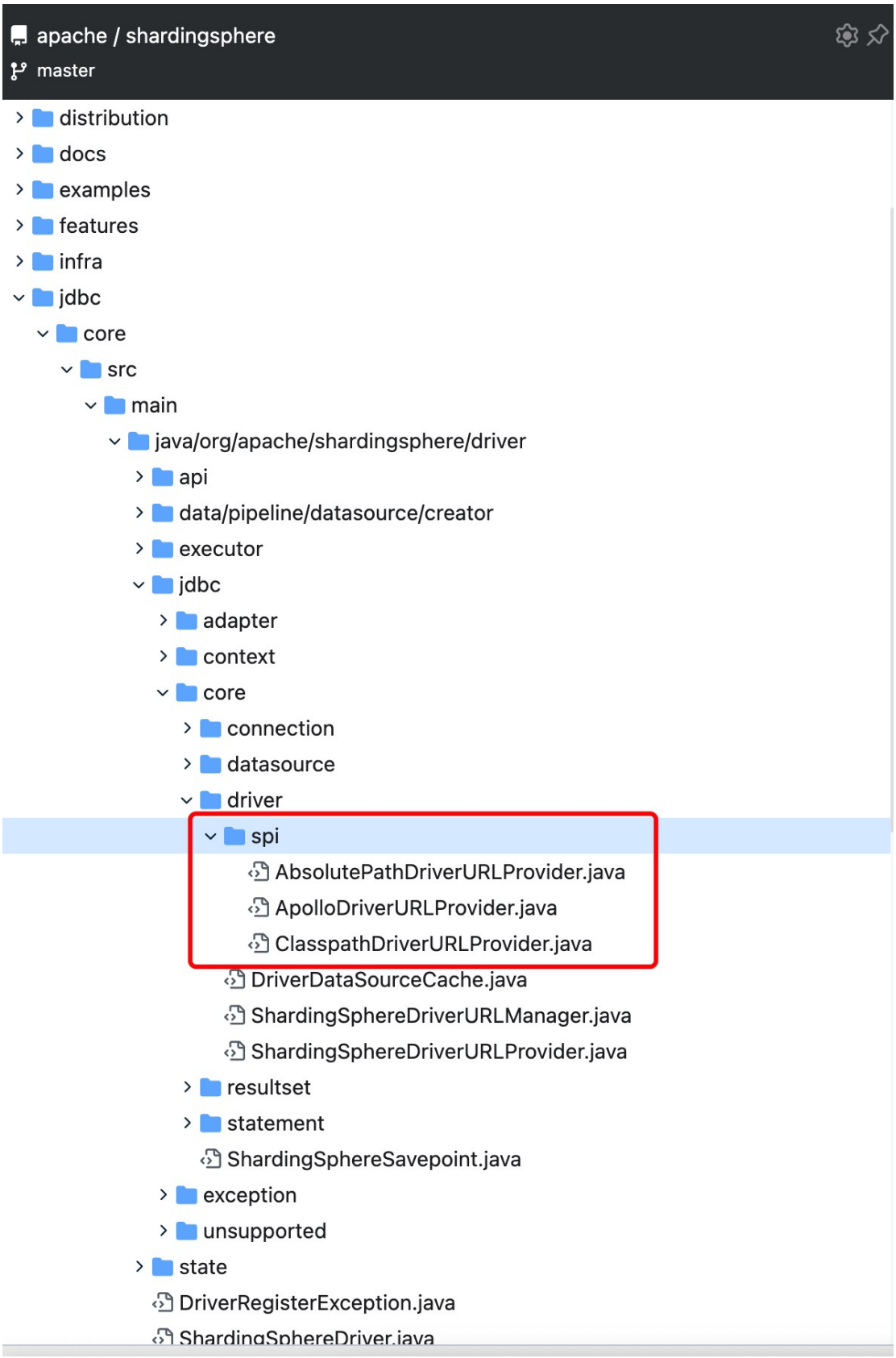
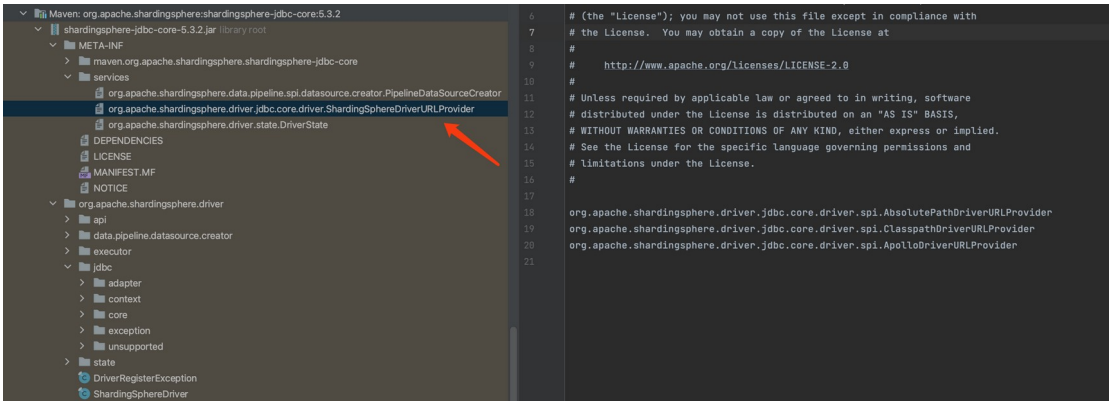
6-7 | 基于 SPI 机制修改 ShardingJDBC 底层，实现 Nacos 配置数据源

