社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－SE－151（1）
2006／3／23

Web アプリケーションの Web サービス変換

高橋　健一[†]　立堀　道昭[††]　紫合　治[†††]
† 東京電機大学 情報環境学研究科
†† 日本IBM 東京基礎研究所
††† 東京電機大学 情報環境学部

Web アプリケーションを再利用し Web サービスとして公開することは，開発コストの面からも大変有効である．本論文では，ページ遷移単位の Web サービス変換と，セッション情報のチェーンによる Web アプリケーション変換アーキテクチャを提案する．ページ遷移単位で変換された Web サービスを BPEL のようなフロー記述言語で繋ぐことにより，Web アプリケーションの提供するユースケース単位での Web サービス公開と，拡張や変更に柔軟に対応可能な Web サービス変換を実現した．さらに，本アーキテクチャを実装した Web サービス変換フレームワーク「H2W」を開発した．最後に，実際に運用されている 3 種の Web アプリケーションに対して適用し，Web サービス変換に成功した．

Converting Web Services to Web Applications

Kenichi Takahashi[†], Michiaki Tatsubori[††] and Osamu Shigo[†††]
† Graduate School of Information Environment, Tokyo Denki University
†† IBM Tokyo Research Laboratory
††† School of Information Environment, Tokyo Denki University

Developing Web Services from scratch is costly task. Converting Web Applications to Web Services are considered one of the most effective methods for Developing Web Services. This paper describes a framework named H2W that can be used for constructing Web Service Wrappers from existing Web Applications. We propose a page-transition-based decomposition model and page access model with context propagation. We have successfully developed Web Service wrapper applications for three real world Web Applications.

1. はじめに

近年，分散コンポーネント技術であるWebサービスが盛んに研究され，実用段階を迎えつつある．Webサービスでは，XMLベースの標準プロトコルを用いることで高い相互運用性を実現することが出来る．しかし，Webサービスを用いたシステム開発には多くのコストがかかることが問題とされている．

新規システムの開発コストを削減する有効な方法の一つは，既存のシステムを再利用することである．我々は，現在様々なシステムがWebアプリケーションとして開発・利用さていることに注目し，既存のWebアプリケーションをラッピングしてWebサービスへと変換することで，Webサービス開発におけるコストを抑えることができると考えた．

Webアプリケーションをラッピングして変換するための方式は，運用中のWebサーバに変換システムを搭載する方法や，クライアントマシン上で受信したデータを変換する方式などがあるが，プロキシサーバを用いる方法がもっと有効であると考えられる．プロキシサーバがWebアプリケーションにアクセスすることで，十分なテストを行ったアプリケーションロジックや，運用に耐えうるセキュリティ機能を有効に利用できる．また，別のサーバ上で動作させることも可能であるから，既存システムへの影響を最小に抑えることができる．また，クライアントマシンに新しいアプリケーションを導入する必要もない．

しかし，Webアプリケーションをラッピングするプロキシサーバを，実用的なWebアプリケーションに適用する際には，次の2点が課題となる．

- Webブラウザがレンダリングすることで，人間が読みやすいように記述されているHTMLから，Webサービスのレスポンスするべき重要なデータを抽出する．
- 複数のWebページを移動しながら要求を満たす対話型のアプリケーションを，単一のリクエストによって結果が返されるアプリケーションへと変換す

る.

1つ目の課題を解決するために，Webマイニングの分野ではHTMLから重要なデータを抽出するWebラッパーや，Web Data Extraction Toolと呼ばれる技術が盛んに研究されている．例えば，Noga等[1]は，単一のリクエストによって得られたWEBページからデータを抽出し，WEBサービスで提供することが出来る方式を提案している．Stefan等[2]は，テキストからデータ抽出を行うラッパーの調査と，LAPISによるラッパー生成の提案を行っている．Wang等[3]は，データ抽出ラッパーの生成法として，あるスキーマに従って動的に生成される複数のWEBページからそのスキーマを導出し，適切なデータを抽出する方式を提案している．Gupta等[4]は，非有用コンテンツを削除することで，結果的に有用なコンテンツを抽出するためのラッパー生成法を提案している．さらにLeander等[6]は，Web上のコンテンツからデータを抽出するツールを調査し，6種類に分類・評価を行った．

