链滴

封装 ftp-spring-boot-starter 并实现文件上传

作者：kangaroo1122
原文链接：https://ld246.com/article/1614095490514
来源网站：链滴

1 准备

- FTP服务器，匿名or用户名/密码均可

2 封装 ftp-spring-boot-starter

2.1 配置类FtpProperties.java

```
package com.coctrl.ftp.configuration;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;

/**
 * 类 FtpProperties 功能描述:
 *
 * @author kangaroo hy
 * @version 0.0.1
 * @date 2020/11/10
 */
@Component
@ConfigurationProperties(prefix = "coctrl.ftp")
public class FtpProperties {

    /**
     * 服务器地址，默认 localhost
     */
    private String host = "localhost";

    /**
```

```
     * 通信端口，默认 21
     */
    private Integer port = 21;

    /**
     * 用户名
     */
    private String username = "";

    /**
     * 密码
     */
    private String password = "";

    /**
     * 文件上传路径，支持一级，可以不用 /
     *  如：test，不用：/test
     */
    private String path = "";

    public String getHost() {
        return host;
    }

    public void setHost(String host) {
        this.host = host;
    }

    public Integer getPort() {
        return port;
    }

    public void setPort(Integer port) {
        this.port = port;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getPath() {
        return path;
```

```
    }

    public void setPath(String path) {
        this.path = path;
    }
}
```

2.2 自动装配类FtpAutoConfiguration.java

```
package com.coctrl.ftp.configuration;

import com.coctrl.ftp.service.FtpService;
import org.springframework.boot.autoconfigure.condition.ConditionalOnClass;
import org.springframework.boot.autoconfigure.condition.ConditionalOnMissingBean;
import org.springframework.boot.autoconfigure.condition.ConditionalOnProperty;
import org.springframework.boot.context.properties.EnableConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * @author kangaroo hy
 * @date 2020/4/25
 * @desc ftp-spring-boot-starter
 * @since 0.0.1
 */
@Configuration
@EnableConfigurationProperties(FtpProperties.class)
@ConditionalOnClass(FtpService.class)
@ConditionalOnProperty(prefix = "coctrl.ftp", value = "enabled", matchIfMissing = true)
public class FtpAutoConfiguration {
    private final FtpProperties properties;

    public FtpAutoConfiguration(FtpProperties properties) {
        this.properties = properties;
    }

    @Bean
    @ConditionalOnMissingBean(FtpService.class)
    public FtpService ftpService() {
        return new FtpService(properties);
    }
}
```

2.3 服务类FtpService.java

```
package com.coctrl.ftp.service;

import com.coctrl.ftp.configuration.FtpProperties;
import org.apache.commons.net.ftp.FTP;
import org.apache.commons.net.ftp.FTPClient;
import org.apache.commons.net.ftp.FTPReply;
```

```
import org.apache.log4j.Logger;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.Queue;
import java.util.concurrent.ConcurrentLinkedQueue;

/**
 * 类 FtpService 功能描述:
 *
 * @author kangaroo hy
 * @version 0.0.1
 * @date 2020/11/10
 */
@Service
public class FtpService {
    public static final Logger logger = Logger.getLogger(FtpService.class);

    private final FtpProperties properties;

    private Queue<FTPClient> queue = new ConcurrentLinkedQueue<>();

    public FtpService(FtpProperties properties) {
        this.properties = properties;
    }

    public boolean uploadFile(String fileName, InputStream inputStream) {
        FTPClient ftpClient = connect();
        if (ftpClient != null) {
            try {
                boolean b = ftpClient.storeFile(fileName, inputStream);
                if (b) {
                    logger.info("上传成功");
                } else {
                    logger.info("上传失败");
                }
                inputStream.close();
                return b;
            } catch (IOException e) {
                logger.info("上传失败：" + e.getMessage());
            } finally {
                disconnect(ftpClient);
            }
        }
        return false;
    }

    private FTPClient connect() {
        FTPClient client = queue.poll();
        if (client != null) {
            return client;
        }
```

```
        FTPClient ftpClient = new FTPClient();
        ftpClient.setControlEncoding("utf-8");
        try {
            ftpClient.connect(properties.getHost(), properties.getPort());
            if (StringUtils.isEmpty(properties.getUsername()) || StringUtils.isEmpty(properties.getP
ssword())) {
                ftpClient.login("anonymous", null);
            } else {
                ftpClient.login(properties.getUsername(), properties.getPassword());
            }
            int replyCode = ftpClient.getReplyCode();
            if (!FTPReply.isPositiveCompletion(replyCode)) {
                logger.error("连接服务器失败");
                throw new IOException("不能连接到FTP服务器：" + properties.getHost());
            }
            logger.info("连接服务器成功");
            ftpClient.setFileType(FTP.BINARY_FILE_TYPE);
            ftpClient.enterLocalPassiveMode();
            boolean b = ftpClient.makeDirectory(properties.getPath());
            if (b) {
                logger.info("路径创建成功");
            } else {
                logger.info("路径已存在，直接使用");
            }
            ftpClient.changeWorkingDirectory(properties.getPath());
            queue.add(ftpClient);
            return ftpClient;
        } catch (IOException e) {
            logger.error("连接服务器成功", e);
            return null;
        }
    }

    void disconnect(FTPClient client) {
        queue.add(client);
    }
}
```

2.4 spring.factories

```
# Auto Configure
org.springframework.boot.autoconfigure.EnableAutoConfiguration = \
  com.coctrl.ftp.configuration.FtpAutoConfiguration
```

2.6 pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
MLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xs
/maven-4.0.0.xsd">
```

```
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.coctrl</groupId>
    <artifactId>ftp-spring-boot-starter</artifactId>
    <version>0.0.1</version>
    <name>ftp-spring-boot-starter</name>
    <description>ftp-spring-boot-starter</description>

    <properties>
        <java.version>1.8</java.version>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <spring-boot.version>2.1.12.RELEASE</spring-boot.version>
        <commons-net.version>3.6</commons-net.version>
        <slf4j.version>1.7.25</slf4j.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-autoconfigure</artifactId>
            <version>${spring-boot.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-configuration-processor</artifactId>
            <version>${spring-boot.version}</version>
        </dependency>
        <dependency>
            <groupId>commons-net</groupId>
            <artifactId>commons-net</artifactId>
            <version>${commons-net.version}</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>log4j-over-slf4j</artifactId>
            <version>${slf4j.version}</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>${slf4j.version}</version>
        </dependency>
    </dependencies>

</project>
```

3 使用

将starter打包到本地maven仓库，在项目中引用即可。如果不配置用户名或密码，则以匿名方式连接TP服务器，否则，用户名/密码方式连接FTP服务器。