

# 高级开发必须理解的Java中SPI机制



分布式系统架构 [关注](#)

6 2018.09.14 10:19:26 字数 1,783 阅读 47,243

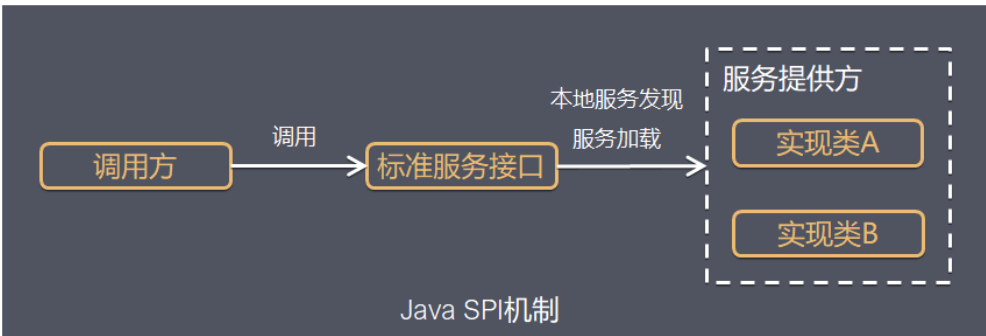


本文通过探析JDK提供的，在开源项目中比较常用的Java SPI机制，希望大家在实际开发实践、学习开源项目提供参考。

## 1 SPI是什么

SPI全称Service Provider Interface，是Java提供的一套用来被第三方实现或者扩展的API，它可以用来启用框架扩展和替换组件。

整体机制图如下：



Java 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机制。

简书

[首页](#)

[下载APP](#)

[搜索](#)



Aa

beta

[登录](#)

[注册](#)

间基于接口编程，模块之间不对实现类进行硬编码。一旦代码里涉及具体的实现类，就违反了可拔插的原则，如果需要替换一种实现，就需要修改代码。为了实现在模块装配的时候能不在程序里动态指明，这就需要一种服务发现机制。

### 推荐阅读

闭关修炼21天，“啃完”283页pdf，

Java SPI就是提供这样的一个机制：为某个接口寻找服务实现的机制。有点类似IOC的思想，就是将装配的控制权移到程序之外，在模块化设计中这个机制尤其重要。所以SPI的核心思想就是解耦。

## 2 使用场景

概括地说，适用者根据实际使用需要，启用、扩展、或者替换框架的实现策略

比较常见的例子：

- 数据库驱动加载接口实现类的加载  
JDBC加载不同类型数据库的驱动
- 日志门面接口实现类加载  
SLF4J加载不同提供商的日志实现类
- Spring  
Spring中大量使用了SPI,比如：对servlet3.0规范对ServletContainerInitializer的实现、自动类型转换Type Conversion SPI(Converter SPI、Formatter SPI)等
- Dubbo  
Dubbo中也大量使用SPI的方式实现框架的扩展, 不过它对Java提供的原生SPI做了封装，允许用户扩展实现Filter接口

## 3 使用介绍

要使用Java SPI，需要遵循如下约定：

- 1、当服务提供者提供了接口的一种具体实现后，在jar包的META-INF/services目录下创建一个以“接口全限定名”为命名的文件，内容为实现类的全限定名；
- 2、接口实现类所在的jar包放在主程序的classpath中；
- 3、主程序通过java.util.ServiceLoader动态装载实现模块，它通过扫描META-INF/services目录下的配置文件找到实现类的全限定名，把类加载到JVM；
- 4、SPI的实现类必须携带一个不带参数的构造方法；

## 示例代码

步骤1、定义一组接口 (假设是org.foo.demo.IShout)，并写出接口的一个或多个实现，(假设是org.foo.demo.animal.Dog、org.foo.demo.animal.Cat)。

```
1 public interface IShout {  
2     void shout();  
3 }  
4 public class Cat implements IShout {  
5     @Override  
6     public void shout() {  
7         System.out.println("mia mia");  
8     }  
9 }
```

写下你的评论...