上記のようなWebラッパーの技術を用いることで，HTMLから重要な情報を抽出することは十分に可能となっている．しかしながら，対話型というWebアプリケーションの特性を変換するという課題は未だ解決されていない．

そこで我々は，複雑なWebページの遷移を伴うWebアプリケーションを，Webサービスへと変換するためのアーキテクチャを提案し，そのアーキテクチャを実装したフレームワークの開発を本研究の目的とした．

本論文では，まず2章でWebアプリケーションの特性の分析を行う．その後，Webアプリケーションの特性を考慮したWebサービス変換アーキテクチャを示し（3章），我々が開発したフレームワークについて述べる（4章）．5章では，実際に運用されている3つのWebアプリケーションに対してWebサービス変換を行った結果を紹介する．最後に，6章で今後の課題を，7章でまとめを述べる．

2. Web アプリケーション

Webアプリケーションは，Webクライアント（Webブラウザ）がWebサーバと通信を行い，ネットワーク経由でアプリケーションの機能を提供するネットワークアプリケーションである．我々は，Webアプリケーションの特性を明らかにするために，「ページの遷移」というミクロな視点と，「提供するサービス」というマクロな視点の2つの観点からWebアプリケーションを分析した．

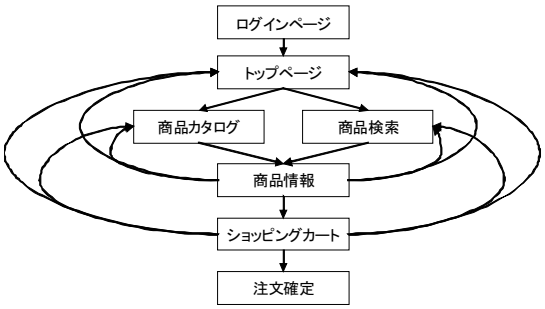

図1 ショッピングアプリケーションのページフロー

2.1. ページ遷移

Webアプリケーションは，Webクライアントを用いてWebサーバと対話を行うインタラクティブなアプリケーションである．ユーザは，<a>タグで表現されるWebページ上のハイパーリンクや，<form>タグで表現される入力フォームやボタンを利用して，Webサーバへリクエストを送信すし，Webサーバはレスポンスとして次のWebページを返却する．ユーザはこの操作を繰り返すことで，複数のページを移動しながらWebアプリケーションの提供する機能を利用する．本研究では，このWebページの移動を「ページ遷移」と定義する．

Webアプリケーションは，Webページをレスポンスとして返す際に，クライアントから受信したリクエストが妥当なものであるかどうかを検証している．妥当でないと判断された場合は，HTTPプロトコルのエラーコードやエラーページをクライアントに返却する．最も基本的な判断基準はURI（Uniform Resource Identifiers）である．また，フォームを利用したページ遷移では，入力されたパラメータの妥当性が検証される．さらに，多くの実用的なWebアプリケーションでは，Cookieを用いたセッション管理が行われる．我々は，この3点をページ遷移に必要な「セッション情報」と定義した．

正しい「ページ遷移」を行うためには，正しい「セッション情報」を送信することが必要となる．

2.2. Web アプリケーションにおけるサービス

Webアプリケーションは、前述のページ遷移を組み合わせることでユーザの要求を満たす．例えば，ショッピングアプリケーションでは，図1のようにページ遷移を組み合わせて，「商品購入」という要求を満たしている．我々は，このページ遷移を組み合わせて提供されるWebアプリケーションの機能を，Webアプリケーションの提供する「サービス」と呼ぶ．

図1のアプリケーションを利用するユーザは，ログインページからトップページへ移動し，商品を探すために，「商品

カタログ」か，キーワードによる検索を行う「商品検索」のどちらかのハイパーリンクを選択する．次に，希望の商品を選び，ショッピングカートに追加，注文処理を行って，注文確定画面へ到達し，購入処理を完了する．

