B-036

Plagger サーバの構築

Plagger Server

坂田 真幸† 奥野 拓†
Masayuki Sakata Taku Okuno

1. はじめに

現在，インターネット上には，多数のユーザが利用している Web アプリケーションが存在しているが，これらの提供しているサービスはさまざまなデータ形式があり，ユーザが必要とする情報やデータだけを抽出し，管理することは難しくなっている．

このような中，Plagger というソフトウェアが公開されている[1][2]．このソフトウェアはクライアント PC 上で動作し，複数のプラグインを組み合わせて利用することで，Web 上のリソースをさまざまな形式に出力することができる．

しかし現状では，インストールや実行するための設定が複雑であるなど，容易に利用できるソフトウェアとはなっていない．

本研究では，サーバ上で Plagger を動作させることでインストールを不要にし，設定を支援するシステムを構築する．

2. Plagger

Plagger とは，Pluggable RSS/Atom aggregator の略で，オープンソースの Perl フレームワークである．

Plagger はレシピと呼ばれる設定ファイルに記述されているプラグインを順番に実行するソフトウェアである．

Plagger のプラグインには，RSS などの構造化されたデータや，構造化されていないデータをスクレイピングなどの方法を用いて取得する入力系プラグイン，データ形式の変更やフィルタリング処理を行うフィルタ系プラグイン，データを特定の Web アプリケーションやソフトウェアに出力する出力系プラグインの 3 種類のプラグインがある．

これら 3 種類のプラグインを組み合わせて実行することで，さまざまな Web サイトから必要な情報を収集し，加工して出力することができる．

例えば，入力系プラグインとして Bloglines プラグインを使い，RSS リーダの一つである Bloglines から未読フィードを取得，HTML メールに加工し，出力系プラグインとして Gmail プラグインを使い，Web メールの一つである Gmail へ送信し，Gmail で管理することができる．

しかし現状では誰でも手軽に利用できるものではなく，以下のような不便な点がある．

1. 数十もの依存モジュールが存在し，インストールが複雑である．
2. 利用するプラグインに必要なパラメータを理解し，設定ファイルを記述する必要がある．
3. コマンドでのみ実行可能であり，コマンド操作に慣れていない人には使いにくい．
4. レシピが YAML[3]という形式で記述されているため，YAML の知識が必要である．

†公立はこだて未来大学

3. Plagger サーバの構築

3. 1 アプローチ

本研究では上記の Plagger の不便な点を改善するために，以下のアプローチを提案する．

1. Plagger をサーバ上で動作させることで，インストールを不要にする．
2. ユーザによって設定され，正常に実行された Plagger のレシピを蓄積する．
3. プラグインごとのパラメータ設定を行う際に，パラメータ記述例を表示し，過去に何度も同じ値が入力されているパラメータは表示せず，設定の手間を軽減する．

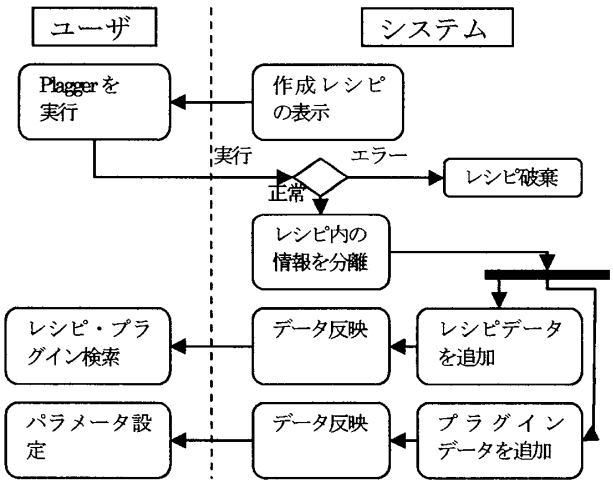

図 1. レシピ再利用のアクティビティ図

3. 2 レシピとパラメータの再利用

ユーザがレシピを作成し，Plagger を実行をさせると，作成されたレシピ通りにプラグインが正常に動作するかシステム内で判定する．

そして正常に動作した際には，レシピを蓄積し，そのレシピ内の情報をレシピデータとプラグインデータの 2 種類に分ける(図 1)．

このプロセスで作成されたレシピデータをシステム内で利用することにより，ユーザは 1 つのレシピを選択するだけでよく，複数のプラグインを選択する手間を軽減できる．プラグインファイルをシステム内で作成することで，ユーザがパラメータ設定時に入力するパラメータ数を減らし，入力の手間を軽減できる．