目前最新版本的 ShardingJdbc 中，没有支持基于 Nacos 作为配置数据源的功能，所以这块老师决定带着大家一起来对 ShardingJdbc 进行二次开发，使其可以支持 Nacos 的动态配置功能。


apache / shardingsphere
master
distribution
docs
examples
features
infra
jdbc
core
src
main
java/org/apache/shardingsphere/driver
api
data/pipeline/datasource/creator
executor
jdbc
adapter
context
core
connection
datasource
driver
spi
AbsolutePathDriverURLProvider.java
ApolloDriverURLProvider.java
ClasspathDriverURLProvider.java
DriverDataSourceCache.java
ShardingSphereDriverURLManager.java
ShardingSphereDriverURLProvider.java
resultset
statement
ShardingSphereSavepoint.java
exception
unsupported
state
DriverRegisterException.java
ShardingSphereDriver.java

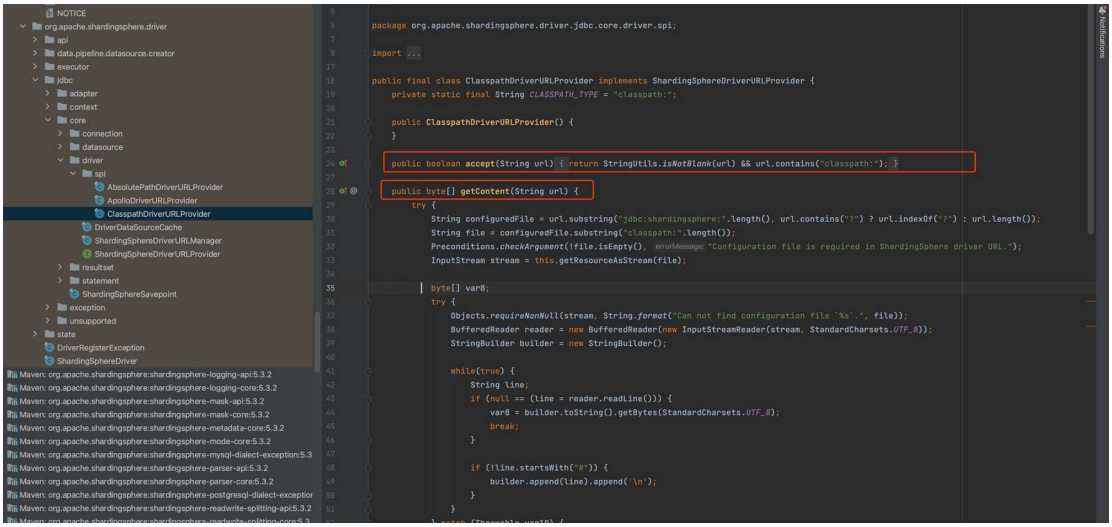
修改思路介绍

这里需要翻看 ShardingJDBC 的底层源代码，

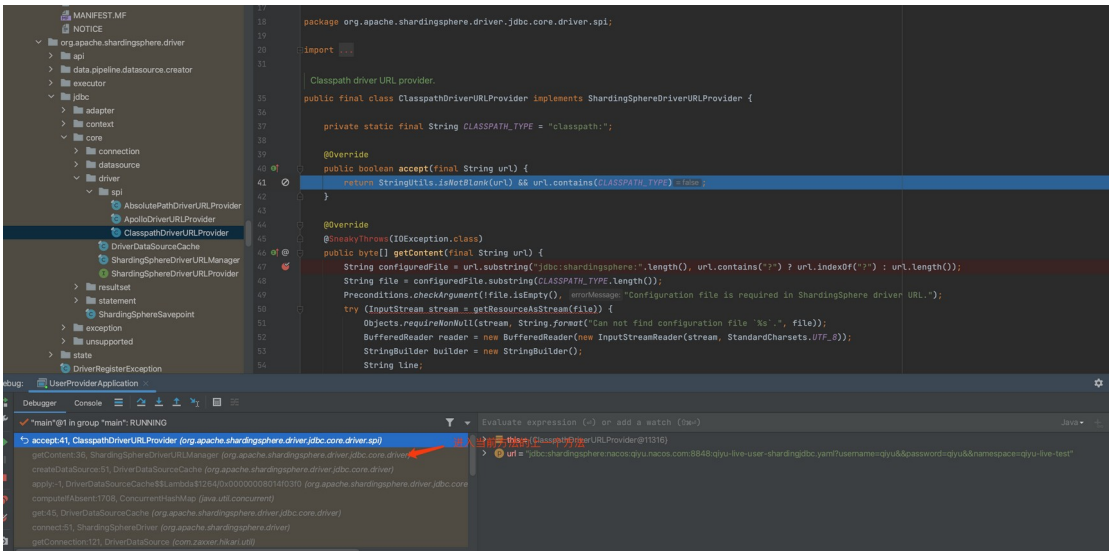


从它的 jar 包中可以发现，有些类似于 spi 机制的文件。（开始推测，这个地方是不是可以做二次开发）

接着，我们根据这份 spi 文件内部记录的类名，可以深入进行观察，发现它们都存在相同的接口。在接口中定义了 accept 和 getContent 函数，这两个函数从实现类的逻辑上看，感觉是在根据 url 的格式去判断用哪个 URLProvider 读取配置。



为了验证这个逻辑是否正确，我们需要在 ClasspathDriverURLProvider 的 accept 处加入断点进行观察。



这里你会发现，在 org.apache.shardingsphere.driver.jdbc.core.driver.ShardingSphereDriverURLManager 中有一个 for 循环，它会将 spi 文件中的所有类都进行一次校验，如果 accept 返回成功，那么就会使用匹配的 URLProvider 对象去进行配置的进一步读取。代码如下图所示：

所以基本上，我们看到了这里，相信大家基本上可以生成设计思路了，自定义一个扩展类，也是采用 SPI 的思路去实现，在新定义的扩展类中实现对 nacos 的相关配置读取，然后在 getContent 函数中返回出去。

基于 ShardingSphereDriverURLProvider 接口实现 SPI 扩展类

在 qiyu-live-framework-datasource-starter 模块中，编写一个基于 Nacos 配置的类，代码内容如下：

```
SQL
package org.idea.qiyu.live.framework.datasource.starter.config;

import com.alibaba.nacos.api.NacosFactory;
import com.alibaba.nacos.api.PropertyKeyConst;
import com.alibaba.nacos.api.config.ConfigService;
import com.alibaba.nacos.api.exception.NacosException;
```

```
import org.apache.commons.lang3.StringUtils;
import
org.apache.shardingsphere.driver.jdbc.core.driver.ShardingSphereDr
iverURLProvider;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.Properties;

/**
 * @Author idea
 * @Date: Created in 20:51 2023/6/4
 * @Description
 */
public class NacosDriverURLProvider implements
ShardingSphereDriverURLProvider {

    private static Logger logger =
LoggerFactory.getLogger(NacosDriverURLProvider.class);
    private static final String NACOS_TYPE = "nacos:";
    private static final String GROUP = "DEFAULT_GROUP";

    @Override
    public boolean accept(String url) {