Webアプリケーションを利用するユーザは，ある決まった経路を通ることで，提供されている機能を利用することが出来る．もし，ユーザがトップページから注文確定画面へ直接進もうとしても，Webアプリケーションはエラーメッセージを表示して，不正な処理であることを通知するはずである．Webアプリケーションの提供する「サービス」を利用する場合には，正しい経路を通り，正しいセッション情報を取得しなければならない．

3. Web サービス変換アーキテクチャ

本項では，我々の提案するWebサービス変換アーキテクチャについて述べる．

1章で，対話型であるというWebアプリケーションの特性を変換することは困難であると述べた．その理由の一つは「セッション情報管理の困難さ」であると考えられる．Apache HttpClient[6]のような，ローレベルなHTTPクライアントであれば，セッション情報の管理は簡単に行うことが出来るが，単体ではサービスの経路に合わせた非常に静的な変換しか行うことが出来ないため，変更や拡張に対してとても弱くなってしまう．

そこで我々は，ページ遷移を実行する「コンポーネントサービス」と，ワークフローとしてコンポーネントサービスをサービスの単位にまとめる「コンポジットサービス」の2種類のWebサービスを使った変換アーキテクチャを提案する（図2）．

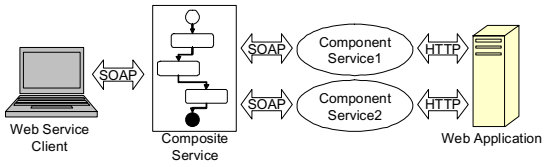

図2 2層構造によるWebサービス変換アーキテクチャ

3.1. コンポーネントサービス

コンポーネントサービスは，Webアプリケーションのページ遷移単位でモジュール化されたWebサービスである．すなわち，特定のページから次のページへ遷移する動作がそれぞれコンポーネントサービスとして定義される．

図3の例では，ログインページからトップページへのページ遷移がloginという一つのコンポーネントサービスとなり，トップページから商品カタログへ遷移がgetCatalogというコンポーネントサービスになる．他のページ遷移についても同様である．

入力フォームのあるページから，次のページへ遷移するためのコンポーネントサービスには，入力パラメータを渡す必要がある．ログインページに，ユーザIDとパスワードの入力フォームがあるならば，loginサービスにも同様に，ユーザIDとパスワードをパラメータとして渡す必要がある．また，サービスの重要な結果を表示するようなページへ遷移するコンポーネントサービスには，データの返却を行う必要がある．注文確定画面に進むbuyサービスでは，遷移後のページに存在する配送予想日を変伽する．

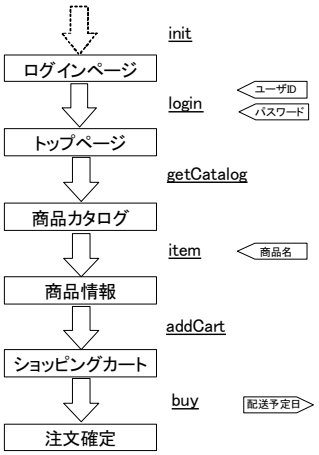

図3 ページ遷移とコンポーネントサービス

3.2. コンポジットサービス

コンポジットサービスは，複数のコンポーネントサービスをWebアプリケーションの提供するサービス単位にまとめたものである．図3の例では，init，login，getCatalog，item，addCart，buyというフローがShopサービスのようなコンポジットサービスとして構築される．コンポジットサービスを利用するクライアントは，送信するリクエスト内に，フロー内のコンポーネントサービスで利用されるすべての入力パラメータを含めなければならない．図3の例では，ユーザIDとパスワード，商品名である．コンポジットサービスは，呼び出すコンポーネントサービスに応じたパラメータを，受信したリクエストの中から選択し適切に送信を行う．また，最終的な出力データ（例では配送予定日）をレスポンスとしてクライアントに返却する．

