IBM Cloud Liberty 配置log4j2

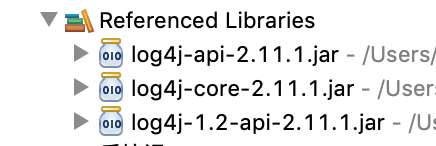
下载log4j2:

<https://logging.apache.org/log4j/2.x/download.html>

我下载的版本：

<https://www.apache.org/dyn/closer.lua/logging/log4j/2.11.1/apache-log4j-2.11.1-bin.zip>

需要先导入log4j2版本的jar包：



日志直接输出到控制台：

新建一个Class文件

**package** log4j2Test;

**import** org.apache.logging.log4j.LogManager;

**import** org.apache.logging.log4j.Logger;

**public** **class** Log4j2Test {

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

// Logger logger = LogManager.getLogger(LogManager.ROOT\_LOGGER\_NAME);

// Logger logger = LogManager.getLogger(Log4j2Test.class);

Logger logger = LogManager.*getLogger*("this is log4j");

logger.trace("trace level");

logger.debug("debug level");

logger.info("info level");

logger.warn("warn level");

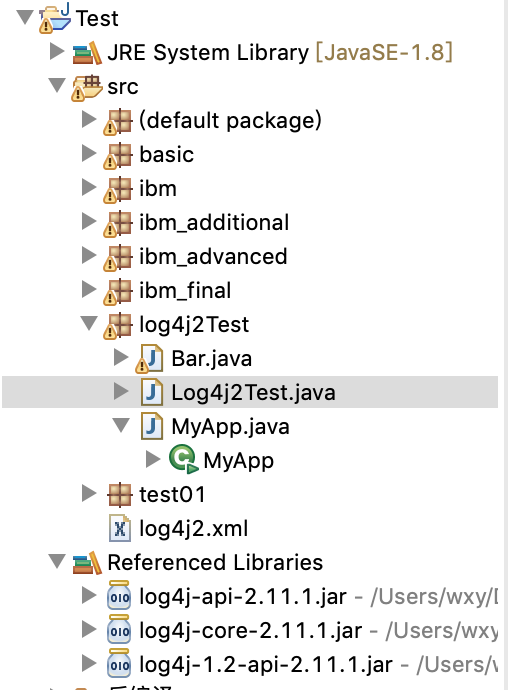
logger.error("erro level");

logger.fatal("fatal level");

}

}

配置log4j2.xml文件，位置在src目录下面：



Log4j2.xml文件配置信息内容如下：

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<Configuration status=*"WARN"*>

<Appenders>

<Console name=*"Console"* target=*"SYSTEM\_OUT"*>

<PatternLayout pattern=*"%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"*/>

