#### 1、什么是SPI机制

SPI（Service Provider Interface）机制，是一种将服务器接口与服务实现分离以达到解耦、大大提升了程序可扩展性的机制。引入服务提供者就是引入了spi接口的实现者，通过本地的注册发现获取到具体的实现类，轻松可插拔，SPI实际上是“基于接口的编程+策略模式+配置文件”组合实现的动态加载，为某个接口寻找服务实现的机制。

#### 2、Spring SPI机制

很多开源框架库里都直接或间接使用了Java 的SPI机制。比如Spring就有类似的SPI机制，通过SpringFactoriesLoader代替JDK中ServiceLoader，通过META-INF/spring.factories文件代替META/services目录下的描述文件，具体实现步骤不同，但原理都是使用Java的反射机制。

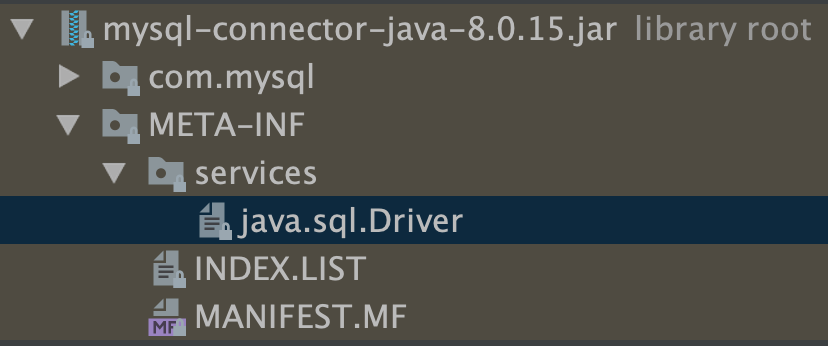
#### 3、SPI具体约定

Java SPI的具体约定为：当服务的提供者，提供了服务的一种实现之后，在jar包的META-INF/services/目录里同时创建一个以服务接口命名的文件。该文件里就是实现该服务接口的具体实现类。而当外部程序装配这个模块的时候，就能通过该jar包META-INF/services/里的配置文件找到具体的实现类名，并装载实例化，完成模块的注入。基于这样一个约定就能很好的找到服务接口的实现类，而不需要在代码里制定。JDK提供服务实现查找的一个工具类：java.util.ServiceLoader

#### 4、应用场景

（1）可替换的插件机制

典型应用，jdbc的实现机制。通常各大厂商（如Mysql、Oracle）会根据一个统一的规范（java.sql.Driver）开发各自的驱动实现逻辑。客户端使用jdbc时不需要去改变代码，直接引入不同的spi接口服务即可。Mysql则是com.mysql.jdbc.Driver，Oracle则是oracle.jdbc.driver.OracleDriver。



（2）框架的自定义扩展，比如spring的一些接口，阿里的dubbo也是通过spi来实现自定义扩展

#### 5、简单入门示例

（1）定义服务接口

（2）定义具体实现类

（3）src/main/resources/下建立META-INF/services目录，新增一个以接口命名的文件

（4）接口命名的文件填如如下内容



注：文件的内容可以多行，每行都是该接口对应的具体实现类的全限定名。

（5）使用ServiceLoader来加载配置文件中指定的实现

public OperationService getService(String key) {

if (StringUtils.isBlank(key)) {

throw new RuntimeException("key is null");

}

key = key.toLowerCase();

OperationService operationService = serviceMap.get(key);

if (operationService == null) {

ServiceLoader<OperationService> services = ServiceLoader.load(OperationService.class);

Iterator<OperationService> iterator = services.iterator();

while (iterator.hasNext()) {

operationService = iterator.next();

String clzName = operationService.getClass().getSimpleName();

String type = StringUtils.substring(clzName, 0, StringUtils.lastIndexOf(clzName, OperationService.class.getSimpleName()));

if (StringUtils.isNotBlank(type)) {

type = type.toLowerCase();

}

if (type.equals(key)) {

serviceMap.put(key, operationService);

break;

}

}

operationService = serviceMap.get(key);

}

if(operationService == null){

throw new RuntimeException("operationService is not found with key :"+key);

}

return operationService;

}