通过 ServiceLoader 实现链式处理

作者：jsy
原文链接：https://ld246.com/article/1498037518256
来源网站：链滴

ServiceLoader：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（通常为抽象类）集合。
- 也可通过将提供者加入应用程序类路径，或者通过其他某些特定于平台的方式使其可用。唯一强制求的是，提供者类必须具有不带参数的构造方法，以便它们可以在加载中被实例化。

通过在资源目录META-INF/services中放置提供者配置文件来标识服务提供者。

该文件包含一个具体提供者类的完全限定二进制名称列表，每行一个。忽略各名称周围的空格、制表和空行。文件必须使用 UTF-8 编码。

以延迟方式查找和实例化提供者，也就是说根据需要进行。服务加载器维护到已经加载的提供者缓存。

每次调用 iterator 方法返回一个迭代器，它首先按照实例化顺序生成缓存的所有元素，然后以延迟方查找和实例化所有剩余的提供者，依次将每个提供者添加到缓存。可以通过 reload 方法清除缓存。

实例参考

1）基础服务：IService

```
public interface IService {

  String sayHello();
  String getScheme();

}
```

2）具体服务实现1：HDFSService

```
public class HDFSService implements IService {

  @Override

  public String sayHello() {

    return "Hello HDFSService";

  }

  @Override

  public String getScheme() {

    return "hdfs";

  }

}
```

3）具体服务实现2：LocalService

```
public class LocalService implements IService {

  @Override
```

```
    public String sayHello() {
      return  "Hello LocalService";
    }
    @Override
    public String getScheme() {
      return  "local";
    }
  }
```

4）配置：META-INF/services/com.service.IService

```
  com.impl.HDFSService
  com.impl.LocalService
```

5)测试类

```
  public  class  Test {
    public  static  void main(String[] args) {
      ServiceLoader serviceLoader = ServiceLoader.load(IService.class);
      for (IService service : serviceLoader) {
        System.out.println(service.getScheme()+"="+service.sayHello());
      }
    }
  }
```

结果：

hdfs=Hello HDFSService

local=Hello LocalService

可以看到ServiceLoader可以根据IService把定义的两个实现类找出来，返回一个ServiceLoader的实，而ServiceLoader实现了Iterable接口，所以可以通过ServiceLoader来遍历所有在配置文件中定义类的实例。