|  |  |
| --- | --- |
| |  | | --- | | **Java版的实现JavaScript中的eval()函数** | |
| |  | | --- | |  | | |  | | --- | | 实现步骤：  1.自定义一个Java类，该Java类中定义一个方法来包含需要被运行的代码。 2.动态编译刚刚生成的Java源码，不在磁盘上生成源码，而是直接编译内存中的Java源码。 3.动态加载刚刚创建编译的Java二进制码，编译好的Java二进制码不是在磁盘上，而是放在内存中，并定义自己的类加载器，负责加载内存中的class文件。 4.通过反射运行前一步加载的类。    **[java]** [view plaincopy](http://blog.csdn.net/leeyohn/article/details/5179422)   1. **import** java.util.Arrays; 2. **import** javax.tools.SimpleJavaFileObject; 3. **import** javax.tools.JavaFileObject; 4. **import** javax.tools.JavaCompiler; 5. **import** javax.tools.ToolProvider; 6. **import** javax.tools.DiagnosticCollector; 7. **import** java.net.URI;  10. /\*\* 11. \* Description: 12. \* <br/>网站: <a href="http://www.crazyit.org" mce\_href="http://www.crazyit.org">疯狂Java联盟</a> 13. \* <br/>Copyright (C), 2001-2010, Leeyohn 14. \* <br/>This program is protected by copyright laws. 15. \* <br/>Program Name: 16. \* <br/>Date: 17. \* @author Leeyohn leeyohn@hotmail.com 18. \* @version 1.0 19. \*/ 21. **public** **class** MyClassLoader 22. **extends** ClassLoader 23. { 24. @Override 25. **public** Class<?> findClass(String str) **throws** ClassNotFoundException 26. { 27. JavaCompiler compiler = ToolProvider.getSystemJavaCompiler(); 28. //用于诊断源代码编译错误的对象 29. DiagnosticCollector diagnostics = **new** DiagnosticCollector(); 30. //内存中的源代码保存在一个从JavaFileObject继承的类中 31. JavaFileObject file = **new** JavaSourceFromString("Temp", str.toString()); 32. Iterable compilationUnits = Arrays.asList(file); 33. //建立一个编译任务 34. JavaCompiler.CompilationTask task = compiler.getTask(**null**, **null**, **null**, **null**, **null**, compilationUnits); 35. //编译源程序 36. **boolean** result = task.call(); 37. **if** (result) 38. { 39. **return** Class.forName("Temp"); 40. } 41. **return** **null**; 42. } 43. } 45. **class** JavaSourceFromString **extends** SimpleJavaFileObject 46. { 47. **private** String name; 48. **private** String code; 49. **public** JavaSourceFromString(String name, String code) 50. { 51. **super**(URI.create("string:///" + name.replace('.', '/') + Kind.SOURCE.extension), Kind.SOURCE); 52. **this**.code = code; 53. } 55. **public** CharSequence getCharContent(**boolean** ignoreEncodingErrors) 56. { 57. **return** code; 58. } 59. }     **[c-sharp]** [view plaincopy](http://blog.csdn.net/leeyohn/article/details/5179422)   1. import java.lang.reflect.Method; 3. /\*\* 4. \* Description: 5. \* <br/>网站: <a href="http://www.crazyit.org" mce\_href="http://www.crazyit.org">疯狂Java联盟</a> 6. \* <br/>Copyright (C), 2001-2010, Leeyohn 7. \* <br/>This program is protected by copyright laws. 8. \* <br/>Program Name: 9. \* <br/>Date: 10. \* @author Leeyohn leeyohn@hotmail.com 11. \* @version 1.0 12. \*/ 14. **public** **class** Eval 15. { 16. **public** **static** Object eval(String str) throws Exception 17. { 18. StringBuffer sb = **new** StringBuffer(); 19. sb.append("public class Temp"); 20. sb.append("{"); 21. sb.append(" public Object getObject()"); 22. sb.append(" {"); 23. sb.append(" " + str + "return new Object();"); 24. sb.append(" }"); 25. sb.append("}"); 26. //调用自定义类加载器加载编译在内存中class文件 27. Class clazz = **new** MyClassLoader().findClass(sb.toString()); 28. Method method = clazz.getMethod("getObject"); 29. //通过反射调用方法 30. **return** method.invoke(clazz.newInstance()); 31. } 33. **public** **static** **void** main(String[] args) throws Exception 34. { 35. Object rval = eval("System.out.println(/"Hello World/");"); 36. System.**out**.println(rval); 37. } 38. } | | |