4. H2W フレームワークの実装

我々は，3章で提案したWebサービス変換アーキテク

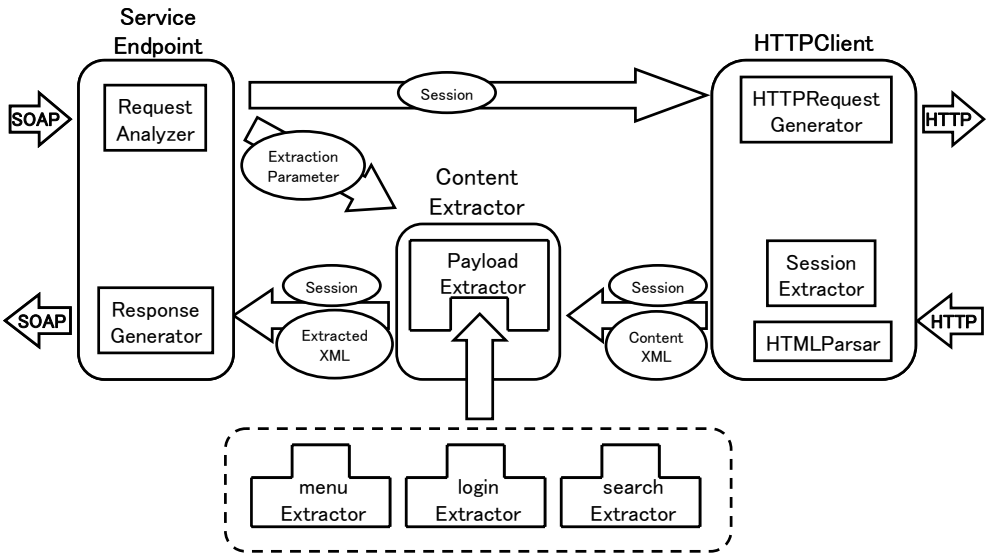

図4 コンポーネントサービスの実装

チャを実装したフレームワーク「H2W」を開発した．H2W フレームワークは，オープンソースプロダクトとして開発されており，以下の Web サイトからダウンロードすることが出来る．また，5 章で紹介するサンプルアプリケーションを利用することも可能である．

http://h2w.sourceforge.jp/

H2W フレームワークは，コンポーネントサービスの実現するためのランタイムと，開発ツールから構成される．コンポジットサービスについては，BPEL4WS のような十分な機能を持ったフロー記述言語と，その実行環境があるため，本フレームワークのサポート対象とはしなかった．

本項では，始めにコンポーネントサービスの実装の概要を紹介する．次に，コンポーネントサービスの特徴である，セッション情報を抽出・伝播するための仕組みと，コンテンツ抽出プラグインについて述べる．さらに，コンポーネントサービスの開発を支援するサポートツールについて述べ，最後にコンポジットサービスの実現方法について述べる．

4.1. コンポーネントサービス実装の概要

コンポーネントサービスのランタイムの構成図を図4に示す．ランタイムは3つのモジュールから構成される．

Service Endpointモジュールは，コンポジットサービスとのインタフェースとなるモジュールである．Webサービスの実装としてApache Axix[7]を利用している．

HTTP Clientモジュールは，Webサーバとの通信を行うモジュールである．また，レスポンスの情報の解析や，HTMLのパース処理も行う．HTMLは，閉じタグの欠如など，厳密なツリー構造になっていない場合が多い．HTMLパーサは，不完全なツリー構造を保管しながらXMLのインタフェースでHTMLを処理することを可能にするライブラリである．なお，HTTPクライアントとして，Apache HttpClientを，HTMLパーサとして，NekoHTML[8]を使用している．

Content Extractorモジュールは，HTMLから重要な情報を抽出するためのモジュールである．HTTP Clientモジュールで XML形式にパースされたHTMLを処理し，必要な情報を抽出する．

コンポジットサービスからのリクエスト開発者が作成しなければならないのはService Endpointと各ページに対応するPayload Extractor（login Extractorやsearch Extractorなど）だけである．さらに，Service Endpointは後述のサポートツールによって生成することが可能である．

4.2. セッション情報の抽出と伝播

コンポーネントサービスはWebアプリケーションのページ遷移をラッピングするサービスであり，正しいページ遷移を行うためにセッション情報の抽出と伝播を行う必要がある．

