# 常用类

## CLientGlobal

用于加载配置文件的公共客户端工具。

常用方法：

- `init(String conf_filename);`根据配置文件路径及命名，加载配置文件并设置客户端公共参数，配置文件类型为 conf 文件。可以使用绝对路径或相对路径加载。
- `initByProperties(Properties props);`根据 Properties 对象设置客户端公共参数。

注意：**使用 conf 或 properties 进行客户端参数配置时，参数 key 命名不同。**

## TrackerClient

跟踪器客户端类型。创建此类型对象时，需传递跟踪器组，就是跟踪器的访问地址信息。无参构造方法默认使用 ClientGlobal.g_tracker_group 常量作为跟踪器组来构造对象。

创建对象的方式为：

`new TrackerClient();`或`new TrackerClient(ClientGlobal.g_tracker_group)`

## TrackerServer

跟踪器服务类型。此类型的对象是通过跟踪器客户端对象构建的。实质上就是一个与FastDFS Tracker Server 的链接对象。是代码中与 Tracker Server 链接的工具。

构建对象的方式为：

```
trackerClient. getTrackerServer ();
```

## StorageServer

存储服务类型。此类型的对象是通过跟踪器客户端对象构建的。实质上就是一个与FastDFS Storage Server 的链接对象。是代码中与 StroageServer 链接的工具。获取的具体存储服务链接，是由 Tracker Server 分配的，所以构建存储服务对象时，需要依赖跟踪器服务对象。

构建对象的方式为：

```
trackerClient.getStoreStorage(trackerServer);
```

## StorageClient

存储客户端类型。此类型的对象是通过构造方法创建的。创建时，需传递跟踪服务对象和存储服务对象。此对象实质上就是一个访问 FastDFS Storage Server 的客户端对象，用于实现文件的读写操作。

创建对象的方式为：

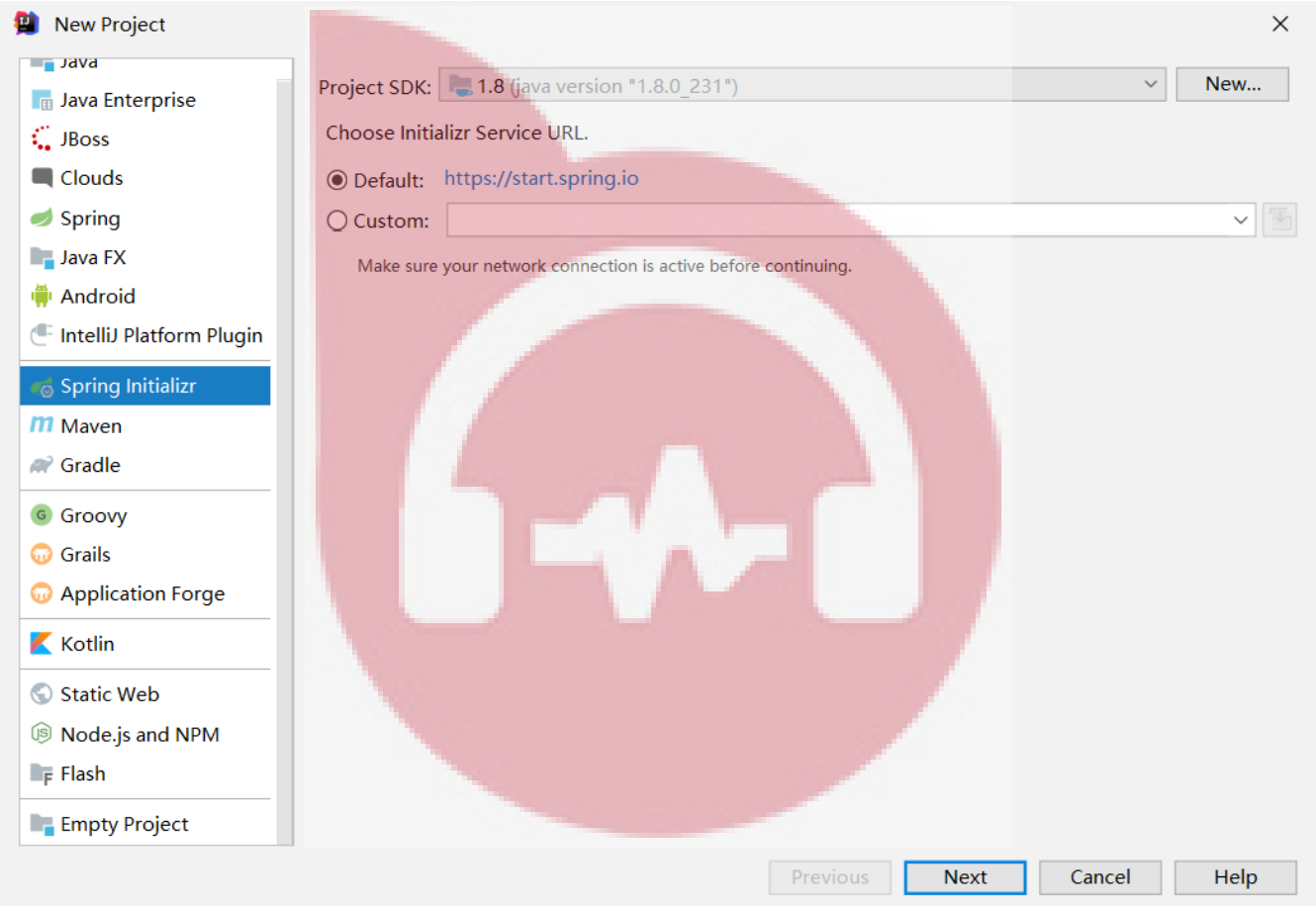
```
new StorageClient(trackerServer, storageServer);
```

