

동적 Jar loading 프로그램

동적 jar 로딩 및 동적 method 호출

```
1 package com.exec;
2
3 import java.io.File;
4 import java.io.FileFilter;
5 import java.io.FilenameFilter;
6 import java.lang.reflect.InvocationTargetException;
7 import java.lang.reflect.Method;
8 import java.net.URL;
9 import java.net.URLClassLoader;
10 import java.util.Arrays;
11
12 public class ExecTest {
13
14     public static void main(String[] args) throws NoSuchMethodException, S
15
16     // 코리안리 EAI customer function 위치 (고정 값)
17         String dir = "C:\\eclipse-workspace\\com.exec\\src\\main\\reso
18     // 코리안리 EAI customer function prefix (고정 값)
19         String jarName = "kr.co.koreanre.eai";
20     // 코리안리 EAI customer function Class 명 분류 할 수도 있다.
21         String className = "kr.co.koreanre.eai.execEaiFunctionClass";
22     // 코리안리 EAI customer function method 명은 고정 예정
23         String methodName = "execMathod";
24     // 코리안리 EAI customer function method에 전달되는 문자열 전달
25         String param = "Call method ,";
26
27         ExecTest ex = new ExecTest();
28
29
30         // kr.co.koreanre.eai-202110280920.jar의 include 여부를 확인, inclu
31         if(!ex.checkLoadedJarList(dir, jarName))
32         {
33             System.out.println(" > Do not include : " + jarName );
34         // 최신 버전으로 include
35             ex.loadJar(dir);
36         }
37
38         String returnString = ex.execMethod(className, methodName, par
39         System.out.println(" > Return Value : " + returnString);
40         ex.checkLoadedJarList(dir, jarName);
41     }
42 }
```

```

43
44
45
46      /* =====
47
48
49
50
51
52      /**
53       * kr.co.koreanre.eai-202110280920.jar의 로드 여부를 확인
54       *
55       * @param jarName
56       * @return boolean
57       */
58      private boolean checkLoadedJarList(String dir, String jarName) {
59          System.out.println("\n>> Start Check loaded Jar list " );
60          boolean includJar = false;
61
62          ClassLoader classLoader = ClassLoader.getSystemClassLoader();
63          URL[] urls = ((URLClassLoader)classLoader).getURLs();
64
65          for(URL url: urls){
66              if( url.getFile().toLowerCase().contains(".jar") ) {
67                  System.out.println(" > Included Jar : " + url.getFile());
68                  if(url.getFile().toLowerCase().contains(jarName))
69                      {
70                          if(isLoadNewVersionJar(dir, url.getFile()))
71                          {
72                              includJar = true;
73                          }else {
74                              includJar = false;
75                          }
76                      }else {
77                          includJar = false;
78                      }
79              }
80          }
81          return includJar;
82      }
83
84      /**
85       * 디렉토리 안에서 파라미터로 전달한 jar 파일이 최신인지 여부를 판단
86       * @param dir 검토 폴더
87       * @param jarName 비교 jar
88       * @return boolean
89       */
90      private boolean isLoadNewVersionJar(String dir, String jarName) {
91          boolean isNweVersion = false;
92

```

```

93         File file = new File(dir);
94         File[] fileNameList = file.listFiles(new FilenameFilter() {
95             public boolean accept(File dir, String name) {
96                 return name.endsWith(".jar");
97             }
98         });
99
100         if(fileNameList.length==0) {
101             throw new NoClassDefFoundError("Koreanre EAI jar not e
102         }else {
103             Arrays.sort(fileNameList);
104             String oldVersion = jarName.replace("\\", "/").substri
105             String newVersion = fileNameList[fileNameList.length-1
106             System.out.println(" > OLD : " + oldVersion );
107             System.out.println(" > NEW : " + newVersion.substring(
108             if(oldVersion.equals(newVersion.substring(newVersion.l
109             {
110                 System.out.println(" > is New Version");
111                 isNweVersion = true;
112             }else {
113                 System.out.println(" > is Not New Version");
114                 isNweVersion = false;
115             }
116         }
117
118         return isNweVersion;
119     }
120
121     /**
122     * Method를 실행
123     * @param className
124     * @param methodName
125     * @param param
126     * @return
127     * @throws ClassNotFoundException
128     * @throws NoSuchMethodException
129     * @throws SecurityException
130     * @throws InstantiationException
131     * @throws IllegalAccessException
132     * @throws IllegalArgumentException
133     * @throws InvocationTargetException
134     */
135     private String execMethod(String className, String methodName, String ;
136         System.out.println("\n>> Start exec method ! " );
137
138         ClassLoader cl = Thread.currentThread().getContextClassLoader(
139         Class<?> clazz = cl.loadClass(className);
140
141         Method m = clazz.getMethod(methodName, param.getClass() );
142         m.setAccessible(true);

