いろんな OSS ライブラリを超簡単に使っちゃおうフレームワーク

～おもに接続まわり～

JACUG（日本 Apache Camel ユーザ会）

java で FTP 送信


java ftp
約 167,000,000 件（0.11 秒）
他のキーワード: java ftp転送 java ftp socket java ftpライブラリ java ftpサンプル
java ftp受信
Jakarta Commons NETによるFTP処理（FTPClient） - Java入門
www.syboos.jp › ホーム › Java IO › ネットワーク関連 - キャッシュ
2009年1月28日 – Jakarta Commons NET（FTPClient）によるFTP処理（ファイルのアップロード、ダウンロード）の例です。,Java入門,オープンソース,オープンソースソフトウェア,OSS,
javaでFTPする方法は、大きく二つあります
www.searchman.info/tips/2630.html - キャッシュ
javaでFTPする方法は、大きく二つあります. <IT技術の処方箋：javaでFTPする方法は、大きく二つあります>. javaプログラムの中で、FTPサーバーにファイルを転送(put)したり、取得(get)するためには、どうしたらよいのでしょうか？ 方法としては、大きく分けて二 ...
javaでFTP(cooomons-net)
www.searchman.info/tips/2640.html - キャッシュ
<IT技術の処方箋：javaでFTP(cooomons-net)>. javaでFTPをする方法は、大きくわけて二つあります。http://www.searchman.info/tips/2630.html そのうち、commons-netを使ってFTPするサンプルを掲載しておきます。import java.io.FileInputStream; ...

Javaで実装するサンプル

```
//ファイルアップロード
public static void sendFile (String host,
        int port,
        String user,
        String password,
        String remoteFilename,
        InputStream is
        ) throws Exception {
    FTPClient ftpclient = new FTPClient();

    try {
        //指定するホスト、ポートに接続します
        ftpclient.connect(host, port);
        int reply = ftpclient.getReplyCode();
        if (!FTPReply.isPositiveCompletion(reply)) {
            //接続エラー時処理
            Exception ee = new Exception("Can't Connect to :" + host);
            throw ee;
        }

        //ログイン
        if (ftpclient.login(user, password) == false) {
            // invalid user/password
            Exception ee = new Exception("Invalid user/password");
            throw ee;
        }

        //ファイル転送モード設定
        ftpclient.setFileType(FTP.BINARY_FILE_TYPE);
        //ftpclient.cwd("filetype=pdf");

        //ファイル転送
        ftpclient.storeFile(remoteFilename, is);
        //"ファイル転送完了

        // ファイル受信
        FileOutputStream fos = new FileOutputStream("localfile");
        ftpclient.retrieveFile("remotefile", fos);

    } catch (IOException e) {
        //TODO エラー処理
        throw e;
    } finally {
        try {
            ftpclient.disconnect(); //接続解除
        } catch (IOException e) {
        }
    }

}
```

出典：Jakarta Commons NETによるFTP処理（FTPClient） - Java入門

OSSライブラリを個別に勉強する

Apache Commons™
http://commons.apache.org/

commons net™

Last Published: 02 June 2011 | Version: 3.0.1

ApacheCon | Apache | Commons

Apache Commons Net

Apache Commons Net™ library implements the client side of many basic Internet protocols. The purpose of the library is to provide fundamental protocol access, not higher-level abstractions. Therefore, some of the design violates object-oriented design principles. Our philosophy is to make the global functionality of a protocol accessible (e.g., TFTP send file and receive file) when possible, but also provide access to the fundamental protocols where applicable so that the programmer may construct his own custom implementations (e.g, the TFTP packet classes and the TFTP packet send and receive methods are exposed).

Features

Supported protocols are:

- FTP/FTPS
- NNTP
- SMTP(S)
- POP3(S)
- Telnet
- TFTP
- Finger
- Whois
- rexec/rcmd/rlogin
- Time (rdate) and Daytime
- Echo
- Discard
- NTP/SNTP

Background

Apache Jakarta Commons Net started as a commercial Java library called NetComponents, originally developed by ORO, Inc. in the early days of Java. After its 1.3.8 release in 1998, the source code was donated to the Apache Software Foundation and made available under the Apache License. Since then, many programmers have contributed to the continued development of Jakarta Commons Net. The current version numbering scheme bears no relation to the old. In other words, Jakarta Commons Net 1.0 succeeded and supplanted NetComponents 1.3.8. Apache Jakarta Commons is now an independent project and is called Apache Commons.