</Console>

</Appenders>

<Loggers>

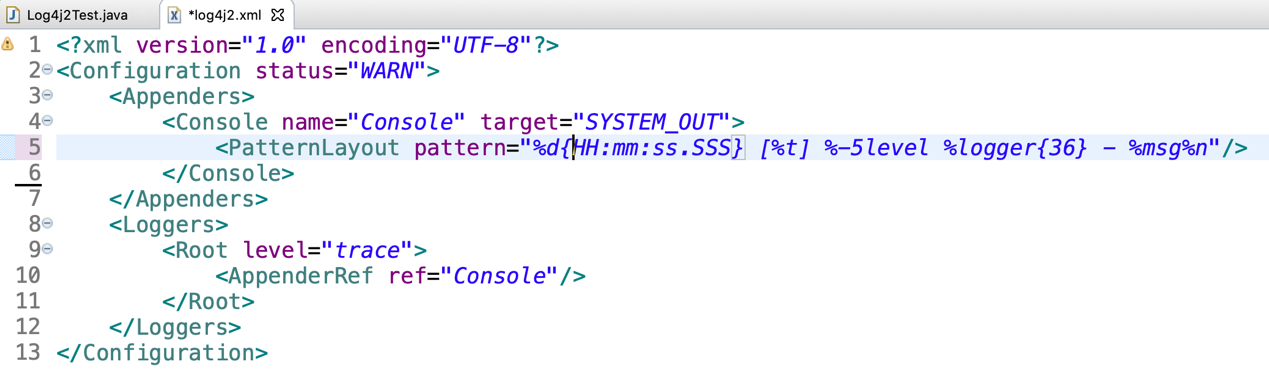
<Root level=*"trace"*>

<AppenderRef ref=*"Console"*/>

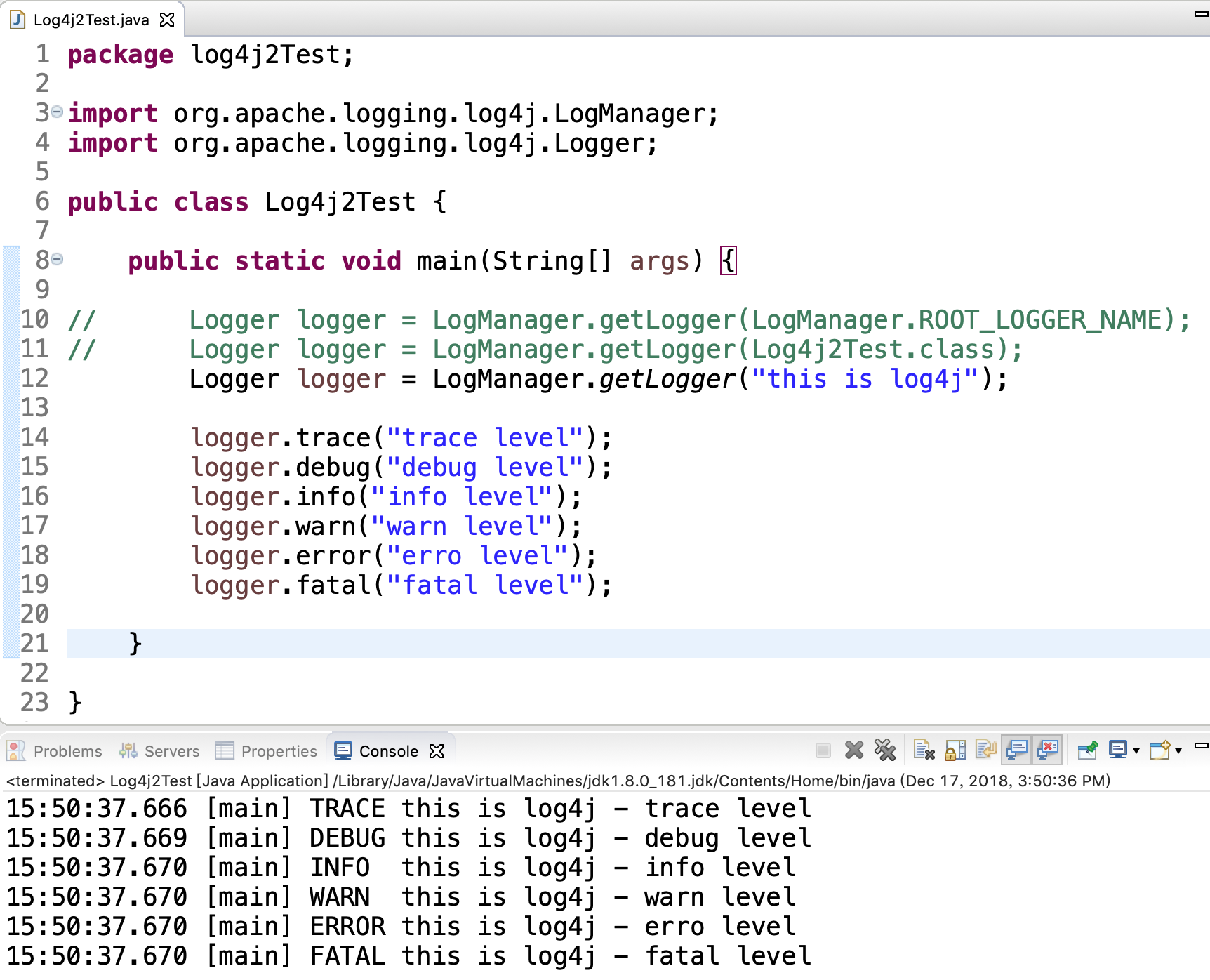
</Root>

</Loggers>

</Configuration>



运行即可



配置log4j2.xml将日志输出到指定文件：

修改配置文件如下：

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<Configuration status=*"WARN"*>