まずセッション情報抽出の仕組みについて述べる．セッション情報の抽出は，HTTP Client内のSession Extractorで行われる．始めに，HTML内のハイパーリンク（<a>タグや<form>タグなど）から，遷移先のURIを抽出する．aタグならばhref属性，formタグならばaction属性の値が対象のURIとなる．次に，formタグ内のパラメータを抽出する．ここでは，type属性がhiddenのinputタグの値を抽出する．最後に，HTTPヘッダからCookieの情報を抽出し，3点をまとめることでセッション情報の抽出が完了する．

次に，セッション情報の伝播について述べる．抽出されたセッション情報は，Service Endpointでコンテンツ情報と共にコンポジットサービスへ返却される．コンポジットサービスは，次のコンポーネントサービスに対して，受け取ったセッション情報をそのまま送信することで，セッション情報を伝播することが可能となる．

4.3. コンテンツ抽出プラグイン

本フレームワークでは，Webページから必要なコンテンツ情報を抽出するために，まずHTMLをDOMツリーへとパースする．DOMツリーへとパースされたHTMLは，Content Extractorで必要なコンテンツ情報を抽出される．DOMを用いたコンテンツ抽出の手法は，Gupta等が既に十分な研究を行っているため，我々は既存のコンテンツ抽出システムをプラグインとして組み込むという選択をした．現在，Content Extractorでは2つの抽出手法を選択することが出来る．1つ目は，Javaで直接DOMツリーを処理する方式であり，2つ目はXSLによって処理する方式である．

例えば，XSLを用いたコンテンツ抽出を行うためには，図5のようなXSLスタイルシートを記述する．この例では，/HTML/BODY/TABLE/TR/TD/UL/LIというXPathが指し示す要素内の，Aタグと，そのタグによってハイパーリンクが張られているテキストをペアで抽出している．また，Javaによるプログラム例も図6に示す．

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" />

  <xsl:template match="/HTML/BODY/TABLE/TR/TD/UL/LI">
    <payLoad>
      <key>
        <!-- Aタグと属性のコピー -->
        <xsl:element name="A">
          <xsl:attribute name="href">
            <xsl:value-of select="concat('https://ds.sie.dendai.ac.jp', ./A/@href)" />
      </xsl:attribute>
        </xsl:element>
      </key>
      <!-- 関連付けるテキストの出力 -->
      <value>
        <xsl:value-of select="./A/text()" />
      </value>
    </payLoad>
    <xsl:text>
    </xsl:text>
  </xsl:template>

  <xsl:template match="text()" />

  <xsl:template match="/">
    <payLoads>
      <xsl:apply-templates/>
    </payLoads>
  </xsl:template>

</xsl:stylesheet>
```

図5 コンテンツ抽出XSL例

```
public class MerchandiseInformationExtractor implements IPayloadExtractor {

    public Document extract(Document htmlDocument) {
        …
        Document document = builder.newDocument();
        Element payLoadsElement = document.createElement("payLoads");
        document.appendChild(payLoadsElement);

        NodeList forms = htmlDocument.getElementsByTagName("form");

        for (int i = 0; i < forms.getLength(); i++) {

            Element formElement = (Element) forms.item(i);
            if (!formElement.getAttribute("action").equals(
                    "http://order.step.rakuten.co.jp/rms/mall/basket/vc")) {
                continue;
            }
            Element payLoadElement = document.createElement("payLoad");
            Element keyElement = document.createElement("key");
            Element valueElement = document.createElement("value");