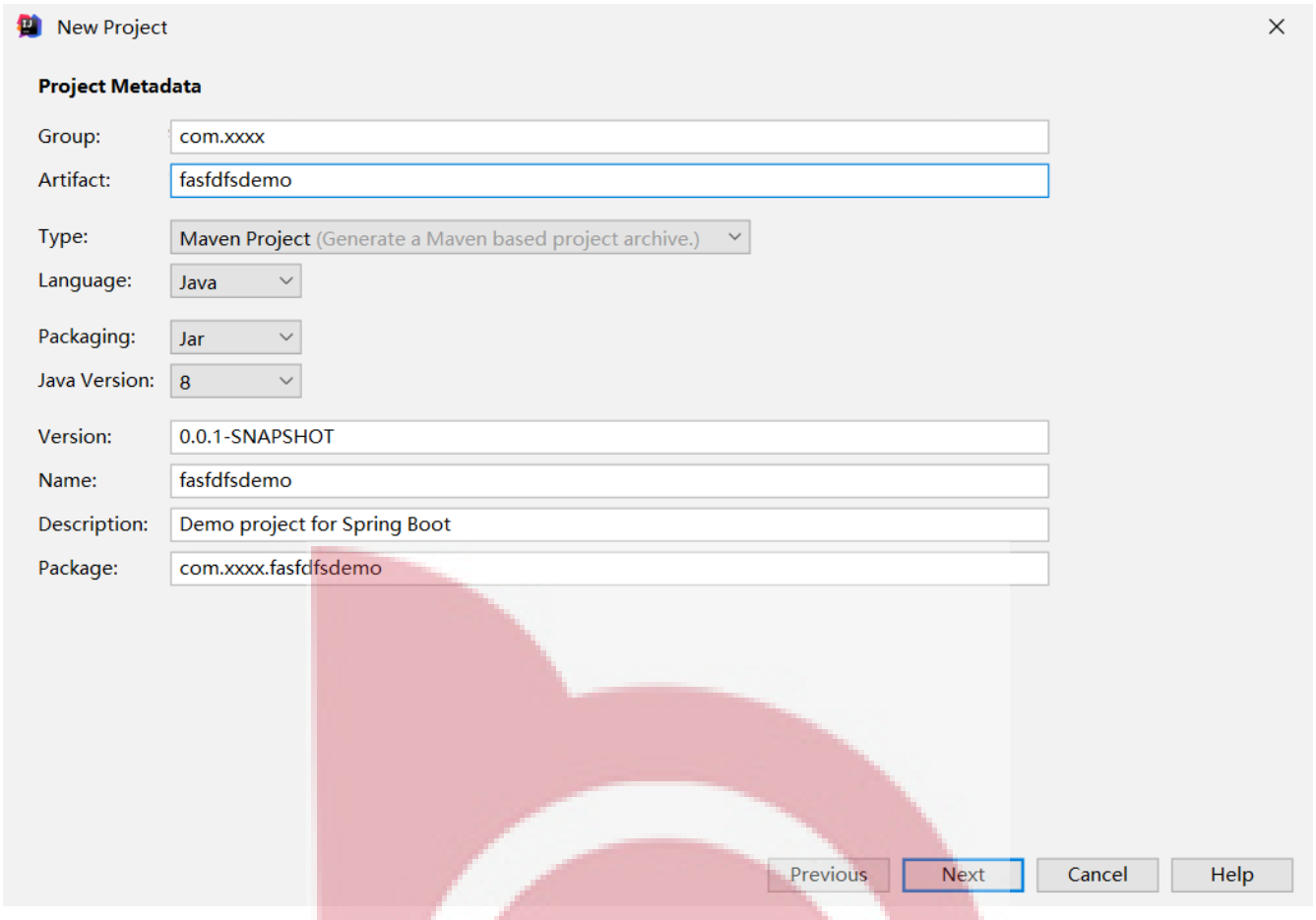
常用方法有：

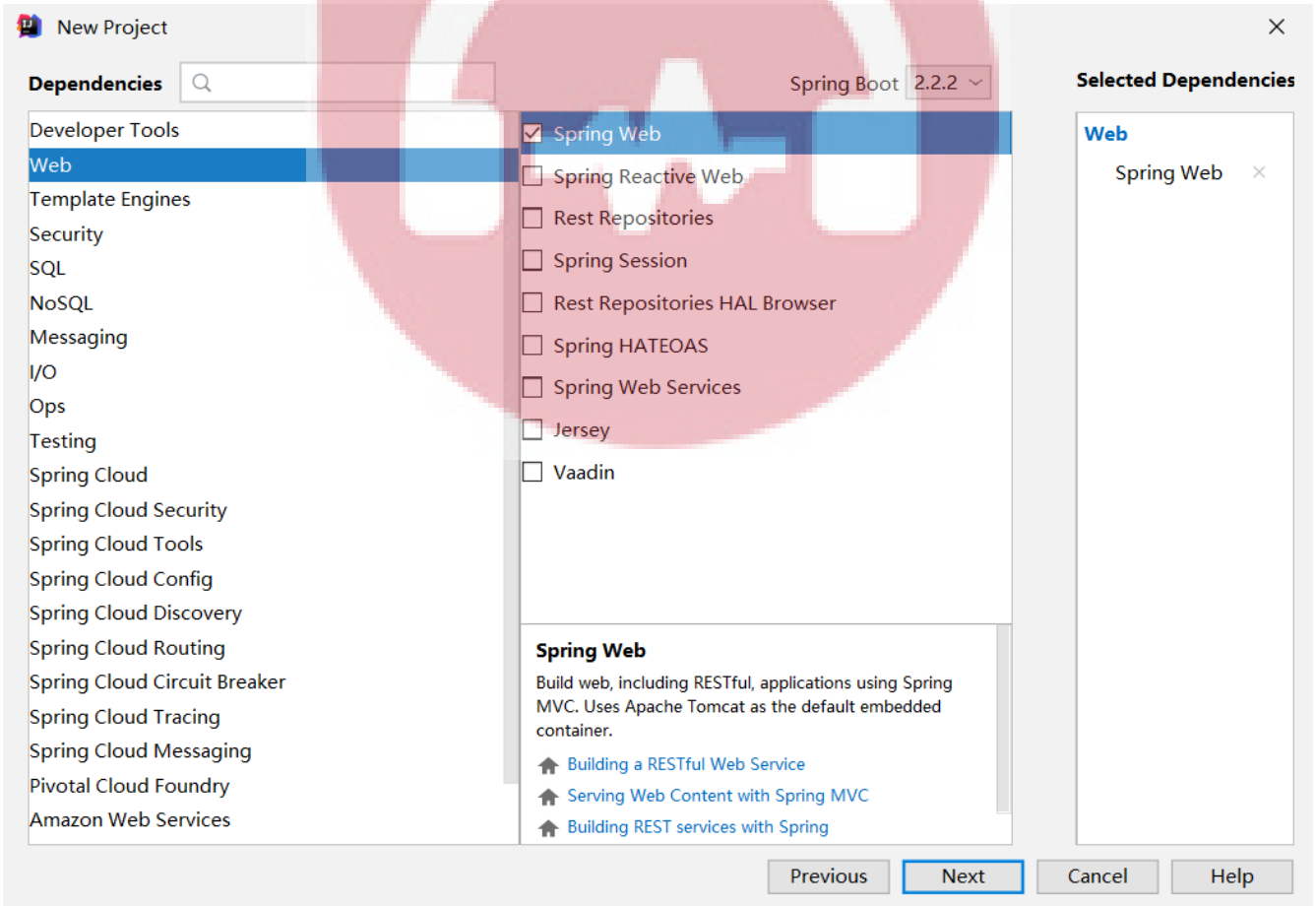
- `upload_file(String local_filename, String file_ext_name, NameValuePair[] meta_list);`上传文件的方法，参数 local_filename 为要上传的本地文件路径及文件名，可使用绝对路径或相对路径；参数 file_ext_name为上传文件的扩展名，如果传递null，则自动解析文件扩展名；参数 meta_list 是用于设置上传文件的源数据的，如上传用户、上传描述等。
- `download_file(String group_name, String remote_file_name);`下载文件的方法，参数group为组名/卷名，就是 Storage Server中/etc/fdfs/storage.conf 配置文件中配置的group_name 参数值，也是要下载的文件所在组/卷的命名；参数 remote_file_name 为要下载的文件的路径及文件名。
- `delete_file(String group_name, String remote_file_name);`删除文件的方法，参数含义同 download_file方法参数。

