FtpServer 配置详解

FtpServer 配置是基于 XML 的，它匹配了 XML Schema，在 XML 编辑器里可以方便的修改进行配置。XML 文档结构如下：

```
<server xmlns="http://mina.apache.org/ftpserver/spring/v1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://mina.apache.org/ftpserver/spring/v1
http://mina.apache.org/ftpserver/ftpserver-1.0.xsd"
    id="myServer">
</server>
```

其中 id 是必需的，设置服务器的标识。

在 server 元素里面，可以添加新的元素配置 server 实例中的组件，如 listeners（监听器）和 user managers 用户管理器。

1 与 Spring 整合

Apache FtpServer 使用了 Spring 框架实现配置，因此从 Spring 框架中可以得到很多配置的好处。例如，可以将 server 元素置于 Spring 配置文件的任何位置，当 FtpServer 在 classpath 下时，Spring 可以为你启动 FtpServer。

2 server 元素

一些配置是针对整个 server 的，影响到所有的 listener 和登陆限制，强加的 listener 登陆人数。采用 XML 配置，可有的配置项如下：

```
<server xmlns="http://mina.apache.org/ftpserver/spring/v1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://mina.apache.org/ftpserver/spring/v1
http://mina.apache.org/ftpserver/ftpserver-1.0.xsd"
    id="server"
    max-logins="500"
    anon-enabled="false"
    max-anon-logins="123"
    max-login-failures="124"
```

```
    login-failure-delay="125"
    >
...
</server>
```

Server 元素详解

元素	描述	是否必需	默认值
id	XML 配置文件中的唯一标识符	Yes	
max-logins	最大并发用户数	No	10
max-anon-logins	最大匿名用户数	No	10
anon-enabled	是否允许匿名用户登陆	No	true
max-login-failures	连接关闭前登陆失败后尝试次数	No	3
login-failure-delay	登陆失败后的时间延迟（ms），用于限制配置暴力破解密码	No	500

3 listener 元素

Listener 是 FtpServer 中负责监听网络 socket 接口，当客户端连接后创建用户 session、执行命令等。一个 FtpServer 可以同时拥有多个 listerner，以下是一些例子：

1）一个在 21 端口的默认监听；

2）一个在 21 端口的超速通道监听，一个隐式的在 22 端口上的 SSL 监听

3）一个在内部 IP 超速通道监听，一个外部 IP 的 SSL 监听

Listener 以 name 标识，默认 Listener 的名称是“default”。

主要的网络配置表现在 listener 上，例如监听端口、SSL 配置。

在 XML 配置格式内，所有的监听器以“listeners”元素进行配置。因此一个 XML 配置看起来像：

```
<listeners>
        <nio-listener name="default" port="2222" implicit-ssl="true" idle-timeout="60" local-address="1.2.3.4">
            <ssl>
              <keystore file="mykeystore.jks" password="secret" key-password="otherSecret" />
                    <truststore file="mytruststore.jks" password="secret"/>
            </ssl>
            <data-connection idle-timeout="60">
              <active enabled="true" local-address="1.2.3.4" local-port="2323" ip-check="true"/>
              <passive ports="123-125" address="1.2.3.4" external-address="1.2.3.4" />
            </data-connection>
            <blacklist>1.2.3.0/16, 1.2.4.0/16, 1.2.3.4</blacklist>
    </nio-listener>
</listeners>
```

nio 元素

装配 FtpServer 的监听器叫做 nio-listener，因为它是基于 Java NIO 的以提升性能和规模。

元素	描述	是否必需	默认值
name	监听名称，如果是“default”，它会覆盖默认的监听器	Yes	
port	监听器接受连接的端口	No	21
local-address	监听器绑定的服务器地址	No	All available
implicit-ssl	如果使用隐式的 SSL，其值为 true	No	false
idle-timeout	非活跃客户端断开时间。如果值设为 0,则闲置时间不可用(一个客户端可以永远闲置，如果没有从服务器断开）如果一个较低的最大闲置时间配置给用户(比如使用 PropertiesUserManager 闲置时间配置)，它会覆盖 listener 值.这样 listener 值强加以更高的阈值 ，但是低层值可以提供给每个用户。	No	300

ssl 元素

对于需要 FTPS 支持的 Listener 必须。

元素	描述	是否必需	默认值
protocol	使用 SSL 协议，支持的值有“SSL”和“TLS”	No	TLS
client-authentication	是否客户端认证？支持的值有：“NEED”， “WANT”和“NONE”	No	NONE
enabled-ciphersuites	对于本连接的一个可用的逗号分隔的密码组件列表。可用的加密组件依赖于所采用的Java版本， 这里 是Sun's JSSE 提供.	No	所有的加密组件均可用