            String value = this.extractPayLoad(formElement);

            keyElement.appendChild(this.copyElement(document, formElement));
            valueElement.appendChild(document.createTextNode(value));
            payLoadElement.appendChild(keyElement);
            payLoadElement.appendChild(valueElement);
            payLoadsElement.appendChild(payLoadElement);
        }

        return document;
    }
…
}
```

図6 コンテンツ抽出Javaプログラム例

4.4. 支援ツール

我々は，コンポーネントサービスをより簡単に開発できるように，XML形式の設定ファイルからService EndpointとWSDL，Apache Axisへデプロイするための設定ファイルを生成することが出来るサポートツールを開発した．設定ファイルには，サービス全体の情報を記述するservice要素と，各オペレーションの情報を記述するoperation要素の2種類を記述する．

サービス全体の情報は，service要素のname属性としてサービス名を，service要素の子要素namespaceの値としてサービスのネームスペースを，同様に子要素locationにサービスを公開するロケーションを，子要素uriにWebアプリケーションのURIを記述する．

オペレーション情報では，まずオペレーション名がoperation要素のname属性に記述される．operation要素の子要素として，input要素，param要素，extractor要素があるが，input要素とparam要素はいずれもHTMLのフォームへ入力するパラメータの情報である．input要素として記述されたパラメータは常に同じ値がWebサーバに送信される．param要素として記述されたパラメータはサービスを呼び出すクライアントが送信すべきパラメータである．また，extractor要素ではコンテンツ抽出プラグインとして利用するモジュール名を指定する．

図7に設定ファイルの記述例を，図8に生成されたコンポーネントサービスのソースコード例を示す．

```
<h2w>
    <service name="WineShop">
        <nameSpace>urn:wineshop</nameSpace>
        <location>http://localhost:8080/axis/services</location>
        <uri>http://www.rakuten.co.jp</uri>
    </service>

    <operation name="getCagories">
        <extractor class="generic.XSLTExtractor">
            <param name="xsl">rakuten/category.xsl</param>
        </extractor>
    </operation>
…
    <operation name="getLoginForm">
        <input name="units_0" value="1" />
        <input name="submit" value=" 注文画面へ進む " />
        <extractor class="rakuten.LoginFormExtractor" />
    </operation>

    <operation name="authenticate">
        <param name="userID" input-name="u" />
        <param name="password" input-name="p" />
        <input name="deliver_to" value="me" />
        <input name="submit" value=" 次へ " />
        <extractor class="rakuten.AuthenticateExtractor" />
    </operation>
…
</h2w>
```

図7 コンポーネントサービス生成用設定ファイル例

```
public class WineShop {

    private static final String URI = "http://www.rakuten.co.jp/";

    public WineShop() {
    }
…
    public HTTPData[] getCategories(ApplicationSession session) {
        final services.decomposition.extractor.CategoryExtractor payLoadExtractor
                = new services.decomposition.extractor.CategoryExtractor();
        final ServiceStub invoker = new ServiceStub();
        return invoker.invokeHTTP(session, payLoadExtractor);
    }
…
    public HTTPData[] authenticate(ApplicationSession session, String userID,
            String password) {
        final services.decomposition.extractor.AuthenticateExtractor payLoadExtractor
                = new services.decomposition.extractor.AuthenticateExtractor();

        session.addParameter(new Parameter("u", userID));
        session.addParameter(new Parameter("p", password));

        final ServiceStub invoker = new ServiceStub();
        return invoker.invokeHTTP(session, payLoadExtractor);
    }
…
}
```

図8 サポートツールによって生成されるコンポーネントサービスのソースコード例

4.5. コンポジットサービスの実装方法

コンポジットサービスは，BPEL4WSのようなビジネスプロセス記述言語や，Webサービスクライアントとなるプログラムとして簡単に開発することが出来る．図9にJavaを用いた実装の例を示すが，複数のコンポーネントサービスを順番に呼び出し，最後に必要な情報をサービスの呼び出し元に返却するようなWebサービスを開発すればよい．

5. 適用例

我々は，実運用されている3つのWebアプリケーションに対してH2Wフレームワークを適用し，Webサービスへと変換することに成功した．また，最後に履修管理システムをラッピングしたWebサービスを使ったサンプルアプリケーションを紹介する．

5.1. ショッピングサイト

1つ目のWebアプリケーションは，ショッピングサイト「楽天（http://www.rakuten.co.jp/）」である．楽天は，複数のショップが集まったショッピングサイトであり，ショップを選んでの購入や，商品の検索などの機能を備えている．

始めに，楽天をラッピングしたコンポーネントサービスである，WineShopサービスのオペレーションを示す．

- init
- getCategories
- getSubCategories
- getMerchandiseInformation
- addCart
- getLoginForm
- authenticate
- selectPayment

WineShopサービスに対するコンポジットサービスであるSommelierサービスでは，2つのオペレーションを定義している．年代を入力パラメータとして，最も安いワインの情報を取得するgetCheapestWineと，購入処理までを行うbuyCheapestWineである．getCheapestWineは，init，getCategories，getSubCategoris，getMerchandiseList，getMerchandiseInformationを順番に呼び出す．また，buyCheapestWineでは，その後addCart，getLoginForm，authenticate，selectPaymentを順番に呼び出す．