<Appenders>

<Console name=*"Console"* target=*"SYSTEM\_OUT"*>

<PatternLayout

pattern=*"%d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"* />

</Console>

<!-- 新建一个app.log文件 -->

<File name=*"FileAppender"* fileName=*"app.log"*>

<PatternLayout

pattern=*"%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"* />

</File>

<!-- 发现Async 好像PatternLayout的输出格式配置的和输出的格式不一样，不用异步就完全一样 -->

<Async name=*"AsyncAppender"*>

<AppenderRef ref=*"FileAppender"* />

</Async>

</Appenders>

<Loggers>

<Logger name=*"AsyncFileLogger"* level=*"trace"* additivity=*"true"*>

<AppenderRef ref=*"AsyncAppender"* />

</Logger>

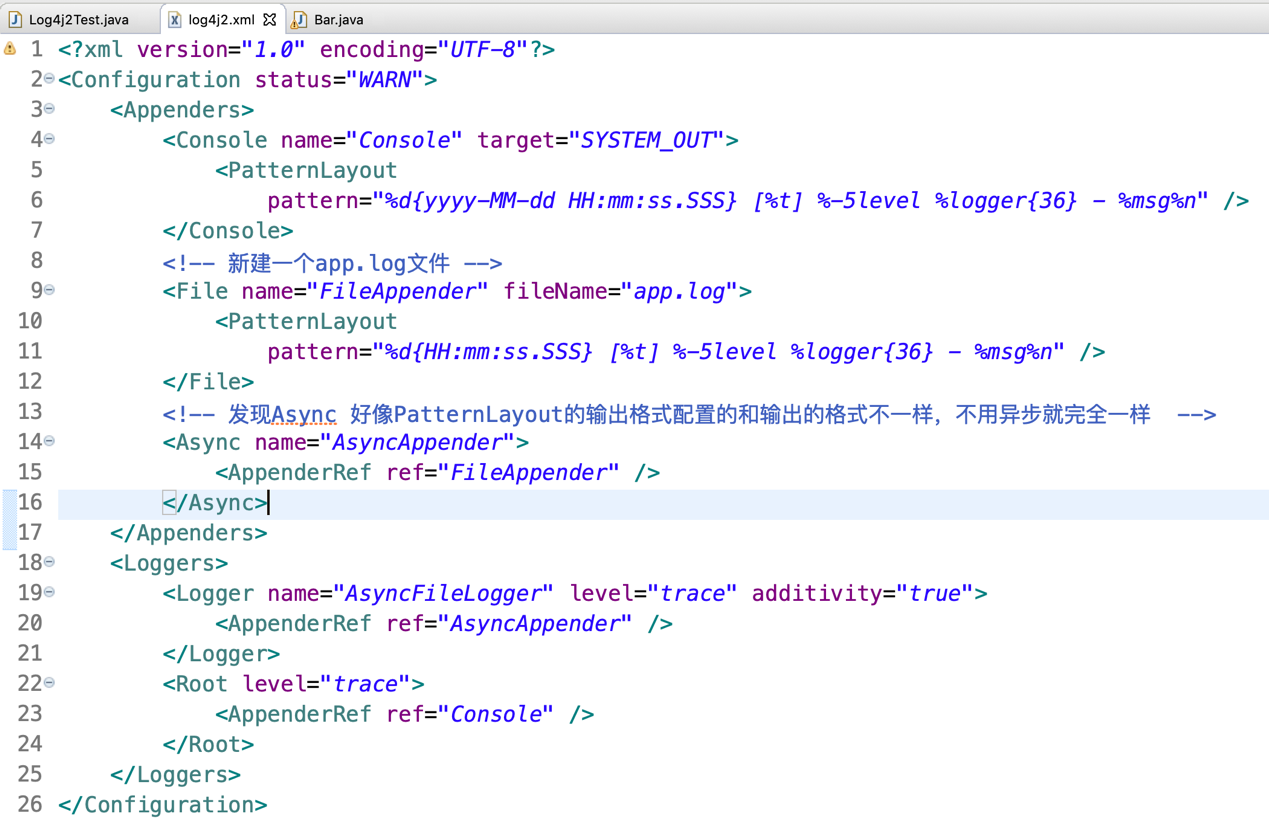
<Root level=*"trace"*>

<AppenderRef ref=*"Console"* />

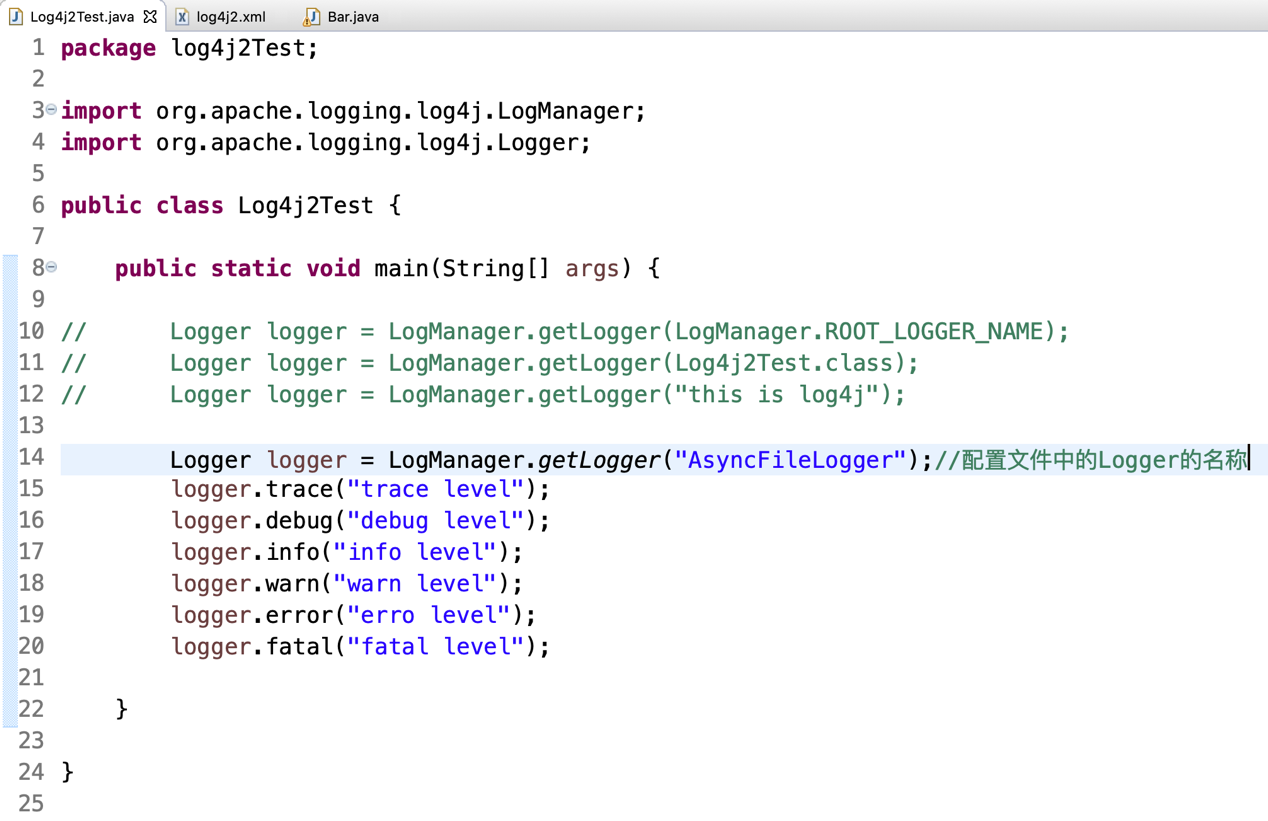
