链滴

[solo 开发][插件] 插件加载不上

作者：lbb4511

原文链接：https://ld246.com/article/1553069512919

来源网站：链滴

```
[WARN ]-[2019-03-20 02:22:45]-[org.b3log.latke.plugin.PluginManager:148]: Load plugin und
r directory[/plugins/kanbanniang/] failed
java.lang.NullPointerException
    at java.base/java.util.ArrayDeque.addLast(ArrayDeque.java:304)
    at java.base/java.util.ArrayDeque.add(ArrayDeque.java:495)
    at java.base/jdk.internal.loader.URLClassPath.<init>(URLClassPath.java:154)
    at java.base/jdk.internal.loader.URLClassPath.<init>(URLClassPath.java:171)
    at java.base/java.net.URLClassLoader.<init>(URLClassLoader.java:120)
    at org.b3log.latke.plugin.PluginManager.load(PluginManager.java:185)
    at org.b3log.latke.plugin.PluginManager.load(PluginManager.java:142)
    at org.b3log.solo.SoloServletListener.contextInitialized(SoloServletListener.java:124)
    at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:4668)
    at org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:5136)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)
    at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1377)
    at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1367)
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
    at org.apache.tomcat.util.threads.InlineExecutorService.execute(InlineExecutorService.java:7
)
    at java.base/java.util.concurrent.AbstractExecutorService.submit(AbstractExecutorService.ja
a:140)
    at org.apache.catalina.core.ContainerBase.startInternal(ContainerBase.java:902)
    at org.apache.catalina.core.StandardHost.startInternal(StandardHost.java:831)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)
    at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1377)
    at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1367)
    at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)
    at org.apache.tomcat.util.threads.InlineExecutorService.execute(InlineExecutorService.java:7
)
    at java.base/java.util.concurrent.AbstractExecutorService.submit(AbstractExecutorService.ja
a:140)
    at org.apache.catalina.core.ContainerBase.startInternal(ContainerBase.java:902)
    at org.apache.catalina.core.StandardEngine.startInternal(StandardEngine.java:262)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)
    at org.apache.catalina.core.StandardService.startInternal(StandardService.java:423)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)
    at org.apache.catalina.core.StandardServer.startInternal(StandardServer.java:928)
    at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:183)
    at org.apache.catalina.startup.Catalina.start(Catalina.java:638)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorI
pl.java:62)
    at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodA
cessorImpl.java:43)
    at java.base/java.lang.reflect.Method.invoke(Method.java:566)
    at org.apache.catalina.startup.Bootstrap.start(Bootstrap.java:350)
    at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:492)
```

@88250

org.b3log.latke.plugin.PluginManager

```
    /**
     * Loads a plugin by the specified plugin directory and put it into the specified holder.
```

```
     *
     * @param pluginDirPath the specified plugin directory
     * @param holder        the specified holder
     * @return loaded plugin
     * @throws Exception exception
     */
    private AbstractPlugin load(final String pluginDirPath, final Map<String, HashSet<AbstractP
ugin>> holder) throws Exception {
        final Properties props = new Properties();
        final ServletContext servletContext = AbstractServletListener.getServletContext();

        String plugin = StringUtils.substringAfter(pluginDirPath, "/plugins");
        plugin = plugin.replace("/", "");

        final File file = Latkes.getWebFile("/plugins/" + plugin + "/plugin.properties");
        props.load(new FileInputStream(file));

        final URL defaultClassesFileDirURL = servletContext.getResource("/plugins/" + plugin + "
lasses");// 没有初始化
        URL classesFileDirURL = null; // 没有初始化
        try {
            classesFileDirURL = servletContext.getResource(props.getProperty("classesDirPath"));
        } catch (final MalformedURLException e) {
            LOGGER.log(Level.ERROR, "Reads [" + props.getProperty("classesDirPath") + "] failed",
e);
        }

        final URLClassLoader classLoader = new URLClassLoader(new URL[]{
                defaultClassesFileDirURL, classesFileDirURL}, PluginManager.class.getClassLoader());

        classLoaders.add(classLoader);

        String pluginClassName = props.getProperty(Plugin.PLUGIN_CLASS);
        if (StringUtils.isBlank(pluginClassName)) {
            pluginClassName = NotInteractivePlugin.class.getName();
        }

        final String rendererId = props.getProperty(Plugin.PLUGIN_RENDERER_ID);

        if (StringUtils.isBlank(rendererId)) {
            LOGGER.log(Level.WARN, "no renderer defined by this plugin[" + plugin + "],  this plu
in will be ignore!");
            return null;
        }

        final Class<?> pluginClass = classLoader.loadClass(pluginClassName);

        LOGGER.log(Level.TRACE, "Loading plugin class[name={0}]", pluginClassName);
        final AbstractPlugin ret = (AbstractPlugin) pluginClass.newInstance();

        ret.setRendererId(rendererId);

        setPluginProps(plugin, ret, props);
        registerEventListeners(props, classLoader, ret);
```

```
        register(ret, holder);

        ret.changeStatus();

        return ret;
    }
```

new URLClassLoader()的时候有用到ArrayDeque

```
        ArrayList<URL> path = new ArrayList<>(urls.length);
        ArrayDeque<URL> unopenedUrls = new ArrayDeque<>(urls.length);
        for (URL url : urls) {
            path.add(url);
            unopenedUrls.add(url);
        }
```

ArrayDeque 是 Deque 接口的一种具体实现，是依赖于可变数组来实现的。ArrayDeque 没有容量制，可根据需求自动进行扩容。ArrayDeque不支持值为 null 的元素。

Queue 也是 Java 集合框架中定义的一种接口，直接继承自 Collection 接口。除了基本的 Collection 接口规定测操作外，Queue 接口还定义一组针对队列的特殊操作。通常来说，Queue 是按照先进先出FIFO)的方式来管理其中的元素的，但是优先队列是一个例外。

Deque 接口继承自 Queue接口，但 Deque 支持同时从两端添加或移除元素，因此又被成为双端队。鉴于此，Deque 接口的实现可以被当作 FIFO队列使用，也可以当作LIFO队列（栈）来使用。官方是推荐使用 Deque 的实现来替代 Stack。

```
    final URL defaultClassesFileDirURL = servletContext.getResource("/plugins/" + plugin + classes") != null
        ? servletContext.getResource("/plugins/" + plugin + "classes")
        : new URL("file:");
    URL classesFileDirURL = new URL("file:");
```