Further Information

For more info, see the JavaDoc, or look at some of the following articles:

- http://www.informit.com/guides/content.asp?g=java&seqNum=40 Jakarta Commons - Net Class Library
- http://www.onjava.com/pub/a/onjava/2003/06/25/commons.html?page=3 Using the Jakarta Commons, Part 1
- http://safari.phptr.com/0131478303/ch04 Apache Jakarta Commons: Reusable Java Components

Camel で FTP 送信

```
// Camel の初期化
....

//  送信データを作成
  data = ....

// FTP の設定
String endpoint = ”ftp://user@host:port/path?password=pass&binary=true”;

//  送信☆～
producer.send(endpoint, data);
```

Camel でメール

```
// Camel の初期化
....

//  送信データを作成
  data = ....

//  プロトコルの設定
String endpoint = ”smtp://.....”;

//  送信☆～
producer.send(endpoint, data);
```

Camel で Twitter

```
// Camel の初期化
....

// 送信データを作成
  data = ....

// プロトコルの設定
String endpoint = ”twitter://.....”;

// 送信☆～
producer.send(endpoint, data);
```

Camel で Hadoop（HDFS）

```
// Camel の初期化
....

//  送信データを作成
  data = ....

//  プロトコルの設定
String endpoint = ”hdfs://.....”;

//  送信☆～
producer.send(endpoint, data);
```

Camel でコマンド実行

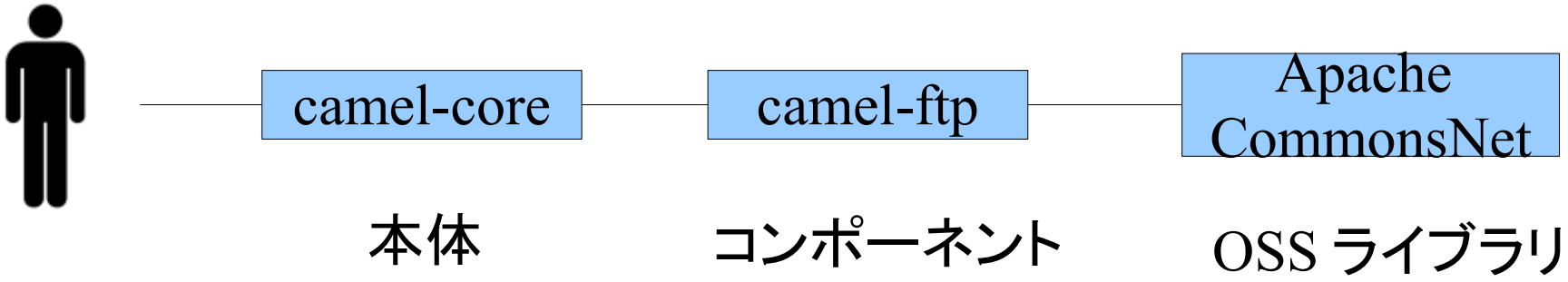
```
// Camel の初期化
....

//  送信データを作成
  data = ....

//  プロトコルの設定
String endpoint = ”exec://.....”;

//  送信☆～
producer.send(endpoint, data);
```

Camel は OSS ライブラリを使っている



URI の書き方

Camel includes the following Component implementations via URIs.

Component / ArtifactId / URI	Description
AHC / camel-ahc `ahc:hostname:[port]`	To call external HTTP services using Async Http Client
AMQP / camel-amqp `amqp:[topic:]destinationName`	For Messaging with AMQP protocol
APNS / camel-apns `apns:notify[?options]`	For sending notifications to Apple iOS devices
Atom / camel-atom `atom:uri`	Working with Apache Abdera for atom integration, such as consuming an atom feed.
AWS-SDB / camel-aws `aws-sdb://domainName[?options]`	For working with Amazon's SimpleDB (SDB).
AWS-SES / camel-aws `aws-ses://from[?options]`	For working with Amazon's Simple Email Service (SES).
AWS-SNS / camel-aws `aws-sns://topicname[?options]`	For Messaging with Amazon's Simple Notification Service (SNS).
AWS-SQS / camel-aws `aws-sqs://queuename[?options]`	For Messaging with Amazon's Simple Queue Service (SQS).
AWS-S3 / camel-aws `aws-s3://bucketname[?options]`	For working with Amazon's Simple Storage Service (S3).

FTP の URI オプション

More options
See File2 for more options as all the options from File2 is inherited.