</Root>

</Loggers>

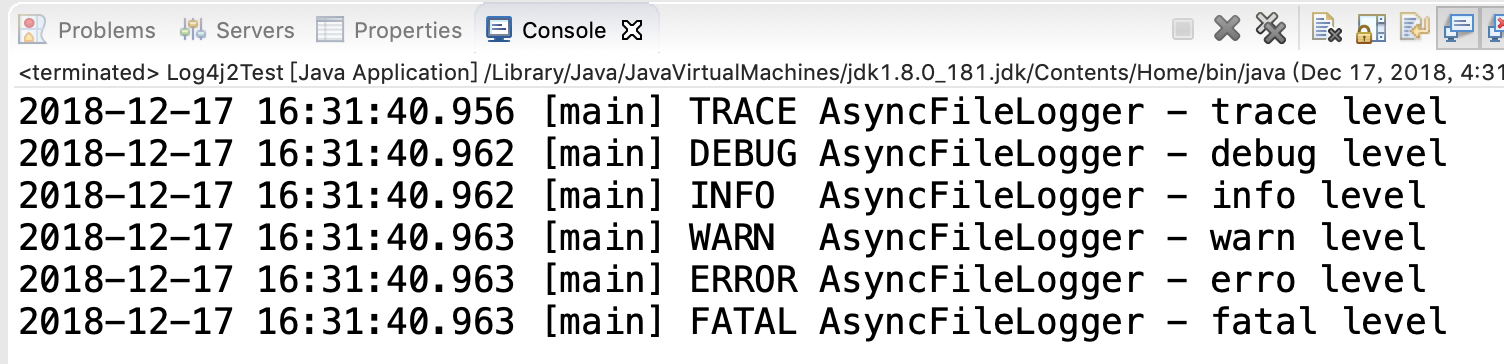
</Configuration>



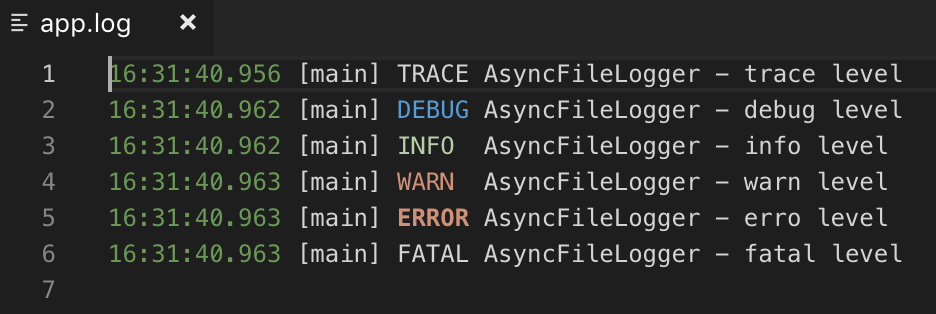
之后再次修改Log4j2Test.java文件



Run:

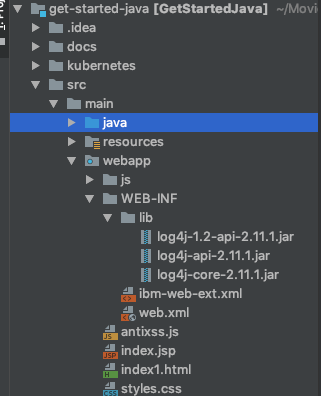


查看工程目录下面的app.log文件：

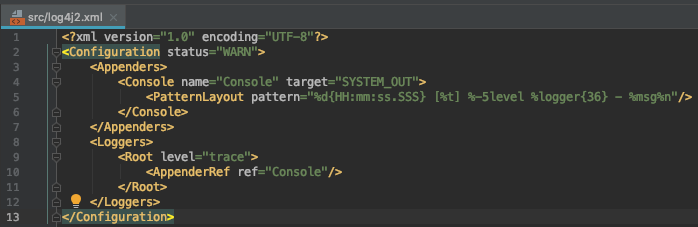


从IBM Cloud clone下来的get-start-java项目直接修改，加入一个servlet测试日志的产生情况。

导入jar包到lib（需要新建）

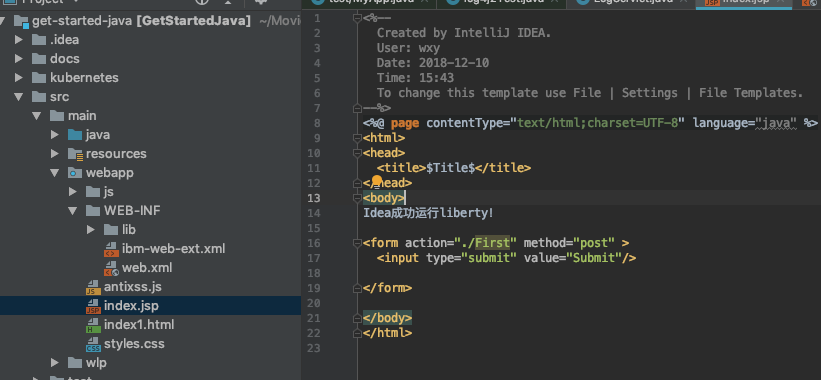


再src文件夹下面新家log4j2.xml文件：

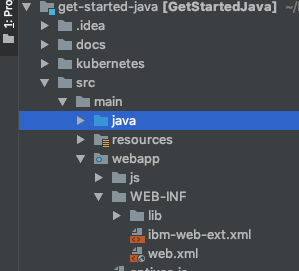


先添加一个index.jsp文件：

<%--  
 Created by IntelliJ IDEA.  
 User: wxy  
 Date: 2018-12-10  
 Time: 15:43  
 To change this template use File | Settings | File Templates.  
--%>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
 <head>  
 <title>$Title$</title>  
 </head>  
 <body>  
 Idea成功运行liberty！  
  
