Atitit 约定大于配置 调用某段函数的方法总结

目录

[1. 调用代码的几种方法 1](#_Toc7353)

[1.1. 直接调用代码文件法 Include法 1](#_Toc26186)

[1.2. 函数名称调用法 2](#_Toc10362)

[2. Java的俩种模式 2](#_Toc26804)

[2.1. Include法之间执行 加载类 2](#_Toc10585)

[2.2. Import +New 关键字法 2](#_Toc25184)

[2.3. 动态读取法Class。Forname（） 2](#_Toc2454)

[3. 函数名称调用法 3](#_Toc4996)

[3.1. 先加载class，在new/static模式 调用，， 3](#_Toc21436)

[3.2. 动态调用需要使用反射机制。。。 3](#_Toc8957)

[3.3. 或者el表达式动态调用。。。 3](#_Toc18345)

# 调用代码的几种方法

## 直接调用代码文件法 Include法

加载代码文件调用代码法 Include法

Java需要import ,+new关键词法。。

动态调用需要使用反射

如何传递参数（环境变量法？？）

使用jvm模式传递环境参数也可以。。

二.系统属性 (特指Java中获取的系统属性)

* 系统属性的设置: 通过JVM参数: -D属性名=值 或者在代码中通过Sytem.setProperty(String key, String value)来设置.
* 系统属性的获取: 在Java中通过System.getProperty(String key)获取属性值.

## 函数名称调用法

先加载代码文件（include，import），然后使用函数名称调用法（

动态调用需要使用反射

# Java的俩种模式

## Include法之间执行 加载类

## Import +New 关键字法

运行文件代码 之间写成static模式加载应该会简单吧。。测试。。

**package** commxx.util.net;

**public** **class** Login {

{

System.***out***.println("logn...");

}

}

## 动态读取法Class。Forname（）

脚本的话Include 动态模式都是一样的。。，java不一样。。

**public** **static** **void** main(String[] args) **throws** Exception {

System.***out***.println(Class.*forName*("commxx.util.net.Login").newInstance());

# 函数名称调用法

## 先加载class，在new/static模式 调用，，

## 动态调用需要使用反射机制。。。

## 或者el表达式动态调用。。。