Java ServiceLoader

ServiceLoader与ClassLoader是Java中2个相互区别又相互联系的加载器：JVM利用ClassLoader将类载入内存，这是一个类生命周期的第一步（Java生命周期会经历加载、连接、初始化、使用和卸载五个阶段）；ServiceLoader，是一个简单的服务提供者加载设施，服务是一个熟知的接口和类集合。服务的提供者是服务的特定实现，提供者中的类通常实现接口，并子类化在服务本身中定义子类。服务提供者可以以扩展的形式安装在Java平台中，也就是将jar文件放入任意常用的扩展目录中，也可以通过将提供者加入应用程序类路径，或者通过其他特定于平台的方式使其可用。

ServiceLoader通过在资源目录META-INF/service中放置提供者配置文件来标识服务提供者，应用程序根据需要以延迟方式查找和实例化提供者。步骤如下：

1. 编写Service，接口

*public interface Animal {*

*void eat();*

*}*

1. 编写实现类（不一定要与接口在同一个工程中，可以存在于其他的jar中），继承Animal接口

*public class Dog implements Animal {*

*@Override*

*public void eat(){*

*System.out.println("Dog eat meat");*

*}*

*}*

*public class Cat implements Animal {*

*@Override*

*public void eat(){*

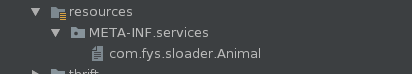
*System.out.println("Cat eat fish");*

*}*

*}*

1. 在实现类的所在project中的META-INF/service目录中配置接口与实现类的映射关系

在项目的src/main/resources中添加META-INF/services目录，创建接口对应的文件，如下图所示：



在文件中添加该接口的实现类，即service，内容如下：

*com.fys.sloader.Cat*

*com.fys.sloader.Dog*

1. 编写测试类

*public class TestServiceLoader {*

*public static void main(String[] args) {*

*ServiceLoader<Animal> serviceLoader = ServiceLoader.load(Animal.class);*

*Iterator<Animal> iter = serviceLoader.iterator();*

*while(iter.hasNext()) {*

*Animal animal = iter.next();*

*animal.eat();*

*}*

*}*

*}*

执行如下：

