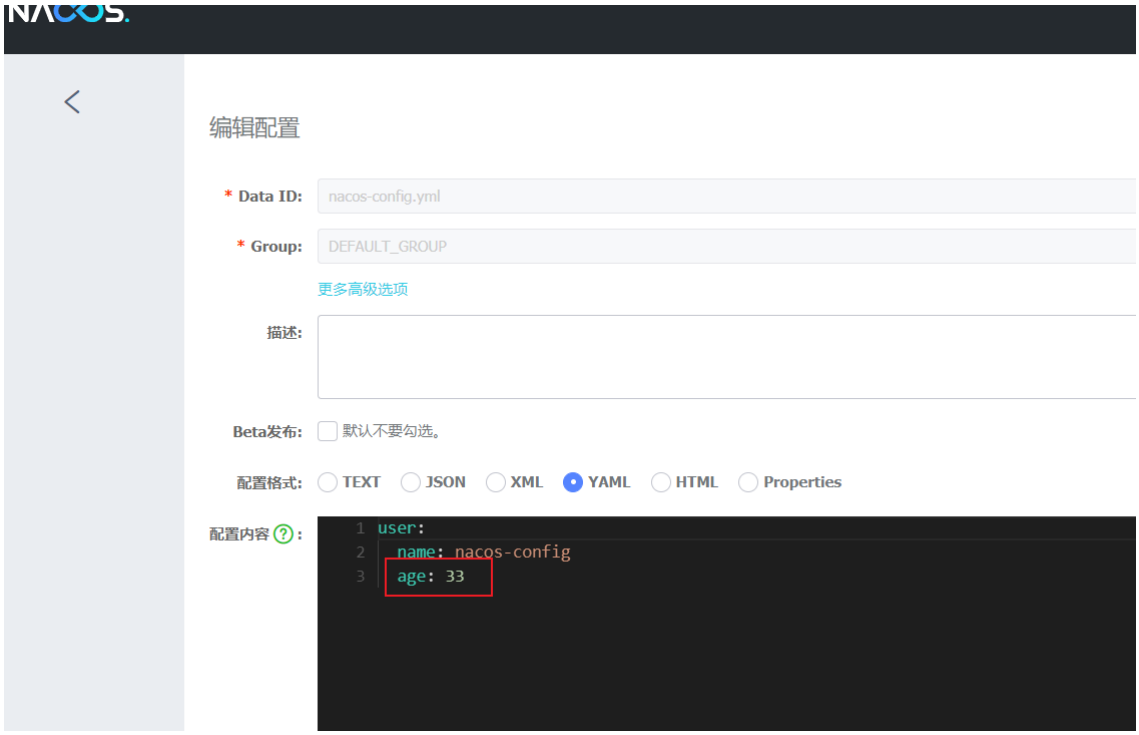
Nacos Config 监听实现Bean属性动态刷新

```
package me.guopop.nacosconfig;

import cn.hutool.core.util.StrUtil;
import cn.hutool.json.JSONUtil;
import com.alibaba.cloud.nacos.NacosConfigManager;
import com.alibaba.nacos.api.config.listener.AbstractListener;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.ApplicationRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.context.properties.EnableConfigurationProperties;
```

```
import org.springframework.cloud.context.config.annotation.RefreshScope;
import org.springframework.context.annotation.Bean;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import org.yaml.snakeyaml.Yaml;

import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import java.io.StringReader;

@EnableConfigurationProperties(User.class)
@RestController
@RefreshScope
@Slf4j
@SpringBootApplication
public class NacosConfigApplication {

    @Value("${user.name}")
    private String userName;

    @Value("${user.age}")
    private int userAge;

    @Autowired
    private User user;

    @Autowired
    private NacosConfigManager nacosConfigManager;

    @Bean
    public ApplicationRunner runner() {
        return args -> {
            String dataId = "nacos-config.yml";
            String group = "DEFAULT_GROUP";
            nacosConfigManager.getConfigService().addListener(dataId, group, new
AbstractListener() {
                @Override
                public void receiveConfigInfo(String s) {
                    log.info("[Listener] {}", s);
                    log.info("[Before User] {}", user);
                    Yaml yaml = new Yaml();
                    String dump = yaml.dump(yaml.load(new StringReader(s)));
                    log.info("[Yaml] {}", dump);
                    String s1 = StrUtil.subAfter(dump, ":", false);
                    log.info("[Sub Yaml] {}", s1);

                    user = JSONUtil.toBean(s1, User.class);

                    log.info("[After User] {}", user);
                }
            });
        };
    }

    @PostConstruct
    public void init() {
        log.info("[init] user name: {}, age: {}", userName, userAge);
    }
```

```
    @PreDestroy
    public void destroy() {
        log.info("[destroy] user name: {}, age: {}", userName, userAge);
    }

    @GetMapping("/user")
    public String user() {
//      return String.format("[HTTP] user name: %s, age: %d", userName,
userAge);
        return "[HTTP] " + user;
    }

    public static void main(String[] args) {
        SpringApplication.run(NacosConfigApplication.class, args);
    }
}
```