```
public class Sommelier {
    public String getCheapestWine(String year) throws ServiceException,
            RemoteException {
        WineShopService locator = new WineShopServiceLocator();
        WineShop service = locator.getWineShop();

        ApplicationSession session = service.init();

        HTTPData[] datas = service.getCategories(session);
        for (HTTPData data : datas) {
            if ("ワイン".equals(data.getPayLoad())) {
                session = data.getSession();
            }
        }
        datas = service.getSubCategories(session);
        for (HTTPData data : datas) {
            String years = (String) data.getPayLoad();
            if (years.startsWith(year)) {
                session = data.getSession();
            }
        }
        datas = service.getMerchandiseList(session);
        session = datas[0].getSession();

        datas = service.getMerchandiseInformation(session);
        return (String) datas[0].getPayLoad();
    }
…
}
```

図9 Javaによるコンポジットサービス例

5.2. 履修管理システム

2つ目のWebアプリケーションは，本学で利用されている履修管理システムである「ダイナミックシラバス（http://www.sie.dendai.ac.jp/ds/）」である．ダイナミックシラバスの提供する機能は，講義情報（担当教員，開講日など）の検索と，各セメスターの履修登録である．

以下に，コンポーネントサービスであるSyllabusSerivceのオペレーションを示す．

- init
- getMainMenu
- getSyllabusSearchForm
- searchSyllabus
- getCourseInformation
- getLoginForm
- getEnrollCourseMenu
- getTimeTable

コンポジットサービスであるCompositeSyllabusでは，2つのオペレーションを定義している．searchSyllabusは，学科コードと講義名，担当教員を入力し，講義情報を検索する．getTimeTableは，学生IDとパスワードを入力し，今セメスターの時間割を取得する．searchSyllabusは，init，getMainMenu，getSyllabusSearchForm，searchSyllabus，getCourseInformationを順番に呼び出す．またgetTimeTableでは，init，getMainMenu，getLoginForm，getEnrollCourseMenu，getTimeTalbeを順番に呼び出す．

5.3. オープンソースコミュニティサイト

最後のWebアプリケーションは，オープンソースのプロジェクト管理・コミュニティサイトである「Sourceforge.net（http://sourceforge.net）」である．Sourceforge.netの提供する機能は，オープンソースソフトウェア開発におけるサポートツール（CVSリポジトリ・バグ管理システムなど）の提供である．

以下に，コンポーネントサービスであるSourceforgeServiceのオペレーションを示す．

- init
- getMenu
- getLoginForm
- login
- searchProjects
- getProjectMenu
- getBugList

また，コンポジットサービスであるCompositeSourceforgeでは，1つのオペレーションを定義している．getBugListでは，SourceforgeのIDとパスワード，プロジェクト名，日付を入力することで，指定された日付以降に投稿されたバグのリストを取得する．getBugListは，init，getMenu，getLoginForm，login，searchProject，getProjectMenu，getBugListを順番に呼び出す．

5.4. 履修管理システム Web サービスの応用例

履修管理システムをラッピングしたWebサービス（CompositeSyllabus）を利用して，時間割取得を拡張したWebアプリケーションを開発した．