 <form action="./First" method="post" >  
 <input type="submit" value="Submit"/>  
  
 </form>  
  
 </body>  
</html>



先简单的直接添加一个web.xml文件，然后写上下面信息：

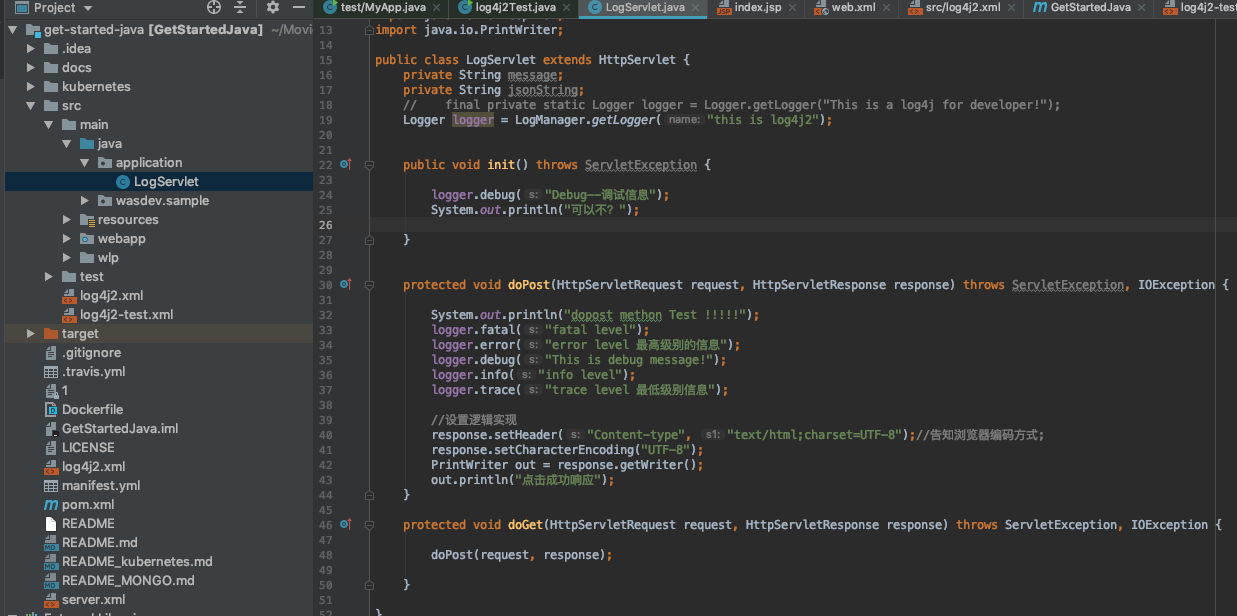


<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns="http://java.sun.com/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd "  
 version="2.5">  
  
 <welcome-file-list>  
 <welcome-file>index.jsp</welcome-file>  
 <welcome-file>index1.html</welcome-file>  
 </welcome-file-list>  
  
 <servlet>  
 <!-- 下面的这个servlet-name是随意命名的，只需要与下面的servlet-mapping对应起来就可以了 -->  
 <servlet-name>HelloWorld</servlet-name>  
 <servlet-class>application.LogServlet</servlet-class>  
 </servlet>  
  
 <servlet-mapping>  
 <servlet-name>HelloWorld</servlet-name>  
 <!-- 下面这个url是客户端访问的url，在form表格里面会有一个action = "./First"，就是为了拼成一个url，其中前面的点是表示当前的路径，可以认为是服务器的地址（http://localhost:8080/。当点击submit之后就会拼接成新的url也就是 http://localhost:8080/First。这个url根据web.xml配置文件中找到这个url应该由谁来处理。）-->  
 <url-pattern>/First</url-pattern>  
 </servlet-mapping>  
  
