链滴

Latke 配置剖析

作者：88250

原文链接：https://ld246.com/article/1474087427032

来源网站：链滴

本贴主要介绍了 Latke 的关键配置及其设计用意，所以无论你是 Latke 的**开发者**还是基于 Latke 开的产品的**使用者**，希望本贴能够帮助到你。

为了说明方便，我们会使用 Solo 博客系统作为主要的应用场景进行举例。

latke.properties

文件路径是 /latke.properties，该配置文件是必须的。通常情况其中的配置项比较少，最简化的情况只需要配置 #### Server #### 部分的 serverScheme，其他项都有默认值，不需要显示设置。

该文件的配置项均可通过启动参数进行覆盖，所以如无必要请勿修改配置文件本身，具体参数名和配项对应关系可通过 --help 进行查看。

Server 部分

部分 Solo 用户在初始化时会遇到“配置错误”或者上线后界面样式显示异常的问题，这是因为 latke.rops 配置不当造成的。该配置文件中 #### Server #### 部分的配置如下：

```
#### Server ####
# Browser visit protocol
serverScheme=http
# Browser visit domain name
serverHost=localhost
# Browser visit port, 80 as usual, THIS IS NOT SERVER LISTEN PORT!
serverPort=8080
```

这 3 个配置项需要配置为用户通过浏览器**访问时候**的值。换句话说，如果你的服务在本机启动，那么认的配置是可以让你在本机通过 http://localhost:8080 访问时一切正常的；但非本机访问时（比如过 http://domain-or-ip:8080） 就**不能**正常加载静态资源了。

解决方案：将这三个配置项的值调整为最终访问时候对应的样子。

比如我的博客域名是 myblog.com，该域名已经正常解析到服务器 IP，此时只需要将 serverHost 的设置为 myblog.com 就可以通过 http://myblog.com:8080 访问了。要进一步消除后面的端口有两方式：

1. 前置 HTTP 服务器做反向代理（比如通过 NGINX）
2. 将 Solo 启动时监听端口调整为 80/443，或者留空

推荐第一种方式，因为 Java 进程主要处理的是动态请求。HTTPS 或者是静态资源请求应该交由更专的 HTTP 处理引擎来做，这样能减少很多复杂的配置（比如配置 Java 的 SSL 证书），也能充分优化能（比如静态资源由 NGINX 处理，配置缓存、限流等）。

一个正确的配置示例如下：

```
#### Server ####
# Browser visit protocol
serverScheme=http
# Browser visit domain name
serverHost=myblog.com
# Browser visit port, 80 as usual, THIS IS NOT SERVER LISTEN PORT!
serverPort=
```

请注意其中 serverPort 项并没有赋值，因为这样就能够使用 HTTP 的浏览器默认端口 80 了，HTTPS 也可以这样留空。

对于进程监听端口，默认使用的是 8080，**只能通过启动参数 listen_port 进行配置。**对于端口部分配置，总结一下就是：

- server_port 指定访问端口：一般是 80 或者 443，如果是这两个浏览器默认端口的话把这个参数值设为空，即 --server_port=
- listen_port 指定容器进程监听端口，默认是 8080，主要用于提供给 NGINX 进行反向代理，如果 80 端口没有其他进程占用的话用默认的就行

Static Server 部分

该部分通常情况是没有显示配置的（比如 Solo 里面默认的配置文件就没有出现这部分），需要显示置的情况是需要做**动静分离**的应用场景。

前面我们提到过可以通过 NGINX 作为前置服务器，将静态资源请求分离出来进行处理。其具体的分规则可以按 path 或后缀。这个动静分离的层次是在具体服务器节点上的分离，是用户请求到达内部络后的处理。

我们可以在用户浏览器请求时就进行动静分离，将静态资源请求分发到配置好的 CDN 服务上，这个DN 服务一般是一个域名地址。而 staticServerScheme、staticServerHost、staticServerPort这三就是用来配置该地址的，如果没有显示配置，则这三项会分别使用 serverScheme、serverHost、sererPort 的值。

其他配置项

下面这些配置项也是在某些场景才会用到，一般来说也不用单独配置的。

项	说明	默认值	备注
runtimeMode	运行模式，可选值 DEVELOPMENT 或 PRODUCTION	RODUCTION	线上环境一定要使用 PRODUCTION
staticResourceVersion	静态资源版本号，主要用于刷缓存	认取服务启动的时间毫秒	
contextPath	上下文路径	空	无必要请勿配置
staticPath	静态资源服务的上下文路径	同 context ath	显示配置的话一般都是留空

其中“上下文路径”是 Java Servlet 中的概念，指的是请求 URL 上第一层目录路径，比如对于 http:/mydomain.com/solo/start 而言 /solo 是上下文路径，这个如果没有必要请不要配置增加复杂度。

配置后如果还遇到问题，请打开浏览器 F12 查看网络请求进行调试。

local.properties

文件路径是 /local.properties，该文件是**必须的**，主要用于配置数据库。比如在 Solo 中默认的配置下：

MySQL runtime

```
runtimeDatabase=MYSQL
jdbc.username=root
jdbc.password=123456
jdbc.driver=com.mysql.cj.jdbc.Driver
jdbc.URL=jdbc:mysql://localhost:3306/solo?useUnicode=yes&characterEncoding=UTF-8&use
SL=false&serverTimezone=UTC

#### H2 runtime ####
#runtimeDatabase=H2
#jdbc.username=root
#jdbc.password=
#jdbc.driver=org.h2.Driver
#jdbc.URL=jdbc:h2:~/solo_h2/db;MODE=MYSQL

# The minConnCnt MUST larger or equal to 3
jdbc.minConnCnt=5
jdbc.maxConnCnt=10

# The specific table name prefix
jdbc.tablePrefix=b3_solo
```

其中 #### H2 runtime #### 和 #### MySQL runtime #### **只能启用一种**，该部分配置了使用一种数据库实现。其他的配置项比较好识别，一般来说使用默认值就好。

H2

如果启用了 H2 数据库，则 jdbc.URL 中可以配置数据持久化的文件路径，比如 jdbc:h2:~/solo_h2/db 使用的路径就是 操作系统用户 home （即 ~ ）下的 solo_h2/db 目录，在 Java 中对于的系统变量是 ${user.home}，如果你不大熟悉 Unix-like 的表示法，也可以将该路径配置成完整的绝对路径，比如 jbc:h2:D:/solo_h2/db。

static-resources.xml

文件路径是 /static-resources.xml，该配置文件是**必须的**，用于配置静态资源路径。路径的匹配模式 Ant-style 匹配，简单说来就是：

- * 匹配多个字符
- ? 匹配单个字符
- ** 匹配多层目录

如果你需要**添加自己的静态资源**（比如 HTML、MP3 等）就需要修改一下该文件。

环境变量

测试环境、生产环境如果需要加载不同的配置文件，可通过设置操作系统环境变量来指定配置文件的对路径。

- LATKE_PROPS：latke.properties 文件的绝对路径
- LATKE_LOCAL_PROPS：local.properties 文件的绝对路径

优先通过环境变量加载，如果没有定义该环境变量则从类路径下加载。