Atitit java rest mvc微服务原理以及框架选型

目录

[第一节 Mvc原理 model controler view 1](#_Toc20911)

[第二章 Spark 最简单 1](#_Toc6933)

[第一节 Sprbt tomcat jetty swoole workman ci tp 1](#_Toc23522)

[第三章 自己实现 1](#_Toc12976)

[第一节 http halder 模式 1](#_Toc21016)

[第二节 Svlt模式 啰嗦 1](#_Toc31261)

[第四章 如何实现进程隔离微服务化多个app启动类 2](#_Toc23106)

[第一节 Mlt prj mode 2](#_Toc12385)

## Mvc原理 model controler view

Rest

微服务 通过同步异步模式调用跨进程调用 ，，rest ws socket mq等。。Cli

# Spark 最简单

## Sprbt tomcat jetty swoole workman ci tp

Sprbt 啰嗦，启动速度慢。。

Tomcat 但是和sprbt冲突，可以使用jetty

Jetty 只能code，

Tp ci yii lalavel等rest lib Codeigniter

Swoole Workerman

# 自己实现

## http halder 模式

## Svlt模式 啰嗦

# 如何实现进程隔离微服务化多个app启动类

## Mlt prj mode

多个app启动类即可了。。。同一份prj，但是多重启动。。不同的端口部署不同的微服务。。

**package** rest;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.eclipse.jetty.webapp.AbstractConfiguration;

**import** org.eclipse.jetty.webapp.Configuration;

**import** org.eclipse.jetty.server.Request;

**import** org.eclipse.jetty.server.Server;

**import** org.eclipse.jetty.server.handler.AbstractHandler;

**import** org.eclipse.jetty.servlet.ServletContextHandler;

**import** org.eclipse.jetty.servlet.ServletHolder;

**import** org.eclipse.jetty.webapp.FragmentConfiguration;

**import** org.eclipse.jetty.webapp.JettyWebXmlConfiguration;

**import** org.eclipse.jetty.webapp.MetaInfConfiguration;

**import** org.eclipse.jetty.webapp.WebAppContext;

**import** org.eclipse.jetty.webapp.WebInfConfiguration;

**import** org.eclipse.jetty.webapp.WebXmlConfiguration;

**import** com.fasterxml.classmate.AnnotationConfiguration;

**public** **class** JettyWebserver {

/\*\*

\* <pre>

\* 一个简单的 jetty 服务

\* </pre>

\*

\* **@author** saligia

\* **@date** 17-10-10

\*/

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

// int port = 13140;

// int port=Integer.parseInt(args[0].trim()) ;

Server server = **new** Server(80); // 创建一个 jetty 服务

// server.setHandler( new AbstractHandler () {

//

// @Override

// public void handle(String target, Request rq, HttpServletRequest htrq, HttpServletResponse rsp)

// throws IOException, ServletException {

// System.out.println("target :" + target);

// System.out.println("request : " + rq.getRequestURI());

// System.out.println("requestServ : " + htrq.getRequestURI());

//

// }});

WebAppContext context = **new** WebAppContext();

context.setResourceBase("webapps/");

// context.setDescriptor("web/WEB-INF/web.xml");

context.setConfigurations(**new** Configuration[]{

**new** WebXmlConfiguration(),

**new** WebInfConfiguration(),

**new** MetaInfConfiguration(),

**new** FragmentConfiguration(),**new** JettyWebXmlConfiguration()});

// AbstractConfiguration

// , new AnnotationConfiguration(), new EnvConfiguration()

context.setContextPath("/");

context.setParentLoaderPriority(**true**);

server.setHandler(context);

//------------ add svlt is ok...

ServletContextHandler context = **new** ServletContextHandler(ServletContextHandler.***SESSIONS***);

context.setContextPath("/ct2");

context.addServlet(**new** ServletHolder(**new** HtmlSvlt()), "/html");

// context.addServlet(new ServletHolder(new Iphone()), "/iphone");

server.setHandler(context);

server.start();

server.dumpStdErr();

server.join();

}

/\*\*

\* <pre>

\* </pre>

\*

\* **@author**

\* **@param** target - 目标请求，可以是一个URI或者是一个转发到这的处理器的名字

\* **@param** request - Jetty自己的没有被包装的请求，一个可变的Jetty请求对象

\* **@param** httpServletRequest - 被filter或者servlet包装的请求，一个不可变的Jetty请求对象

\* **@param** httpServletResponse - 响应，可能被filter或者servlet包装过

\* **@return**

\*/

}