评论 11 赞 76 ...

```
10 public class Dog implements IShout {  
11     @Override  
12     public void shout() {  
13         System.out.println("wang wang");  
14     }  
15 }
```

我终于4面拿下字节跳动offer

阅读 82,041

三面字节跳动被虐得“体无完肤”，  
15天读完这份pdf，终拿下美团研发

阅读 18,137

离开菜鸟&新的面试体验

阅读 2,667

败给“MySQL”的第33天，我重振旗鼓，四面拿下阿里淘系offer

阅读 1,789

对于二本渣渣来说，面试阿里P6也太难了！（两年crud经验，已拿

阅读 7,831



步骤2、在 src/main/resources/ 下建立 /META-INF/services 目录， 新增一个以接口命名的文件 (org.foo.demo.IShout文件)， 内容是要应用的实现类（这里是org.foo.demo.animal.Dog和org.foo.demo.animal.Cat， 每行一个类）。

文件位置

```
1 | - src
2 |   -main
3 |     -resources
4 |       - META-INF
5 |         - services
6 |           - org.foo.demo.IShout
```

文件内容

```
1 | org.foo.demo.animal.Dog
2 | org.foo.demo.animal.Cat
```

步骤3、使用 ServiceLoader 来加载配置文件中指定的实现。

```
1 | public class SPIMain {
2 |     public static void main(String[] args) {
3 |         ServiceLoader<IShout> shouts = ServiceLoader.load(IShout.class);
4 |         for (IShout s : shouts) {
5 |             s.shout();
6 |         }
7 |     }
8 | }
```

代码输出：

```
1 | wang wang
2 | miao miao
```

## 4 原理解析

首先看ServiceLoader类的签名类的成员变量：

```
1 | public final class ServiceLoader<S> implements Iterable<S>{
2 |     private static final String PREFIX = "META-INF/services/";
3 |
4 |     // 代表被加载的类或者接口
5 |     private final Class<S> service;
6 |
7 |     // 用于定位，加载和实例化providers的类加载器
8 |     private final ClassLoader loader;
9 |
10 |    // 创建ServiceLoader时采用的访问控制上下文
11 |    private final AccessControlContext acc;
12 |
13 |    // 缓存providers，按实例化的顺序排列
14 |    private LinkedHashMap<String,S> providers = new LinkedHashMap<>();
15 |
16 |    // 懒查找迭代器
17 |    private LazyIterator lookupIterator;
18 |
19 |    .....
20 | }
```

参考具体ServiceLoader具体源码，代码量不多，加上注释一共587行，梳理了一下，实现的流程如下：

- 1 应用程序调用ServiceLoader.load方法  
ServiceLoader.load方法内先创建一个新的ServiceLoader，并实例化该类中的成员变量，包括：
  - loader(ClassLoader类型，类加载器)
  - acc(AccessControlContext类型，访问控制器)
  - providers(LinkedHashMap<String,S>类型，用于缓存加载成功的类)
  - lookupIterator(实现迭代器功能)
- 2 应用程序通过迭代器接口获取对象实例  
ServiceLoader先判断成员变量providers对象中(LinkedHashMap<String,S>类型)是否有缓存实例对象，如果有缓存，直接返回。  
如果没有缓存，执行类的装载，实现如下：
  - (1) 读取META-INF/services/下的配置文件，获得所有能被实例化的类的名称，值得注意的是，ServiceLoader可以跨越jar包获取META-INF下的配置文件，具体加载配置的实现代码如下：

```
1      try {  
2          String fullName = PREFIX + service.getName();  
3          if (loader == null)  
4              configs = ClassLoader.getResources(fullName);  
5          else  
6              configs = loader.getResources(fullName);  
7      } catch (IOException x) {  
8          fail(service, "Error locating configuration files", x);  
9      }
```

- (2) 通过反射方法Class.forName()加载类对象，并用instance()方法将类实例化。
- (3) 把实例化后的类缓存到providers对象中，(LinkedHashMap<String,S>类型)  
然后返回实例对象。

## 5 总结

优点：

使用Java SPI机制的优势是实现解耦，使得第三方服务模块的装配控制的逻辑与调用者的业务代码分离，而不是耦合在一起。应用程序可以根据实际业务情况启用框架扩展或替换框架组件。

相比使用提供接口jar包，供第三方服务模块实现接口的方式，SPI的方式使得源框架，不必关心接口的实现类的路径，可以不用通过下面的方式获取接口实现类：

- 代码硬编码import 导入实现类
- 指定类全路径反射获取：例如在JDBC4.0之前，JDBC中获取数据库驱动类需要通过Class.forName("com.mysql.jdbc.Driver")，类似语句先动态加载数据库相关的驱动，然后再进行获取连接等的操作
- 第三方服务模块把接口实现类实例注册到指定地方，源框架从该处访问实例