オリジナルの履修管理システムでは，時間割には科目名と指導教員，単位数が表示されているだけであった．応用アプリケーションでは，上記の項目に加えて，教科書名を同時に表示することが出来る．応用アプリケーションは，まず入力された利用者IDとパスワードでgetTimeTableを呼び出し，時間割を取得する．その後，取得した時間割の情報に含まれる科目名を使いsearchSyllabusを複数買い呼び出す．ここで，教科書名を含む科目の詳細情報を取得する．最後に，時間割と教科書名を合成する．システムの動作例を図10示す．

図10 応用アプリケーション例

6. 今後の課題

6.1. ページ遷移のない Web アプリケーションへの対応

近年，AJAX(Asynchronous JavaScript + XML)を代表とするHTMLを動的に書き換えることでページ遷移がなく，

ユーザの操作性を向上させることが出来る新たな形態のWebアプリケーションが注目を集めている．このようなWebアプリケーションは，バックグラウンドで非同期にサーバと通信を行い，表示されているHTMLを直接書き換えているため，ページが書き換わるタイミングを検知するのも．通信内容をハンドリングすることも難しい．したがって，現状のアプローチではページ遷移のないWebアプリケーションに対応することは難しい．これは今後検討事項とする．

6.2. UI フレームワーク Struts との連携

現在，JavaによるWebアプリケーション開発において，頻繁に利用さているフレームワークの一つにApache Struts[9]がある．StrutsはMVC（Model-View-Controller）アーキテクチャに則ったWebアプリケーション開発を支援するView層のフレームワークである．

Strutsの設定ファイルであるstruts-config.xmlには，Webアプリケーションのページ遷移の情報が記述されているため，struts-config.xmlを読み込むことで，Webサービスラッパーを簡単に構築することが可能であると考えられる．Strutsとの連携が可能になれば，簡単に適用できるアプリケーションが増えるのは明らかであり，今後積極的に取り組みたい．

6.3. Web ページのレイアウト変更に伴う課題

Web ページのデザインやレイアウトの変更は，予想していたよりも頻繁に発生するということが，適用実験中にわかった．現在のコンテンツ抽出プラグインでは，HTML 上の位置から情報を抽出する手法を取っているため，Web ページのレイアウトが変わることでコンテンツ抽出の失敗が起こる．しかしながら，レイアウトやデザインの変更に対して適用できるようなデータ抽出の仕組みは，それだけ抽出の厳密性に欠け，正確な抽出が必要なWeb サービス変換には向いていない．解決策については，今後十分に検討したい．

7. おわりに

本論文では，WebアプリケーションをWebサービスへと変換するためのアーキテクチャの提案と，それを実装したH2Wフレームワークの開発を行った．プロキシサーバ上での2層構造による変換は，Webアプリケーションが実装しているロジックや，セキュリティ機能を有効に再利用することが可能であり，Webサービス開発におけるコスト削減に大きな効果が期待できる．また，2層構造によって，

また，実運用されている3つのアプリケーションに対して適用実験を行い，本方式による変換が十分に有効であることを確認した．

今後は，本フレームワークの改良と，サービス変換アーキテクチャに更なる精錬を加えてゆくつもりである．

参考文献

[1] Markus L. Noga, Max Volkel: From Web Pages to Web Service with wal, NCWS2003

[2] Stefan Kuhlins, Ross Tredwell: Toolkits for Generating Wrappers, Objects, Components, Architectures, Services, and Applications for a Networked World 2002

[3] Jiying Wang, Frederick H. Lochovsky: Data Extraction and Label Assignment for Web Databases, ACM 2003

[4] Suhit Gupta, Gail E. Kaiser, David Neistadt, Peter Grimm: DOM-based Content Extraction of HTML Documents, WWW 2003, pp207-214

[5] Alberto H.F. Leander, Berthier A. Ribeiro-Neto, Altigram S. da Silva, Juliana S. Teixeira: A Brief Survey of Web Data Extraction Tools, SIGMOD Record, 31(2) 84-93(2002).

[6] Apache Software Foundation, Apache HttpClient, Jakarta Project, http://jakarta.apache.org/commons/httpclient/

[7] Apache Software Foundation, Apache Axis, http://www.apache.org/axis/

[8] Andy Clark, CyberNeko HTML Parsar, http://people.apache.org/~andyc/neko/doc/html/

[9] Apache Software Foundation, Apache Struts, http://struts.apache.org/