Name	Default Value	
username	null	Specifies the username to use to log in to the remote file systen.
password	null	Specifies the password to use to log in to the remote file system.
binary	false	Specifies the file transfer mode, BINARY or ASCII. Default is ASCII (false).
disconnect	false	**Camel 2.2:** Whether or not to disconnect from remote FTP server right after use. Ca have a consumer which you want to stop, then you need to stop the consumer/rout
localWorkDirectory	null	When consuming, a local work directory can be used to store the remote file conten thus can conserve memory. See below for more details.
passiveMode	false	**FTP and FTPS only**: Specifies whether to use passive mode connections. Default is
securityProtocol	TLS	**FTPS only:** Sets the underlying security protocol. The following values are defined: TLS: Transport Layer Security SSL: Secure Sockets Layer
disableSecureDataChannelDefaults	false	**Camel 2.4: FTPS only**: Whether or not to disable using default values for execPbsz a options execPbsz and execProt should be used.
execProt	null	**Camel 2.4: FTPS only**: Will by default use option P if secure data channel defaults h C: Clear S: Safe (SSL protocol only) E: Confidential (SSL protocol only) P: Private
execPbsz	null	**Camel 2.4: FTPS only**: This option specifies the buffer size of the secure data chan
isImplicit	false	**FTPS only:** Sets the security mode(implicit/explicit). Default is explicit (false).
knownHostsFile	null	**SFTP only:** Sets the known_hosts file, so that the SFTP endpoint can do host key verif
privateKeyFile	null	**SFTP only:** Set the private key file to that the SFTP endpoint can do private key ver

実際に動くコード

```
public class Test {
    public static void main(String args[]) throws Exception {
        // 初期化
        CamelContext ctx = new DefaultCamelContext();
        ctx.start();
        ProducerTemplate producer = ctx.createProducerTemplate();

        // 送信データの作成
        Exchange exchange = new DefaultExchange(ctx);
        String data = "AAAAA";
        exchange.getIn().setBody(data);

        // 送信
        producer.send("file://e:/_work",exchange);
    }
}
```

実際に動くコード

```
public class Test {
    public static void main(String args[]) throws Exception {
        // 初期化
        CamelContext ctx = new DefaultCamelContext();
        ctx.start();
        ProducerTemplate producer = ctx.createProducerTemplate();

        // 送信データの作成
        Exchange exchange = new DefaultExchange(ctx);
        byte[] data = {65,65,65,65,65};
        exchange.getIn().setBody(data);

        // 送信
        producer.send("file://e:/_work",exchange);
    }
}
```

バイト配列でも★

実際に動くコード

```
public class Test {
    public static void main(String args[]) throws Exception {
        // 初期化
        CamelContext ctx = new DefaultCamelContext();
        ctx.start();
        ProducerTemplate producer = ctx.createProducerTemplate();

        // 送信データの作成
        Exchange exchange = new DefaultExchange(ctx);
        File data = new File("e:/from/test.txt");
        exchange.getIn().setBody(data);

        // 送信
        producer.send("file://e:/_work",exchange);
    }
}
```

File クラスでも★

実際に動くコード

```
public class Test {
    public static void main(String args[]) throws Exception {
        // 初期化
        CamelContext ctx = new DefaultCamelContext();
        ctx.start();
        ProducerTemplate producer = ctx.createProducerTemplate();

        // 送信データの作成
        Exchange exchange = new DefaultExchange(ctx);
        FileInputStream data = new FileInputStream("e:/from/test.txt");
        exchange.getIn().setBody(data);

        // 送信
        producer.send("file://e:/_work",exchange);
    }
}
```

InputStream でも★　送信しちゃいます。

Camel 使わなくても

送信はわりと簡単にプログラムできる

受信はわりと難しい

送信もプロトコル以外の色々な要件が出てくる

受信方法のパターン

- ポーリング型　　取ってくる
- イベント駆動型　　待ち受ける（リスナー型）

受信の要件

受信した内容を後続処理へ渡す

処理開始タイミング

完了処理・ロールバック・結果応答

処理スレッド数

重複メッセージフィルタ

コネクション管理（コネクションプール、再接続）

送信の要件（プロトコル以外）

振り分け（条件によって送信先を変更）

負荷分散（ラウンドロビンなど）

フェールオーバ

流量制限（秒何件まで）

集約（複数メッセージを1つにまとめる）

非同期送信（結果が同一 ID で返ってくる）

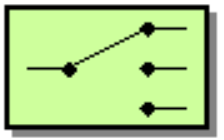
EIP

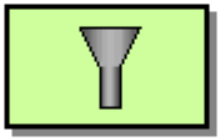
Enterprise Integration Pattern

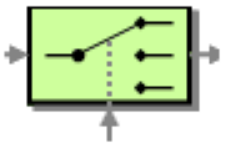
プロトコル以外の要件をまとめたもの

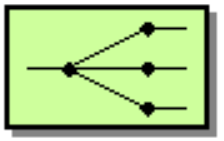
EIP の一部

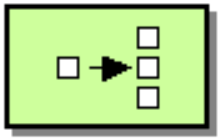
Message Routing

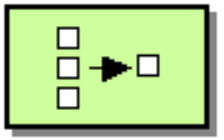

Content Based Router

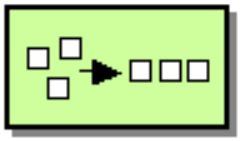

Message Filter


Dynamic Router


Recipient List


Splitter


Aggregator


Resequencer

Apache Camel ホームページ1行目

Apache Camel ™ is a versatile open-source integration framework (with powerful Bean Integration) based on known Enterprise Integration Patterns.