通过SPI的方式，第三方服务模块实现接口后，在第三方的项目代码的META-INF/services目录下的配置文件指定实现类的全路径名，源码框架即可找到实现类

缺点：

- 虽然ServiceLoader也算是使用的延迟加载，但是基本只能通过遍历全部获取，也就是接口的实现类全部加载并实例化一遍。如果你并不想用某些实现类，它也被加载并实例化了，这就造成了浪费。获取某个实现类的方式不够灵活，只能通过Iterator形式获取，不能根据某个参数来获取对应的实现类。
- 多个并发多线程使用ServiceLoader类的实例是不安全的。

## 参考

[Java核心技术36讲](#)

[The Java™ Tutorials](#)

[Java Doc](#)

[Service Provider Interface: Creating Extensible Java Applications](#)

[Service provider interface](#)

[Java ServiceLoader使用和解析](#)

[Java基础之SPI机制](#)

[Java中SPI机制深入及源码解析](#)

[SPI机制简介](#)

更多精彩，欢迎关注公众号 分布式系统架构



"Any amount is appreciated."

赞赏支持

还没有人赞赏，支持一下



分布式系统架构 公众号【分布式系统架构】专注分享分布式系统架构干货  
总资产83 (约7.44元) 共写了11.9W字 获得552个赞 共439个粉丝

关注

## Parallels® Desktop 1

Mac与Window融合。针对macC  
优化。同时运行Windows和Mac



写下你的评论...

全部评论 11

只看作者

按时间倒序 按时间正序



dos91

9楼 2019.12.20 09:31

一个实际参考例子是HibernateValidator

👍 1    💬 回复



分布式系统架构 作者

2019.12.21 07:26

赞

💬 回复

添加新评论



guli\_2018

8楼 2019.12.19 16:00

使用Java SPI机制的优势是实现解耦，更重要的是提高扩展性吧

👍 赞    💬 回复



分布式系统架构 作者

2019.12.19 23:07

对头，可以理解是一种配置方式，方便实现类的替代

💬 回复

添加新评论



大聪明868

7楼 2019.11.12 23:26

👍👍👍

👍 赞    💬 回复



弱水\_穿云天

6楼 2019.10.12 19:50

优秀

👍 1    💬 回复



Lecon \

5楼 2019.09.10 15:19

谢谢大神分享🙏

赞 回复



小杰\_cdf4

4楼 2019.07.12 16:27

清晰明了

赞 回复



不给起这个名字

3楼 2019.05.14 17:34

用spi来调用跟直接依赖jar包然后调用实现类的区别是什么？

赞 回复

分布式系统架构 作者

2019.05.14 21:07

依赖jar包然后调用实现类，调用方需要在代码中通过import硬编码jar中实现类的全路径

用spi调用可以不用硬编码实现类的全路径

回复

添加新评论



ZX

2楼 2019.01.07 18:11

赞

赞 回复

## 被以下专题收入，发现更多相似内容



Java技术升华



程序员



大数据精进之路



Android...



dubbo



Java基础



Java全栈 java架构师

展开更多

## 推荐阅读

[更多精彩内容 >](#)

### 理解的Java中SPI机制

本文通过探析JDK提供的，在开源项目中比较常用的Java SPI机制，希望给大家在实际开发实践、学习开源项目提供参...

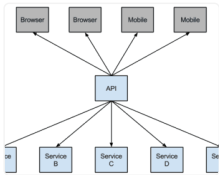


简祥 阅读 379 评论 0 赞 0

### Spring Cloud


Spring Cloud为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智...

 卡卡罗2017 阅读 82,763 评论 12 赞 122



### Java面试宝典Beta5.0

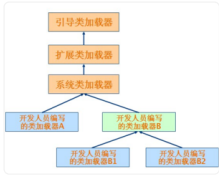
pdf下载地址：Java面试宝典 第一章内容介绍 20 第二章JavaSE基础 21 一、Java面向对象 21 ...

 王震阳 阅读 87,310 评论 26 赞 533

### 深入探讨 Java 类加载器


作者:成 富, 软件工程师, IBM 中国软件开发中心 类加载器（class loader）是Java™中的一个...

 Android技术研究 阅读 3,020 评论 0 赞 73



### 《小岛经济学》笔记

@定义经济：努力使有限的资源，产生最大的效益，以尽可能满足人类的需求。工具，资本及创新是实现这一目标的关键。经济增...

 奔跑d傻子 阅读 69 评论 0 赞 0