

## コンテンツとサービスを対象としたマッシュアップシステムの開発

西村紅美<sup>†</sup> 塚本享治<sup>†</sup>東京工科大学メディア学部メディア学科<sup>†</sup>

## 1. はじめに

インターネット上に散在するコンテンツを組み合わせ、付加価値をつけて提供するマッシュアップ技術が注目を集めている。マッシュアップでは、マッシュアップ要素間のインターフェースとそのフレームワークが重要である。このことを念頭において、マッシュアップシステムを開発したので報告する。

## 2. マッシュアップシステムの設計

コンテンツとサービスを利用して簡単にマッシュアップを行えるように、サービスの組み合わせ方をそのままコーディングすることを目指した。対象とするコンテンツとサービスは以下の通りである。

- 文書や画文書や画像などのデジタルコンテンツ
- SOAP や REST などのプロトコルを利用しているソフトウェアの機能

これらをコンポーネント化して組み合わせるためにとった解決方法は次の通りである。

- (1) サービスの組み合わせ方は GUI 環境でフロー図として描いた。そして、このフロー図を元にシステム制御のための設定ファイルを作成した。
- (2) サービスを組み合わせるためのインターフェースを、既存のマッシュアップツール[2]を利用して作成した。
- (3) 本システムで扱うデータ形式を統一するため、取得したコンテンツのデータ形式を変換するラッパーを作成した。
- (4) コードジェネレータ[1]を利用して、(1)で描いたフロー図を元にシステム設定ファイルを生成した。

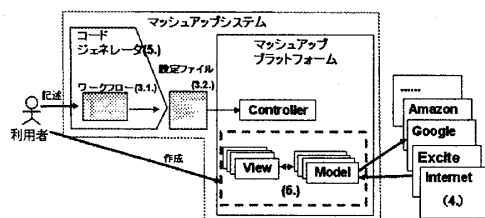


図 1 全体構成

Development of the mash up system for the contents and service

<sup>†</sup>Kumi Nishimura, Michiharu Tsukamoto

<sup>†</sup>Tokyo University of Technology School of Media Science

図中の番号は、以降に詳細を説明している章番号である。

## 3. サービスのフロー記述

## 3.1. フロー記述

コンテンツとサービスを複雑に組み合わせたサービスの設計のフロー図を、グラフィカルエディタである JaWE[3]に入力した。入力したものは、使用するサービス名と組み合わせ方、サービス毎に設定するパラメータである。JaWE でフロー記述を入力すると、XPDL[4]で記述されたファイルが自動生成される。

## 3.2. マッシュアップ設定ファイル

3.1で入力したフロー図から自動生成された XPDL ファイルを元に、2の(4)で述べたコードジェネレータを用いてマッシュアップ設定ファイルを作成した(図2)。

```
<?xml version="1.0" encoding="UTF-8" ?>
<mc:config>
  <form-beans>
    <form-bean name="FormBean" class="app.FormBean" />
    <form-bean name="MashUpActionBean" class="app.MashUpActionBean" />
  </form-beans>
  <global-forwards>
    <forward name="Submit" path="/Submit.jsp" />
    <forward name="MashUpAction" path="/MashUpAction.do" />
  </global-forwards>
  <action-mappings>
    <action path="/MashUpAction" class="app.MashUpAction" name="MashUpActionBean">
      <scope>request</scope>
      <input>/pages/MashUpAction.jsp</input>
      <errorPath>/MashUpActionError.jsp</errorPath>
      <forward name="success" path="/pages/Submit.jsp" />
      <forward name="failure" path="/pages/Failure.jsp" />
    </action>
    <action path="/GakkoAction" class="app.GakkoAction" name="GakkoActionBean">
      <scope>request</scope>
      <input>/pages/GakkoAction.jsp</input>
      <errorPath>/GakkoActionError.jsp</errorPath>
      <forward name="success" path="/pages/GakkoAction.do" />
      <forward name="failure" path="/pages/Failure.jsp" />
    </action>
    <action path="/GakkoAction" class="app.GakkoAction" name="GakkoActionBean">
      <scope>request</scope>
      <input>/pages/GakkoAction.jsp</input>
      <errorPath>/GakkoActionError.jsp</errorPath>
      <parameter name="mode" value="div" />
      <forward name="success" path="/pages/GakkoAction.do" />
      <forward name="failure" path="/pages/Failure.jsp" />
    </action>
    <action path="/Show" class="app.Show" parameter="/pages/Show.jsp" />
    <action path="/Failure" class="app.Failure" parameter="/pages/Failure.jsp" />
  </action-mappings>
</mc:config>
```

図 2 マッシュアップ設定ファイル

## 4. コンテンツのデータ形式

コンテンツのデータ形式は、HTML や XML といった複数の形式で書かれており、文字コードもまちまちであるため、本システム内で扱う際にはデータ形式を統一する必要がある。そこで、ラッパーを作成することで解決を図った。コンテンツのデータ形式を変換するフローは図3の通りである。