  
</web-app>

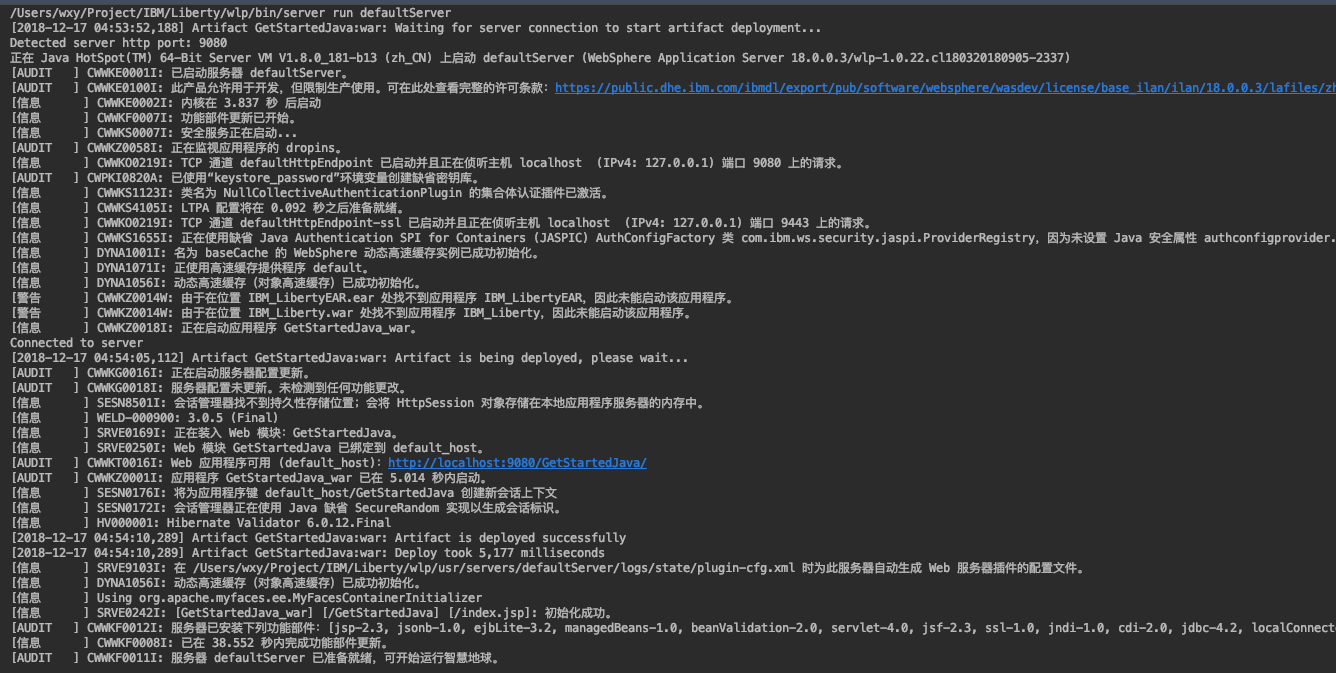


新建一个Servlet文件

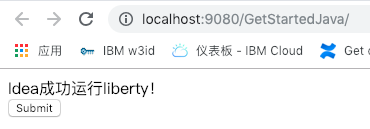
package application;  
  
//import org.apache.log4j.Logger;  
  
import org.apache.logging.log4j.LogManager;  
import org.apache.logging.log4j.Logger;  
  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.io.PrintWriter;  
  
public class LogServlet extends HttpServlet {  
 private String message;  
 private String jsonString;  
 // final private static Logger logger = Logger.getLogger("This is a log4j for developer!");  
 Logger logger = LogManager.*getLogger*("this is log4j2");  
  
  
 public void init() throws ServletException {  
  
 logger.debug("Debug--调试信息");  
 System.*out*.println("可以不？");  
  
 }  
  
  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 System.*out*.println("dopost methon Test !!!!!");  
 logger.fatal("fatal level");  
 logger.error("error level 最高级别的信息");  
 logger.debug("This is debug message!");  
 logger.info("info level");  
 logger.trace("trace level 最低级别信息");  
  
 //设置逻辑实现  
 response.setHeader("Content-type", "text/html;charset=UTF-8");//告知浏览器编码方式;  
 response.setCharacterEncoding("UTF-8");  
 PrintWriter out = response.getWriter();  
 out.println("点击成功响应");  
 }  
  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
  
 doPost(request, response);  
  