keystore 元素

如果提供了 ssl 元素，它将是必须的。它为密钥存储提供配置，用于查找私钥和服务器证书。

元素	描述	是否必需	默认值
file	密钥存储路径	Yes	
password	密钥存储密码	Yes	
key-password	密钥存储中的密钥密码	No	密钥存储密码
key-alias	密钥存储中 Key 的别名	No	Uses first key found
type	密钥存储类型	No	JKS
algorithm	密钥存储算法	No	SunX509

truststore

该元素用于配置信任存储，以定位信任的证书。

元素	描述	是否必需	默认值

file	密钥存储路径	Yes	
password	密钥存储密码	No	无密码可读取证书
type	密钥存储类型	No	JRE 密码钥存储，一般情况为 JKS
algorithm	密钥存储算法	No	SunX509

data-connection 元素

为数据连接提供配置

元素	描述	是否必需	默认值
idle-timeout	闲置数据连接关闭时间	No	300

active 元素

为积极数据连接提供配置

元素	描述	是否必需	默认值
enabled	如果积极数据连接不被允许，值为 false	No	true
local-address	服务器创建数据连接时使用的本地地址	No	Any available
local-port	服务器创建数据连接时使用的本地端口	No	Any available
ip-check	服务器是否检查数据连接 IP 否与控制 socket 的 IP 相同？	No	false

passive 元素

为消极数据连接提供配置

元素	描述	是否必需	默认值
ports	服务器创建消极数据连接时使用的本地端口	No	Any available port
address	服务器监听消极数据连接的地址	No	与控制 socket 地址相同
external-address	服务器声明在 PASV 答复中监听的地址。当服务器在 NAT 防火墙后，客户端看到的 IP 不同于服务回使用的 IP 时很有用。	No	

blacklist 元素

4 User Manager

FtpServer 同时支持文件方式和数据库方式的用户管理。

4．1 个性化 User Manager

可以编写自己的 UserManager 整合应用，自编写的用户管理器需要实现 org.apache.ftpserver.ftplet.UserManager 接口。在 Spring 配置文件中，需要 bean 元素来配置你的 UserManager。这给你了所有 Spring 的优势，可以融合其他的 bean。也可以根据 Spring 的扩展机制，提供自己的 XML 格式。

4．2 数据库用户管理方式

可以将用户信息存储在数据库中，JBDC 用于访问数据库。这个 User Manager 已经在 MYSQL、HSQLDB 和 FireBird 数据库下进行了测试。所有的信息存储在 FTP_USER 表中，一个示例的数据库 DDL 文件在<INSTALL_DIR>/res/ftp-db.sql 下。需要将 JDBC 包加入到 classpath 下。

示例：

```
<db-user-manager encrypt-passwords="salted">
        <data-source>
            <beans:bean class="some.datasoure.class" />
        </data-source>
        <insert-user>INSERT INTO FTP_USER (userid, userpassword,
            homedirectory, enableflag, writepermission, idletime, uploadrate,
            downloadrate) VALUES ('{userid}', '{userpassword}', '{homedirectory}',
            '{enableflag}', '{writepermission}', {idletime}, {uploadrate},
            {downloadrate})</insert-user>
        <update-user>UPDATE FTP_USER SET

    userpassword='{userpassword}',homedirectory='{homedirectory}',enableflag={enableflag},
writepermission={writepermission},idletime={idletime},uploadrate={uploadrate},downloadrate=
{downloadrate}
            WHERE userid='{userid}'</update-user>
        <delete-user>DELETE FROM FTP_USER WHERE userid = '{userid}'
        </delete-user>
        <select-user>SELECT userid, userpassword, homedirectory,
            enableflag, writepermission, idletime, uploadrate, downloadrate FROM
            FTP_USER WHERE userid = '{userid}'</select-user>
        <select-all-users>SELECT userid FROM FTP_USER ORDER BY userid
        </select-all-users>
        <is-admin>SELECT userid FROM FTP_USER WHERE userid='{userid}' AND
            userid='admin'</is-admin>
        <authenticate>SELECT userpassword from FTP_USER WHERE
userid='{userid}'</authenticate>
    </db-user-manager>
```

4.2.1 配置参数

db-user-manager element

元素	描述	是否必需	默认值
encrypt-passwords	它表明了密码是如何存储的。可用值为：clear 普通文本，MD5 哈希算法或者 salted 哈希盐化密码。建议采用 salted。	No	md5
子元素	描述	是否必需	默认值
data-source	普通 Spring bean 配置的数据源	Yes	
insert-user	SQL 语句插入一个新的用户，所有的动态值在运行时会被取代。	Yes	
update-user	SQL 语句插入修改一个用户，所有的动态值在运行时会被取代。	Yes	
delete-user	SQL 语句删除一个用户，所有的动态值在运行时会被取代。	Yes	
select-user	SQL 语句查询一个用户，所有的动态值在运行时会被取代。	Yes	
select-all-users	SQL 语句插入所有用户，所有的动态值在运行时会被取代。	Yes	
is-admin	查询一个用户是否是管理员用户，所有的动态值在运行时会被取代。	Yes	
authenticate	QL 语句认证一个用户，所有的动态值在运行时会被取代。	Yes	