# Demo

## 创建项目




New Project
Project Metadata
Group: com.xxxx
Artifact: fasfdfsdemo
Type: Maven Project (Generate a Maven based project archive.)
Language: Java
Packaging: Jar
Java Version: 8
Version: 0.0.1-SNAPSHOT
Name: fasfdfsdemo
Description: Demo project for Spring Boot
Package: com.xxxx.fasfdfsdemo
Previous
Next
Cancel
Help


New Project
Dependencies
Spring Boot 2.2.2
Selected Dependencies
Developer Tools
Web
Template Engines
Security
SQL
NoSQL
Messaging
I/O
Ops
Testing
Spring Cloud
Spring Cloud Security
Spring Cloud Tools
Spring Cloud Config
Spring Cloud Discovery
Spring Cloud Routing
Spring Cloud Circuit Breaker
Spring Cloud Tracing
Spring Cloud Messaging
Pivotal Cloud Foundry
Amazon Web Services
Spring Web
Spring Reactive Web
Rest Repositories
Spring Session
Rest Repositories HAL Browser
Spring HATEOAS
Spring Web Services
Jersey
Vaadin
Web
Spring Web
Spring Web
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
Building a RESTful Web Service
Serving Web Content with Spring MVC
Building REST services with Spring
Previous
Next
Cancel
Help

## 添加依赖

```
<!--FastDFS依赖-->
<dependency>
   <groupId>org.csource</groupId>
   <artifactId>fastdfs-client-java</artifactId>
   <version>1.29-SNAPSHOT</version>
</dependency>
<!--thymeleaf 依赖-->
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

## 在resources目录下添加配置文件

fdfs_client.conf

注1：tracker_server指向您自己IP地址和端口，1-n个

注2：除了tracker_server，其它配置项都是可选的

```
#连接超时
connect_timeout = 2
#网络超时
network_timeout = 30
#编码格式
charset = UTF-8
#tracker端口号
http.tracker_http_port = 8080
#防盗链功能
http.anti_steal_token = no
#秘钥
http.secret_key = FastDFS1234567890
#tracker ip: 端口号
tracker_server = 192.168.10.100:22122
#连接池配置
connection_pool.enabled = true
connection_pool.max_count_per_entry = 500
connection_pool.max_idle_time = 3600
connection_pool.max_wait_time_in_ms = 1000
```

## 添加FastDFS文件对象

FastDFSFile.java

```
package com.xxxx.fastdfsdemo;

import java.util.Arrays;

/**
 * 文件上传对象类
 * @author zhoubin
 * @since 1.0.0
```

```
 */
public class FastDFSFile {
    private String name;
    private byte[] content;
    private String ext;
    private String md5;
    private String author;
    private String height;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public byte[] getContent() {
        return content;
    }

    public void setContent(byte[] content) {
        this.content = content;
    }

    public String getExt() {
        return ext;
    }

    public void setExt(String ext) {
        this.ext = ext;
    }

    public String getMd5() {
        return md5;
    }

    public void setMd5(String md5) {
        this.md5 = md5;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getHeight() {
        return height;
    }
```

```
    public void setHeight(String height) {
        this.height = height;
    }

    public FastDFSFile() {
    }

    public FastDFSFile(String name, byte[] content, String ext) {
        this.name = name;
        this.content = content;
        this.ext = ext;
    }

    @Override
    public String toString() {
        return "FastDFSFile{" +
                "name='" + name + '\'' +
                ", content=" + Arrays.toString(content) +
                ", ext='" + ext + '\'' +
                ", md5='" + md5 + '\'' +
                ", author='" + author + '\'' +
                ", height='" + height + '\'' +
                '}';
    }
}
```