修改配置




```
2021-03-11 09:14:10.407  INFO 17012 --- [.105.47.81_8848] m.g.nacosconfig.NacosConfigApplication   : [Before User] User{name='nacos-config', age=22}
2021-03-11 09:14:10.428  INFO 17012 --- [.105.47.81_8848] m.g.nacosconfig.NacosConfigApplication   : [Yaml] user: {name: nacos-config, age: 33}

2021-03-11 09:14:10.434  INFO 17012 --- [.105.47.81_8848] m.g.nacosconfig.NacosConfigApplication   : [Sub Yaml]  {name: nacos-config, age: 33}

2021-03-11 09:14:10.534  INFO 17012 --- [.105.47.81_8848] m.g.nacosconfig.NacosConfigApplication   : [After User] User{name='nacos-config', age=33}
```

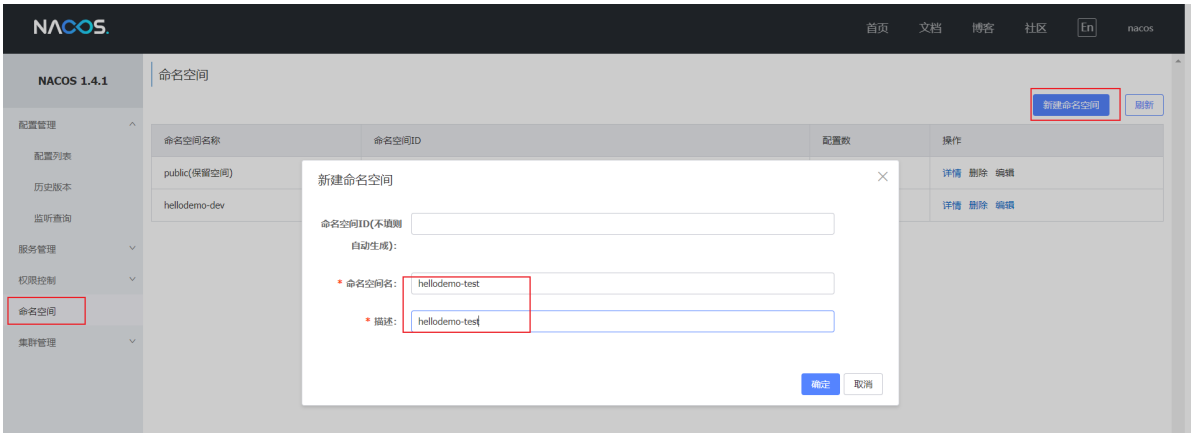
刷新成功！！！

Nacos Config 高级配置

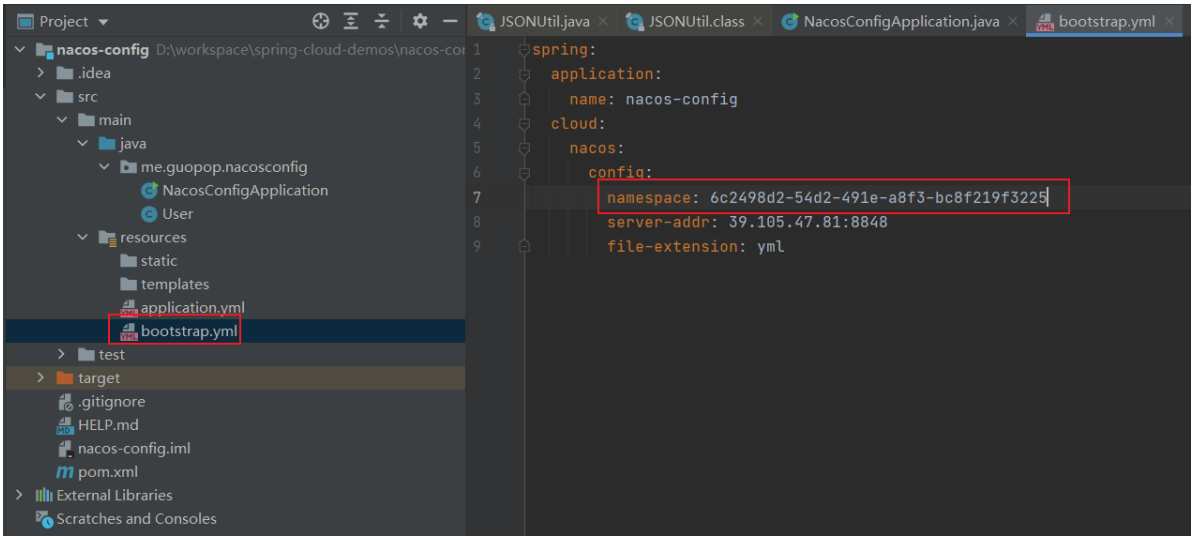
支持自定义 namespace 配置

Namespace 的常用场景之一是不同环境的配置的区分隔离