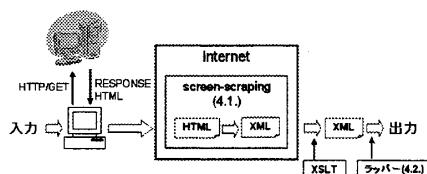


図 3 データ形式変換のフロー

#### 4.1. スクリーンスクレイピング

コンテンツの多くが HTML であることから、HTML から必要な情報のみ抽出する必要がある (Screen Scraping)。しかし HTML は仕様に沿っていないものが多く、レイアウトのための情報も多い。そこで、まず HTML を XML に変換することで仕様に曖昧な部分を解消する。次に、この XML を変換し、情報を整理する。なお、この処理の際に文字コードを UTF-8 に統一する。

#### 4.2. コンテンツ取得後の変換

4.1でコンテンツから必要な情報を取得した後、ラッパーを用いて、サービスが扱うコンテンツのデータ形式に変換する。ラッパーは表形式のもの、文章形式のもの、順序を伴うものと3種類を作成し、サービスの入力形式にあわせて使い分けた。

#### 5. システムの実装

使用するコードジェネレータは、Struts を用いたシステム開発を想定して作られている。このコードジェネレータを用いるために、JaWE から自動生成した XPDL ファイルを、途中ワークフロー言語変換を経て、この定義ファイル (struts-config.xml) に自動変換した。図中の番号は、以降に詳細を説明している章節番号である。

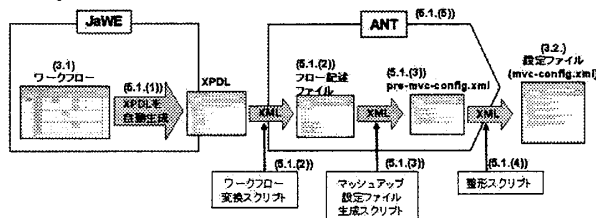


図4 コードジェネレータの構成

##### (1) XPDL ファイルの生成

3.1で述べたように、フロー図を、JaWE を用いて記述すると、XPDL で記述されたファイルが自動生成される。

##### (2) フロー記述ファイルの生成

(1)で生成した XPDL ファイルをフロー記述ファイルに変換するために XSLT で記述した「ワークフロー変換スクリプト」を作成した。

##### (3) pre-mvc-config.xml の生成

(2)で生成されたフロー記述ファイルをマッシュアップ設定ファイルに変換するために、2回 XSLT で記述したスクリプトを用いて生成する。最初に用いる XSLT スクリプトである「マッシュアップ設定ファイル生成スクリプト」を作成した。この XSLT スクリプトで変換をおこなうと、「pre-mvc-config.xml」が生成される。

#### (4) マッシュアップ設定ファイルの生成

(3)で生成された「pre-mvc-config.xml」から、XSLT で記述した「整形スクリプト」によって、すでに3.2で述べたようにマッシュアップ設定ファイルが生成される。

#### (5) ANT による変換スクリプト

(2)から(4)までの一連の流れを ANT スクリプトとして記述した。

#### 6. 検証

5で生成した設定ファイル (図2) から、システムを制御する実行環境を開発した。MVC モデル2を用いて、サービスへの問い合わせ部分やシステム制御部分などのコンポーネント化を行った。

本システムの有効性を検証するために、実際にサービスを以下の手順で構築した。構築したサービスは、フローが単純な場合と繰り返し処理などを含む複雑な場合の2種類である。

まず、2の(4)で述べたコードジェネレータを用いてマッシュアップ設定ファイルを生成した。次に、マッシュアップ設定ファイルにあわせて、部品を作成した。部品とは、サービスにアクセスするための JavaBeans、コンテンツを受け渡すための JavaBeans、ラッパー、JSP である。本システムでは、マッシュアップ設定ファイルを用いてサービスの組み合わせ方を制御した。

複雑な処理を含む場合でも、3.2のマッシュアップ設定ファイルで定義したとおりの動作を確認できた。しかし、検証に取り上げたサービスは、扱うコンテンツのデータ形式が同じ場合が多かったため、作成したラッパーの種類は少なかった。より多くのサービスに対応するためには、ラッパーの種類を増やす必要がある。

#### 7. おわりに

以上のように、本システムの有効性が確認できた。今後は、対応するサービスとコンテンツの幅を増やし、完成度を高めたい。

#### 参考文献

- [1]床田, 塚本, ワークフロー図から Struts 定義ファイルの生成, 第69回全国大会論文, 2007
- [2]山本, 小山, 杉田, 塚本, マッシュアップツールの開発とそれを用いた東京都防災マップの構築, 第69回全国大会論文, 2007
- [3]JaWE, <http://jawe.objectweb.org/>
- [4]XPDL, <http://www.wfmc.org/standards/XPDL.htm>