动态编译

讲师：高淇　邮箱：gaoqi110@163.com

动态编译

- JAVA 6.0引入了动态编译机制。

- 动态编译的应用场景：
  - 可以做一个浏览器端编写java代码，上传服务器编译和运行的在线评测系统。
  - 服务器动态加载某些类文件进行编译
- 动态编译的两种做法：
  - 通过Runtime调用javac，启动新的进程去操作

```
Runtime run = Runtime.getRuntime();
Process process = run.exec("javac -cp d:/myjava/   HelloWorld.java");
```

  - 通过JavaCompiler动态编译

动态编译

- 通过JavaCompiler动态编译

```
public static int compileFile(String sourceFile){
    //动态编译
    JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();
    int result = compiler.run(null, null, null,sourceFile);
    System.out.println(result==0?"编译成功":"编译失败");
    return result;
}
```

- 第一个参数：为java编译器提供参数
- 第二个参数：得到 Java 编译器的输出信息
- 第三个参数：接收编译器的 错误信息
- 第四个参数：可变参数（是一个String数组）能传入一个或多个 Java 源文件
- 返回值：0表示编译成功，非0表示编译失败

动态运行编译好的类

- 通过Runtime.*getRuntime()*运行启动新的进程运行

```
Runtime run = Runtime.getRuntime();
        Process process = run.exec("java -cp  d:/myjava    HelloWorld");
  //      Process process = run.exec("java -cp "+dir+" "+classFile);
```

- 通过反射运行编译好的类

```
//通过反射运行程序
public static void runJavaClassByReflect(String dir,String classFile) throws
Exception{
 try {
            URL[] urls = new URL[] {new URL("file:/"+dir)};
            URLClassLoader loader = new URLClassLoader(urls);
            Class c = loader.loadClass(classFile);
            //调用加载类的main方法
            c.getMethod("main",String[].class).invoke(null, (Object)new
String[]{});
        } catch (Exception e) {
            e.printStackTrace();
        }
}
```