        return StringUtils.isNotBlank(url) &&
url.contains(NACOS_TYPE);
    }

    /**
     * 从 url 中获取到 nacos 的连接配置信息
     *
     * @param url
（jdbc:shardingsphere:nacos:qiyu.nacos.com:8848:qiyu-live-user-
shardingjdbc.yaml?username=qiyu&&password=qiyu&&namespace=qiyu-
live-test）
     * @return
     */
    @Override
    public byte[] getContent(final String url) {
        if (StringUtils.isEmpty(url)) {
            return null;
        }
        //得到例如：qiyu.nacos.com:8848:qiyu-live-user-
shardingjdbc.yaml?username=qiyu&&password=qiyu&&namespace=qiyu-
```

```
live-test 格式的url
        String nacosUrl =
url.substring(url.lastIndexOf(NACOS_TYPE) + NACOS_TYPE.length());
        /**
         * 得到三个字符串，分别是：
         * qiyu.nacos.com
         * 8848
         * qiyu-live-user-shardingjdbc.yaml
         */
        String nacosStr[] = nacosUrl.split(":");
        String nacosFileStr = nacosStr[2];
        /**
         * 得到两个字符串
         * qiyu-live-user-shardingjdbc.yaml
         * username=qiyu&&password=qiyu&&namespace=qiyu-live-test
         */
        String nacosFileProp[] = nacosFileStr.split("\\?");
        String dataId = nacosFileProp[0];
        String acceptProp[] = nacosFileProp[1].split("&&");
        //这里获取到
        Properties properties = new Properties();
        properties.setProperty(PropertyKeyConst.SERVER_ADDR,
nacosStr[0] + ":" + nacosStr[1]);
        for (String propertyName : acceptProp) {
            String[] propertyItem = propertyName.split("=");
            String key = propertyItem[0];
            String value = propertyItem[1];
            if ("username".equals(key)) {
                properties.setProperty(PropertyKeyConst.USERNAME,
value);
            } else if ("password".equals(key)) {
                properties.setProperty(PropertyKeyConst.PASSWORD,
value);
            } else if ("namespace".equals(key)) {
                properties.setProperty(PropertyKeyConst.NAMESPACE,
value);
            }
        }
        ConfigService configService = null;
        try {
            configService =
NacosFactory.createConfigService(properties);
            String content = configService.getConfig(dataId,
GROUP, 6000);
```

```
            logger.info(content);
            return content.getBytes();
        } catch (NacosException e) {
            throw new RuntimeException(e);
        }
    }
}
```