3. 3 画面構成と操作手順

本システムは，ユーザインタフェースにオープンソースのリッチクライアントフレームワークである OpenLaszlo[4]を用いて実装した．それによって，ユーザは画面遷移することなく，Plagger の設定から実行までを行うことができる．

ユーザインタフェースは図 2 に示すように 3 つのペインから構成され，ユーザは以下のペイン順に操作を行う．

(1)検索ペイン

利用したい Web アプリケーション名やファイル形式名を入力し，検索ボタンを押すと，検索結果にレシピ名とプラグイン名のリストが表示される．レシピを選択することで，そのレシピに使用されているすべてのプラグインがレシピ帳ペインに表示される．

利用したいレシピがない場合には，1 つずつプラグインを選択することで，それぞれのプラグインがレシピ帳ペインに追加される．

(2)レシピ帳ペイン

ユーザはレシピ帳ペインに表示されているプラグインごとにパラメータ設定ボタンを押し，パラメータ入力ペインからパラメータを入力する．

そして，レシピ帳ペインにある Plagger 実行ボタンを押すことで，自動的に設定内容のレシピが作成され，Plagger が実行される．

(3) パラメータ設定ペイン

プラグインごとのパラメータの既定値が初期値として入力フォーム内に表示され，必要に応じてパラメータを変更し，設定ボタンを押す．

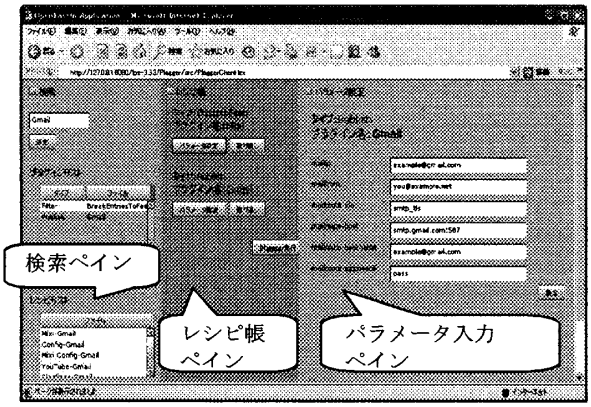

図 2. システムのユーザインタフェース

4. 実験

4. 1 目的と方法

本システムを用いることで，パラメータの入力数をどの程度減らすことができるかについて実験を行った．

そのために，7 種類のプラグインを用い，5 通りのレシピを作成した．

これらのレシピごとに実行に必要な最低限のパラメータを入力し，実行させることで，入力するパラメータ数がどのように減少するかを調べた．

表 1 は使用したレシピである．

4. 2 結果と考察

表 1 のレシピを A から E の順序で 2 度実行させた結果がグラフ 1 である．

図 3 から，1 回目の実行時には 5〜9 であったパラメータ数が，2 回目の実行時においては，5〜7 とパラメータ数が確実に減少していることがわかる．

また，例としてレシピ A の設定時に表示されたパラメータを表 2 に示す．表示されないパラメータはユーザの個人情報に関わらないパラメータや Gmail のホスト名のパラメータなど，必ずしも変更する必要がないパラメータである．

表 1. レシピごとの利用プラグイン名

レシピ名	入力系	フィルタ系	出力系
A	Bloglines	BloglinesSubscription	Gmail
B	Mixi	-	Gmail
C	YouTube	-	Gmail
D	Config	-	Gmail
E	Frepa	-	Gmail

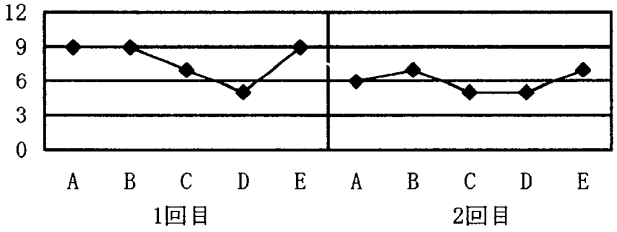

図3. 設定時に表示されるパラメータ数の比較

表 2. レシピ A 設定時に表示されるパラメータ

プラグイン	パラメータ	1 回目	2 回目
Bloglines	username	○	○
	password	○	○
	mark_read	○	
Gmail	mailto	○	○
	mailfrom	○	○
	mailroute(via)	○	
	mailroute(host)	○	
	mailroute(username)	○	○
	mailroute(password)	○	○

5. おわりに

本システムを利用することにより，コマンド操作に慣れていないユーザも Plagger を利用して必要な情報だけを抽出し，管理することができると考えられる．

参考文献

[1] http://plagger.org/trac.
[2] http://www.otsunecom/fswiki/plagger.html.
[3] http://jp.rubyist.net/magazine/?0009-YAML.
[4] http://laszlo.jp/.