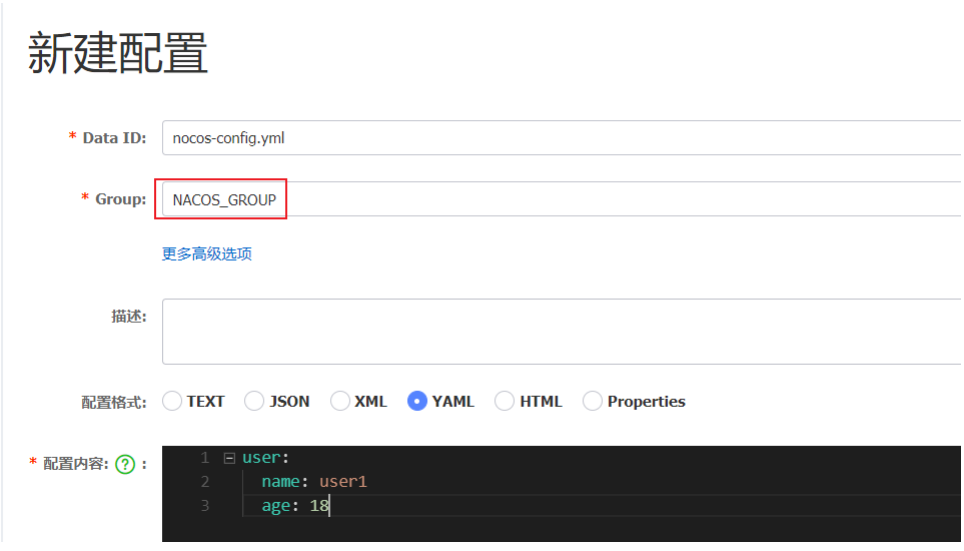
创建namespace



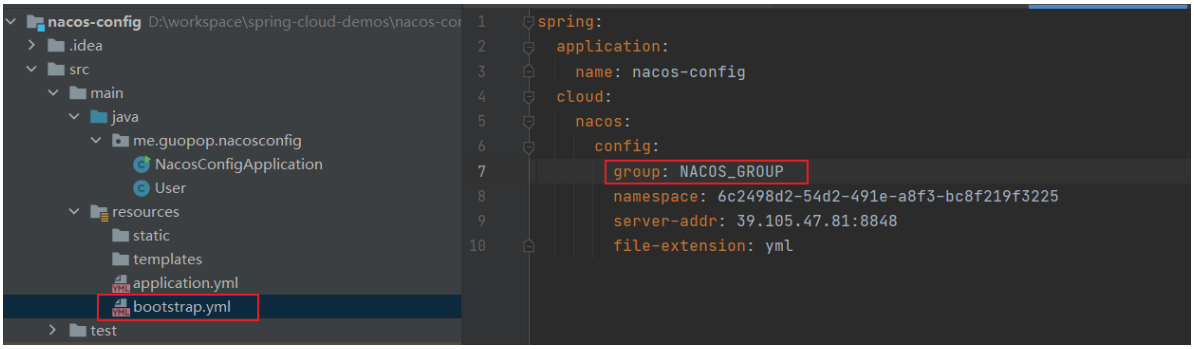
配置到代码



支持自定义Group配置



配置到配置



支持自定义扩展的Data Id配置

Todo

Nacos Config 详细配置

配置项	Key	默认值	说明
服务端地址	spring.cloud.nacos.config.server-addr		Nacos Server 启 动监听的 ip 地址和 端口
配置对应的 DataId	spring.cloud.nacos.config.name		先取 prefix，再取 name，最后取 spring.application.name
配置对应的 DataId	spring.cloud.nacos.config.prefix		先取 prefix，再取 name，最后取 spring.application.name
配置内容编码	spring.cloud.nacos.config.encode		读取的配置内容对应的编码
GROUP	spring.cloud.nacos.config.group	DEFAULT_GROUP	配置对应的组
文件扩展名	spring.cloud.nacos.config.fileExtension	properties	配置项对应的文件 扩展名，目前支持 properties 和 yaml(yml)
获取配置超时时间	spring.cloud.nacos.config.timeout	3000	客户端获取配置的 超时时间(毫秒)
接入点	spring.cloud.nacos.config.endpoint		地域的某个服务的 入口域名，通过此域 名可以动态地拿到服务端地址
命名空间	spring.cloud.nacos.config.namespace		常用场景之一是不 同环境的配置的区 分隔离，例如开发测 试环境和生产环境 的资源（如配置、服 务）隔离等
AccessKey	spring.cloud.nacos.config.accessKey		当要上阿里云时，阿 里云上面的一个云账号名
SecretKey	spring.cloud.nacos.config.secretKey		当要上阿里云时，阿 里云上面的一个云账号密码
Nacos Server 对应的 context path	spring.cloud.nacos.config.contextPath		Nacos Server 对外暴露的 context path
集群	spring.cloud.nacos.config.clusterName		配置成Nacos 集群 名称
共享配置	spring.cloud.nacos.config.sharedDataids		共享配置的 Datal d, "," 分割
共享配置动态刷新	spring.cloud.nacos.config.refreshableDataids		共享配置中需要动态刷新的 DataId, "," 分割
自定义 Data Id 配置	spring.cloud.nacos.config.extConfig		属性是个集合，内部 由 ConfigPOJO 组成。Config 有 3 个属性，分别是 dataId, group以及 refresh

Nacos Config Actuator Endpoint

Todo