这里需要用到关于 shardingjdbc 和 nacos 的依赖，相关依赖配置如下：

```SQL
<dependency>
    <groupId>org.apache.shardingsphere</groupId>
    <artifactId>shardingsphere-jdbc-core</artifactId>
    <version>5.3.2</version>
</dependency>
<dependency>
    <groupId>com.alibaba.nacos</groupId>
    <artifactId>nacos-client</artifactId>
</dependency>
```

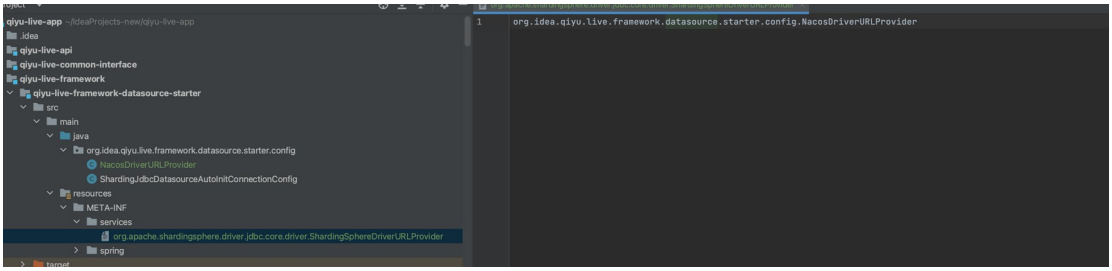
遵循 shardingjdbc 的 spi 机制，创建

/META-INF/services/
org.apache.shardingsphere.driver.jdbc.core.driver.ShardingSphereDriverURLProvider

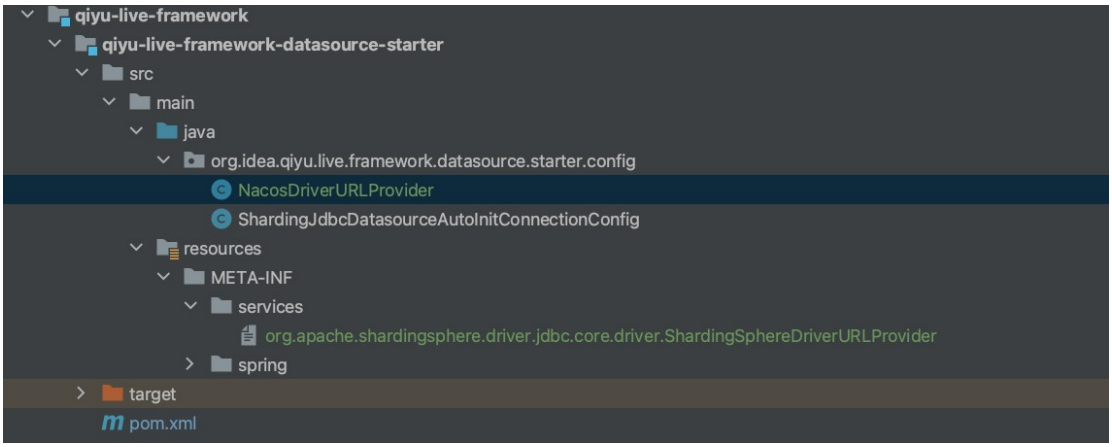
文件，然后在里面写上我们上边的这个 NacosDriverURLProvider 类的全路径地址。

```SQL
org.idea.qiyu.live.framework.datasource.starter.config.NacosDriver
URLProvider
```

spi 的配置如下图所示：



编写完上述内容后，整个 qiyu-live-framework-datasource-starter 模块基本如下图所示：



最后，修改我们在 nacos 的 qiyu-live-user-provider.yaml 中配置的 shardingjdbc 配置 url 参数为：

```SQL
jdbc:shardingsphere:nacos:qiyu.nacos.com:8848:qiyu-live-user-shardingjdbc.yaml?username=qiyu&&password=qiyu&&namespace=qiyu-live-test
```

如下图所示：