4.2.2 数据源配置

数据源必须按数据库提供方描述那样配置，可以用 BasicDataSource 来满足一般的要求，它是由 Apache Commons DBCP 项目组提供。

使用 BasicDataSource 连接 Mysql。

```
<data-source>
    <beans:bean class="org.apache.commons.dbcp.BasicDataSource">
        <beans:property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <beans:property name="url" value="jdbc:mysql://localhost/ftpd" />
        <beans:property name="username" value="myuser" />
        <beans:property name="password" value="secret" />
        </beans:bean>
</data-source>
```

4.2.3FTP_USER 表结构

Column	Type	Default value
userid	VARCHAR(64), Primary key	

userpassword	VARCHAR(64)	
homedirectory	VARCHAR(128)	
enableflag	BOOLEAN	TRUE
writepermission	BOOLEAN	FALSE
idletime	INT	0
uploadrate	INT	0
downloadrate	INT	0
maxloginnumber	INT	0
maxloginperip	INT	0

4.2.4 基于文件的用户管理

这是默认的用户管理方式，使用 properties 文件管理所有的用户。

示例：

```
<file-user-manager file="users.properties" encrypt-passwords="true" />
```

file-user-manager 元素

元素	描述	是否必需	默认值
file	存储用户的 properties 文件路径	Yes	
encrypt-passwords	加密方式，具体见前面	No	md5

5 配置消息端口

当客户端需要一个消极端口连接时，服务器端应当提供相应的端口以供连接使用。默认情况下，FtpServer会选择任何可以利用的端口。然而，这也可以被使用passive data connection配置覆盖。允许的的消极端口可以被指定到一个端口（如 20020），多个端口（如 20020，20030，20040），或者一个范围（20020-20030）。默认的一个开放关闭的范围，以 1 开始，65535 结束；例如范围 60000-，是指从 60000-65535。任何一个指定的值或者范围的组合，都可以使用。

当服务器端使用完所有的端口（一个客户端处理一个数据转换），下一个客户端必须等待可用端口。因此，建议采用多个端口。

如果一个值（某个指定端口，或者一个范围值的开始或者结束），在允许的 0-65535 之外，则在服务器启动的时候就会报错。

示例：

123	端口 123 作为 passive port
123,133	端口 123 和 133 作为 passive port
123-125	从 123 到 125 内任何一个端口都可以作为 passive port
123-125, 127, 129-130	从 123 到 125 内任何一个端口, 端口 127 或从 129 到 130 内的端口都可以作为 passive port
0	任意一个端口均可作为 passive port

6 TLS-SSL Support

这里解释了怎么使 Apache FTP Server 使用 Transport Layer Security (TLS)来加密服务器-客户端的通信。

FtpServer 使用 Java Secure Sockets Extension (JSSE) 基础组件提供 TLS/SSL socket。JSSE 与一些提供商 Java 发布版一起。对于这些发布版本，请遵从供应商的指示配置 JVM 以使用 JSSE 服务。

6．1 安全模式

显式安全模式

在这种模式下服务器支持安全和非安全的连接，一旦得到客户端请求（通过 SSL 认证），服务器端转向 SSL/TLS 模式。在这种情况下，服务器端不应该采用隐式 SSL（默认情况下值）：

```
<nio-listener name="default" implicit-ssl="false">
```

隐式安全模式

采用隐式的 SSL 连接，即总是在控制 socket 上可用。需要做的第一件事情就是告诉 listener：

```
<nio-listener name="default" implicit-ssl="true">
```

如果设置采用隐式的 SSL 连接，应该考虑数据连接的隐式安全。

6．2 数据连接安全

隐式安全监听器并不保证加密数据转换。要在数据连接中使用 SSL/TLS 客户端要么必须发送“PROT P”命令，要么隐式安全也能在数据连接中可用。

```
<data-connection implicit-ssl="true">
```

如果没有显式地配置 SSL 密钥存储（keystores）和信任存储（truststores）为数据连接，它将从 listener 中继承。这是基本的配置。

不同的 FTP 客户端动作不同，关于隐式的数据连接，一些臆想了一个 SSL socket，而一些总是发送“PROT P”命令，下面表格中展示了一些客户端的特性，请报告一些其他的：

FTP 客户端	动作
FileZilla	在隐式模式下自动发送 "PROT P"命令
DartFTP/PowerTCP	臆想了一个 SSL socket