 }  
  
}



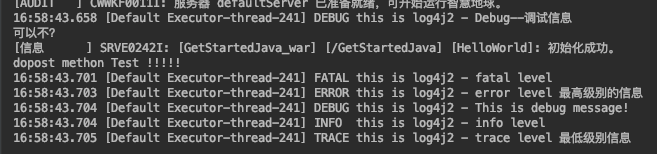
下面开始运行下：



网页显示如下：

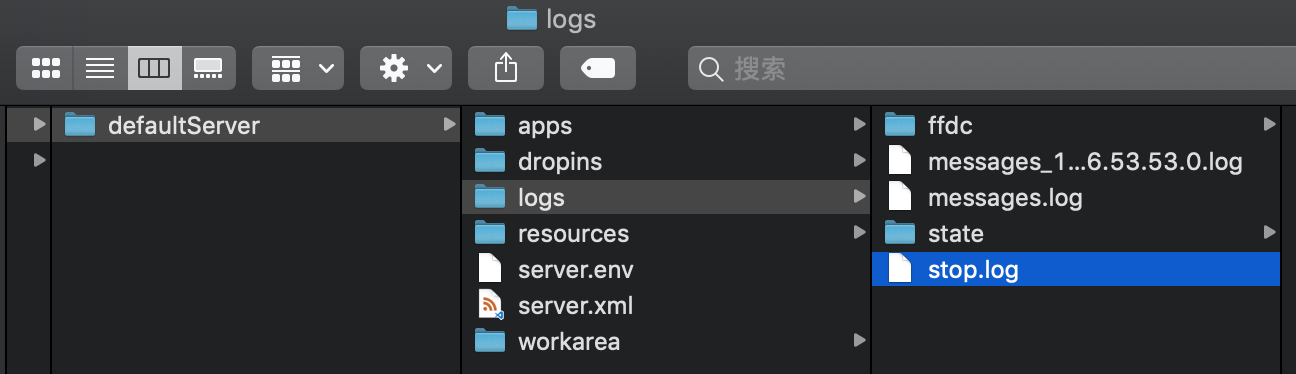




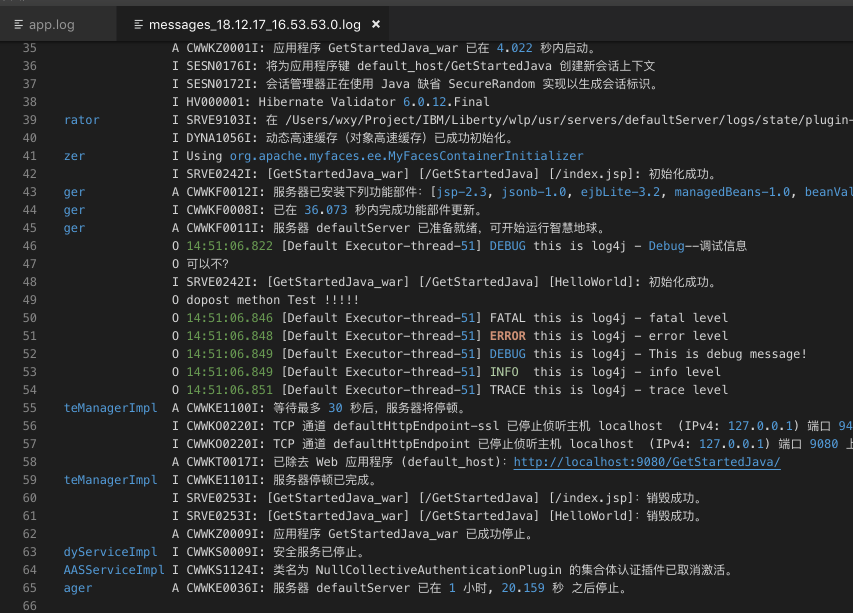


虽然上面并没有配置log4j2输出到文件，但是liberty还是把这些信息输出到message下面了。

查看liberty安装目录下面的logs文件夹下面的message\*.log信息如下：



因为message.log的文件格式太长，所以需要向右移动才看到信息，不容易找到：

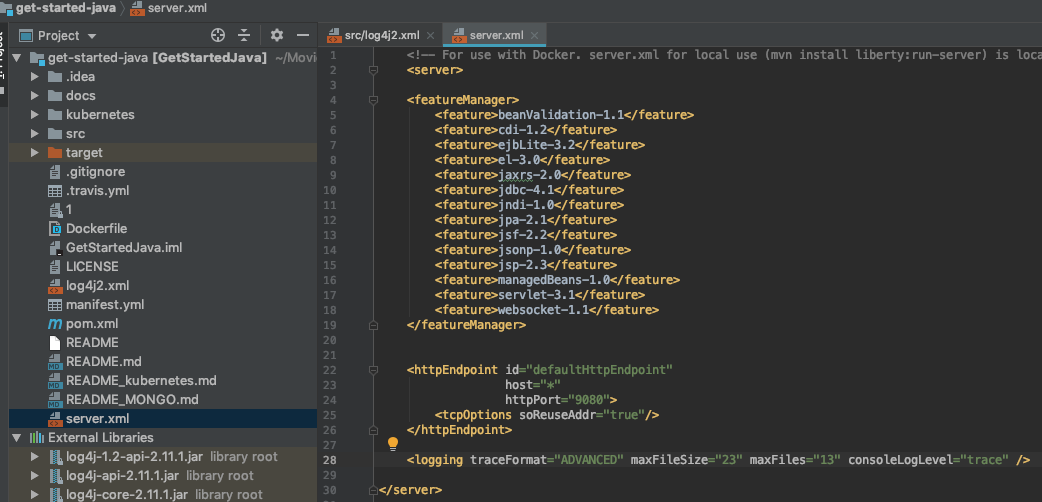


如果上面的信息没有显示所有log级别，建议修改server.xml配置文件！

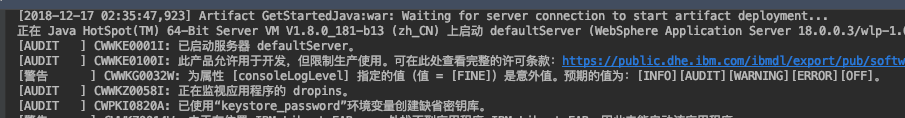
发现一个问题就是idea工程下面的server.xml文件好像是不起作用的

加入：

<logging traceFormat="ADVANCED" maxFileSize="23" maxFiles="13" consoleLogLevel="trace" />