```

```

143
144     System.out.println(" > Classd Name : " + className);
145     System.out.println(" > Method Name : " + methodName);
146     System.out.println(" > Params Value : " + param);
147     System.out.println(" > Return Type : " + m.getReturnType());
148
149     if (m.getReturnType() != String.class) {
150         System.out.println(" > Return type Stirng only : " + m.getRetu
151         throw new NoSuchMethodException(methodName);
152     }
153
154     if (m.getModifiers() == 9) {
155         System.out.println(" > Not allowed Static method : " + m.getMo
156         throw new NoSuchMethodException(methodName);
157     }
158
159     if (m.getModifiers() != 1) {
160         System.out.println(" > Public method only : " + m.getModifiers
161         throw new NoSuchMethodException(methodName);
162     }
163
164     Object bbb = clazz.newInstance();
165     String returnMsg = "";
166     returnMsg = (String)m.invoke(bbb , new Object[] { param });
167
168     return returnMsg;
169
170 }
171
172
173 /**
174  * 해당 디렉토리의 최신 버전의 jar를 적용한다.
175  * @param dir
176  * @throws NoSuchMethodException
177  * @throws SecurityException
178  */
179 private void loadJar(final String dir) throws NoSuchMethodException, S
180     System.out.println("\n>> LoadJar ");
181     final URLClassLoader loader = (URLClassLoader)Thread.currentTh
182     final Method method = URLClassLoader.class.getDeclaredMethod("
183     method.setAccessible(true);
184
185     new File(dir).listFiles(new FileFilter() {
186     public boolean accept(File jar) {
187         if( jar.toString().toLowerCase().contains(".jar") ){
188             try{
189                 if(isLoadNewVersionJar(dir, jar.toString())) {
190                     method.invoke(loader, new Object[]{jar.toURI()
191                     System.out.println(" > " + jar.toURI() + " is
192                 }

```

```

193
194         }catch(Exception e){
195             System.out.println(jar.toURI() + " can't load.");
196         }
197     }
198         return false;
199     }
200 });
201 }
202
203 }

```

• 호출을 받는 소스

줄 번호 보이기/숨기기

```

1 package kr.co.koreanre.eai;
2
3 public class execEaiFunctionClass {
4
5     public execEaiFunctionClass() {
6         System.out.println(" > Init #1" );
7     }
8
9     public String execMathod( String obj ) {
10
11         String reString = new String();
12
13         return reString + "= called test1";
14
15     }
16
17 }

```

• 호출을 받는 곳 pom.xml

```

1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.ap
4     <modelVersion>4.0.0</modelVersion>
5     <groupId>kr.co.koreanre.eai</groupId>
6     <artifactId>kr.co.koreanre.eai</artifactId>
7     <version>202110280920</version>
8     <name>Test1</name>
9
10     <build>
11         <pluginManagement>
12             <plugins>
13                 <plugin>
14                     <groupId>org.apache.maven.plugins</gro
15                     <artifactId>maven-resources-plugin</ar

```

16	<code><version>2.3</version></code>
17	<code></plugin></code>
18	<code><plugin></code>
19	<code><artifactId>maven-resources-plugin</ar</code>
20	<code><version>3.0.2</version></code>
21	<code><executions></code>
22	<code><execution></code>
23	<code><id>copy-resource-one<</code>
24	<code><phase>install</phase></code>
25	<code><goals></code>
26	<code><goal>copy-res</code>
27	<code></goals></code>
28	<code><configuration></code>
29	<code><outputDirecto</code>
30	<code><resources></code>
31	<code><resou</code>
32	
33	
34	
35	
36	<code></reso</code>
37	<code></resources></code>
38	<code></configuration></code>
39	<code></execution></code>
40	<code></executions></code>
41	<code></plugin></code>
42	<code></plugins></code>
43	<code></pluginManagement></code>
44	<code></build></code>
45	
46	<code></project></code>

🔄Revision #1

★Created 31 May 2023 01:23:38 by Hyeon Su Ryu

✎Updated 31 May 2023 14:37:38 by Hyeon Su Ryu