**Log4j日志框架使用**

在程序开发过程中，我们经常需要根据程序的重要级别记录不同的日志信息，这时我们就要用到日志框架，log4j是目前最流行的日志框架，它支持对不同包下面的日志使用不同的日志级别并创建不同的日志文件输出到指定位置，下面我们介绍一下log4j与spring mvc的整合过程

**1.pom中添加依赖**

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

**2.Web.xml文件中添加log4j的配置**

<!--log4jRefreshInterval为6000表示 开一条watchdog线程每6秒扫描一下配置文件的变化-->

<context-param>

<param-name>log4jRefreshInterval</param-name>

<param-value>6000</param-value>

</context-param>

<!--设置log4j的配置文件位置 -->

<context-param>

<param-name>log4jConfigLocation</param-name>

<param-value>WEB-INF/classes/log4j.properties</param-value>

</context-param>

<!--使用监听加载log4j的配置文件 -->

<listener>

<listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>

</listener>

**3.编辑log4j.properties配置文件**

#其中DEBUG表是的是**所有包下**的日志级别为INFO,appender1表示的是日志输出到哪里

log4j.rootLogger=INFO,appender1

#输出**指定包**com.springapp.mvc中的日志,其中DEBUG表示的是该包下的日志级别，其优先级别高于rootLogger规定的全局日志优先级别,appender2表示的是输出到哪里

log4j.logger.com.springapp.mvc =DEBUG,appender2

#配置输出到控制台,如果这里也配置了日志级别,那么实际日志级别就以该日志级别和**指定包(如果没有指定包那就是rootLogger规定的所有包)**下的日志级别中级别较高（输出的少的）的为准

log4j.appender.appender1=org.apache.log4j.ConsoleAppender

log4j.appender.appender1.Threshold=DEBUG

log4j.appender.appender1.layout=org.apache.log4j.PatternLayout

log4j.appender.appender1.layout.ConversionPattern =[%d{yy/MM/ddHH:mm:ss:SSS}][%C-%M]%m%n

#输出到指定文件,如果这里也配置了日志级别,那么实际日志级别就以该日志级别和**指定包(如果没有指定包那就是rootLogger规定的所有包)**下的日志级别中级别较高（输出的少的）的为准

log4j.appender.appender2=org.apache.log4j.FileAppender

log4j.appender.appender2.File=F:/logs/mylog.log

log4j.appender.appender2.Threshold=INFO

log4j.appender.appender2.layout=org.apache.log4j.PatternLayout

log4j.appender.appender2.layout.ConversionPattern=[%d{HH:mm:ss:SSS}][%C-%M]-%m%n

**4.编写代码**

//引入logger

**Logger logger=Logger.*getLogger*(HelloController.class);**

@RequestMapping(value = "hello.do", method = RequestMethod.***GET***)

**public** String printWelcome(ModelMap model) {

logger.info("==================================info");

logger.debug("=================================debug");

model.put("message", "hello world");

**return** "hello";

}

经过上面的配置，我们就可以通过appender1来控制控制台中的日志输出，通过appender2来控制com.springapp.mvc包下的日志输出到指定文件.

**5.精确控制**

但是还有一个问题，就是如果最终的日志级别是DEBUG，那么级别大于DEBUG的INFO,ERROR类别的日志也会被输出，有时候我们希望精确的输出某一个包下的某一种日志级别到某一个文件中。如本例中我们只希望输出DEBUG级别的日志，而不需要INFO,ERROR两种日志，现在我们对appender2做以下修改:

#log4j.appender.appender2=org.apache.log4j.FileAppender

log4j.appender.appender2=com.base.log.SingleLevelFileAppender

log4j.appender.appender2.File=F:/logs/mylog.log

log4j.appender.appender2.Threshold=DEBUG

log4j.appender.appender2.layout=org.apache.log4j.PatternLayout

log4j.appender.appender2.layout.ConversionPattern=[%d{HH:mm:ss:SSS}][%C-%M]-%m%n

注释掉:

log4j.appender.appender2=org.apache.log4j.FileAppender

添加红色部分:

log4j.appender.appender2=com.base.log.SingleLevelFileAppender

其中com.base.log.SingleLevelFileAppender是我们自定义的类，该类继承了FileAppender类，其定义如下：

**public** **class** SingleLevelFileAppender **extends** FileAppender {

@Override

**public** **boolean** isAsSevereAsThreshold(Priority priority) {

//父类FileAppender原始代码

//return threshold == null || priority.isGreaterOrEqual(threshold);

**//改成如下:**

**return** **this**.getThreshold().equals(priority);

}

}

通过上面的配置我们就完成了将指定包下的日志按照某一种日志级别精确的输出到某一个文件中去的需求。