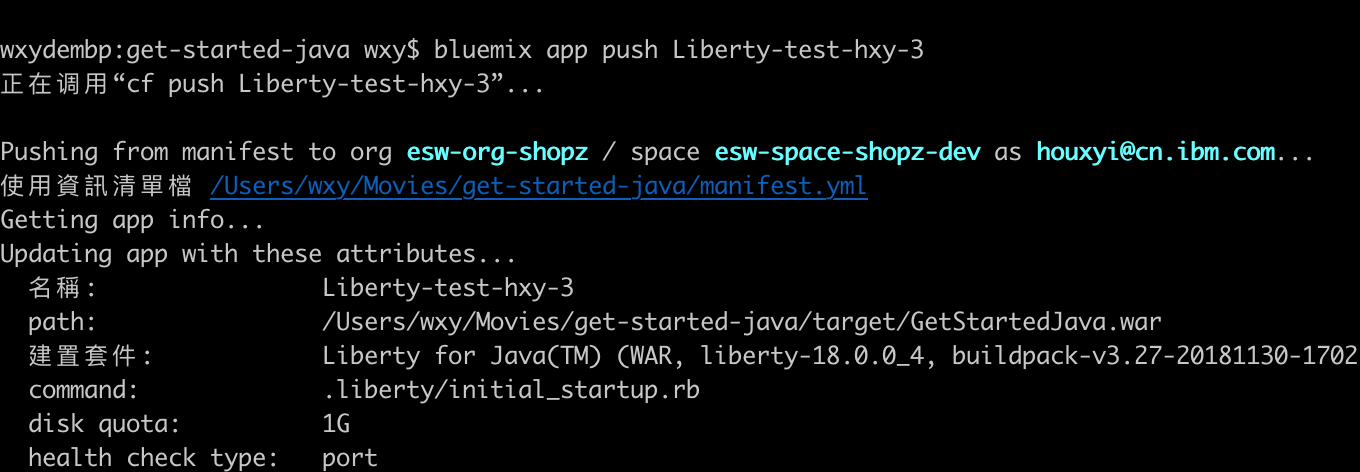


但是运行的时候可能会有下面的提示：

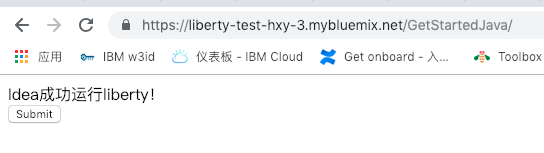


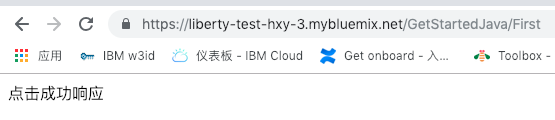
说FINE是个意外值，而这个值是我自己添加在liberty安装目录下面的server.xml文件中的，所以确定的就是最好还是直接修改liberty安装目录下面的配置文件server.xml。项目运行构建之后server.xml配置文件还是会打包在war包中，部署到云端运行也是没有问题的。

本地测试完成，将应用推送到cloud

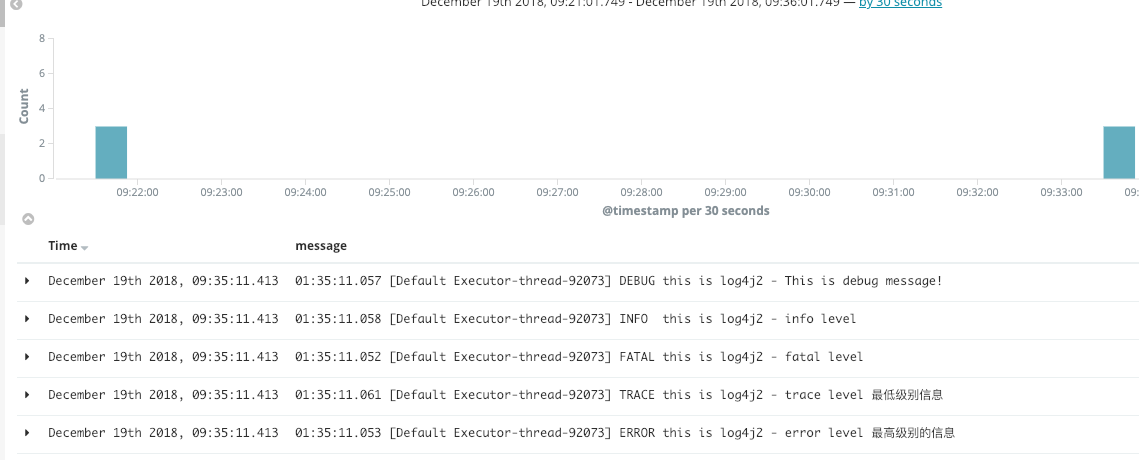


查看cloud的网页

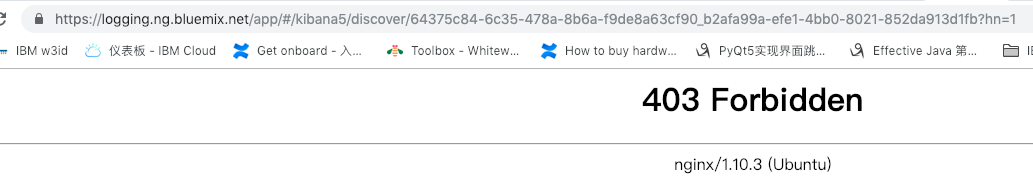




查看Kibana中的信息如下：



Kibana经常禁止：



查看cloud Foundry里面的ssh连接之后的应用里面的logs/message.log可以看到下面信息：



Hou Xiao Yi

Xiao Yi XY Hou/China/Contr/IBM

[Houxyi@cn.ibm.com](mailto:Houxyi@cn.ibm.com)

aohanhongyi

AVNYL4672

18010472947