Memory Class Compiler (MCC)

□ schmittjoaopedro . Java □ 1 01+00:00 Março 01+00:00 201818 18+00:00 Novembro 18+00:00 2019 □ 3 Minutes

The Memory Class Compiler (MCC) is a simple Java library used to compile Java classes at runtime.

(https://github.com/schmittjoaopedro/mcc#purpose)Purpose

The purpose of MCC is to provide a simple API to be used in applications that need to compile and execute Java source code on the fly, where the Java source code is represented as a String.

(https://github.com/schmittjoaopedro/mcc#license)License

This software is available under MIT license (https://opensource.org/licenses/MIT).

(https://github.com/schmittjoaopedro/mcc#technical-overview)Technical Overview

The software was written in Java (http://www.oracle.com/technetwork/java/javase/downloads/jdk9-downloads-3848520.html) and tested with the following versions of Java Development Toolkit (JDK): JDK 7, JDK 8 and JDK 9. There is only one requirement, the library needs the program to be running under a JDK because only JDK grants access to the compiler API.

The following features are available:

- A simple library that facilitates the access to JavaCompiler implementation of JDK.

- A simple library that facilitates the access to execute PMD validation in Java source code.
- A custom ClassLoader to load java source code compiled at runtime.

(https://github.com/schmittjoaopedro/mcc#getting-started)Getting Started

To start using MCC, first, embed the library into your Java application via the following snippet

```
<dependency>
    <groupId>com.github.schmittjoaopedro</groupId>
    <artifactId>mcc</artifactId>
    <version>1.0.1</version>
</dependency>
```

(https://github.com/schmittjoaopedro/mcc#create-a-simple-class)Create a simple class

This is a simple example of compilation using MCC:

```
//Create a simple String with the Java source code
String sourceCode =
    "package comp.test;" +
    "public class Test {" +
    "    public String sayHello() {" +
    "        return \"Hello World!\";" +
    "    }" +
    "}";

//Create a object to encapsulate the source code
SourceClass sourceClass = new SourceClass("comp.test", "Test", sourceCode);

//Compile the SourceClass object
MemoryClassCompiler compiler = new MemoryClassCompiler();
compiler.checkAndCompile(sourceClass);

//Create a class loader and define the compiled class
SourceClassLoader classLoader = new SourceClassLoader(getClass().getClassLoa
Class loadedClass = classLoader.loadSourceClassLoader(sourceClass);

//Invoke the method with reflection
Object o = loadedClass.newInstance();
Method m = loadedClass.getDeclaredMethod("sayHello", null);
Assert.assertEquals(m.invoke(o, null), "Hello World!");
```

(https://github.com/schmittjoaopedro/mcc#executing-batch-compilation)Executing batch compilation

This is an example compiling one or more Java classes with batch jobs:

```
//Create a source task to execute batch compilation
SourceTask sourceTask = new SourceTask();

//Creating some classes
for(int i = 0; i < 10; i++) {
    String sourceCode =
        "package comp.test;" +
        "public class Test" + i + " {" +
        "    public String sayHello(Integer val) { " +
        "        return val + \" - Hello World!\";" +
        "    }" +
        "}";
    //Add the class to the source task
    sourceTask.createSourceClass("comp.test", "Test" + i, sourceCode);
}

//Create a memory compiler and execute passing the source task
MemoryClassCompiler compiler = new MemoryClassCompiler();
compiler.checkAndCompile(sourceTask);

//Create a custom class loader
SourceClassLoader classLoader = new SourceClassLoader(getClass().getClassLoa
for(int i = 0; i < 10; i++) {
    //Invoker the classes using reflection
    SourceClass sourceClass = sourceTask.getSourcesClass().get(i);
    Class loadedClass = classLoader.loadSourceClassLoader(sourceClass);
    Object o = loadedClass.newInstance();
    Method m = loadedClass.getDeclaredMethod("sayHello", Integer.class);
    Assert.assertEquals(m.invoke(o, i), i + " - Hello World!");
}
```

(https://github.com/schmittjoaopedro/mcc#managing-errors)Managing errors

This is a simple example obtaining detailed information when errors are thrown by the compiler:

```
//Example of class with problem (returning int but declared as void)
SourceClass sourceClass = new SourceClass();
sourceClass.setPackageName("teste");
sourceClass.setClassName("Teste");
sourceClass.setSourceCode("package teste; public class Teste { public void t(

MemoryClassCompiler compiler = new MemoryClassCompiler();
try {
    compiler.compile(sourceClass);
} catch (MemoryCompilerException ex) {
    MessageCompiler message = ex.getMessageCompiler();
    Assert.assertNull(sourceClass.getBytecode());
    Assert.assertEquals(message.getMessage(), "Error in compilation of class'
    Assert.assertEquals(message.getStatus(), MessageStatus.FAILED);
    Assert.assertEquals(message.getDiagnostics().get(0).getCode(), "compiler.
    Assert.assertEquals(message.getDiagnostics().get(0).getColumnNumber(), 62
    Assert.assertEquals(message.getDiagnostics().get(0).getEndPosition(), 62)
    Assert.assertEquals(message.getDiagnostics().get(0).getLineNumber(), 1);
    Assert.assertEquals(message.getDiagnostics().get(0).getMessage(null), "in
    Assert.assertEquals(message.getDiagnostics().get(0).getPosition(), 61);
    Assert.assertEquals(message.getDiagnostics().get(0).getStartPosition(), 6
    Assert.assertEquals(message.getDiagnostics().get(0).getKind(), Diagnostic
}
```

(https://github.com/schmittjoaopedro/mcc#managing-pmd-validations)Managing PMD validations

This is a simple example obtaining detailed information when PMD errors are thrown by the compiler:

```
SourceClass sourceClass = new SourceClass();
sourceClass.setPackageName("test");
sourceClass.setClassName("Test");
sourceClass.setSourceCode("package test; public class Test { private int t; }
try {
    new MemoryPMDValidator().check(sourceClass);
} catch(MemoryCompilerException ex) {
    Assert.assertEquals(ex.getMessage(), "PMD_ERROR: PMD Validation failed\n
}
```

The source code is available at GitHub. (https://github.com/schmittjoaopedro/mcc)

Com as etiquetas :
class,
compiler,
development,
java,
Java SE,
JDK,
memory,
programming,
source code

Publicado por schmittjoaopedro

Graduado como bacharel em Sistemas de Informação pelo Centro Universitário Católica de Santa Catarina campus Jaraguá do Sul. Formado no Ensino Médio pelo Senai com Técnico em Redes de Computadores Articulado. Atualmente desenvolvedor JEE/Web em Sistemas de Engenharia na WEG. Pesquisador no período de faculdade em Informática pela Católica de Santa Catarina. Contato 47 - 99615 2305 E-mail: schmittjoaopedro@gmail.com Web page: https://joaoschmitt.wordpress.com/ Linkedin: https://www.linkedin.com/in/joao-pedro-schmitt-60847470/ Curriculum lattes: http://lattes.cnpq.br/9304236291664423 Twitter: @JooPedroSchmitt Ver todos os artigos de schmittjoaopedro