Apache Camel
Enterprise Integration Patterns に基づいている
フレームワークです！

ApacheCamel ３行目

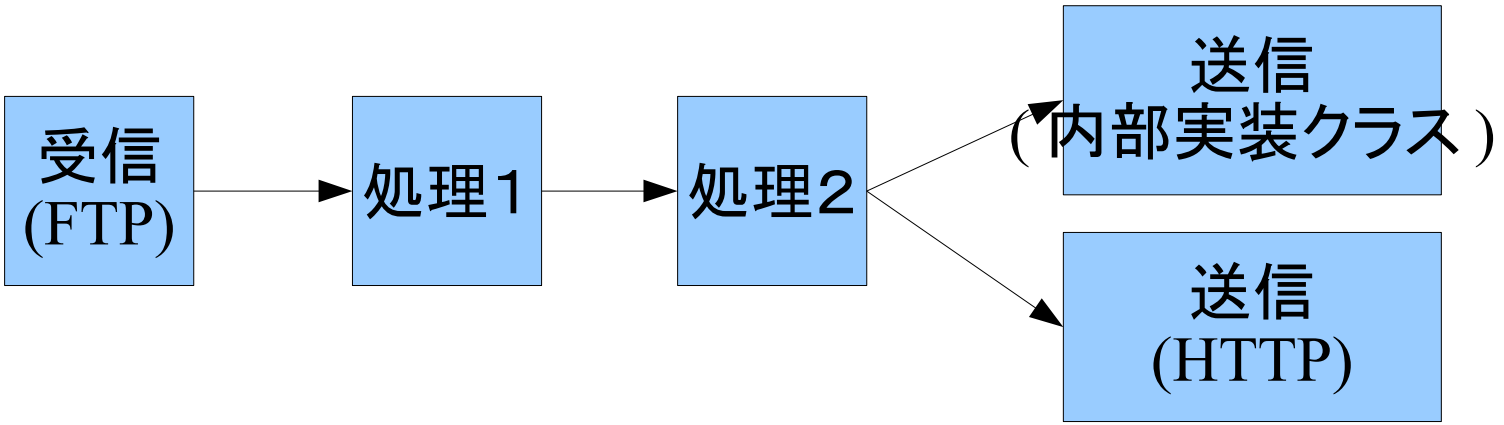
Apache Camel uses URIs to work directly with any kind of Transport or messaging model such as HTTP, ActiveMQ, JMS, JBI, SCA, MINA or CXF, as well as pluggable Components and Data Format options.

Apache Camel は URI を使って色んな種類のプロトコルを扱えます！

EIP を使って受信を実装する

「 Route 」を使う

ルートの概要



ApacheCamel 2行目

Camel empowers you to define routing and mediation rules in a variety of domain-specific languages, including a Java-based Fluent API, Spring or Blueprint XML Configuration files, and a Scala DSL.

Camel は Java や Spring や BlueprintXML や Scala DSL を使ってルートを定義する事ができます。

実際にどうやるか？

ルートの詳細はまた別の機会に

例: Google App Engine で動作

```
import org.apache.camel.CamelContext;
import org.apache.camel.builder.RouteBuilder;
import org.apache.camel.impl.DefaultCamelContext;
import org.apache.camel.impl.SimpleRegistry;

public class MyServletCtxListener implements ServletContextListener{
    public void contextDestroyed(ServletContextEvent arg0) {}

    public void contextInitialized(ServletContextEvent arg0) {
        CamelContext context = new DefaultCamelContext(new SimpleRegistry());
        context.disableJMX();
        try {
            context.addRoutes(new RouteBuilder() {
                @Override
                public void configure() throws Exception {
                    from("ghttp:///greeting?matchOnUriPrefix=true")
                    .transform().constant("Hello");
                }
            });
            context.start();
            System.out.println("- started -");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

例：Google App Engine(web.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">

    <listener>
        <listener-class>jp.jacug.gae1.MyServletCtxListener</listener-class>
    </listener>

    <servlet>
        <servlet-name>CamelServlet</servlet-name>
        <servlet-class>org.apache.camel.component.servlet.CamelHttpTransportServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>CamelServlet</servlet-name>
        <url-pattern>/camel/*</url-pattern>
    </servlet-mapping>

</web-app>
```